

Course Book



WEB APPLICATION DEVELOPMENT

DLBCSWAD01

iu

INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

LEARNING OBJECTIVES

The course **Web Application Development** will first provide you with an overview of the history of the creation and technical structure of the internet. After completing the corresponding lessons, you will be familiar with the basic concepts, and especially the essential communication protocols, of the modern “web.”

This course will provide important insights into the typical structure of current web application architectures and trends. It will also give basic knowledge about the structure and design of static web pages. In particular, the main focus is on HTML and CSS.

To simplify the programming of static and dynamic web pages, this course focuses on the use of relevant tools for development and source code management. A detailed explanation of advanced design techniques for static and dynamic web pages, such as CSS Frameworks (e.g., Bootstrap), will prepare you to utilize the advantages of responsive web pages.

Finally, the course teaches the basics of JavaScript, so you can write simple, dynamic web pages, and web page security testing is introduced.

UNIT 1

ARCHITECTURAL FOUNDATIONS

STUDY GOALS

On completion of this unit, you will be able to ...

- describe how and why the internet was created.
- explain the architectural concepts behind web applications.
- define what is meant by internet protocols and URLs.
- assess the quality of web applications.
- discuss important architecture trends and analyze the advantages and drawbacks of these architectural patterns.

1. ARCHITECTURAL FOUNDATIONS

Introduction

The internet is probably the world's largest network and has become an indispensable part of today's society. Since its inception, it has connected the world's continents to an unprecedented extent and has allowed people from many corners of the world to interconnect. The internet has revolutionized how social networks are formed, making it possible for groups to organize themselves. Furthermore, it has made it possible to provide billions of people with fast and efficient access to information, news, and education. However, the internet has its disadvantages. In socio-economic terms, for example, the development of the internet has reduced personal contact between people. Above all, the anonymity of the internet has created new ways and means for criminal structures to establish themselves, for example, data theft, viruses, and cyberattacks.

The internet has also made it possible to efficiently hold enormous amounts of data in databases, analyze this data for patterns and desired information, and process it accordingly. This has led to the concept of **big data**. This concept has also raised data security concerns, which in turn have necessitated a legislative consideration of the internet. Regardless of the perspective from which it is viewed, it is clear that the internet has significantly revolutionized (and will continue to revolutionize) the way we live together. Here, we will examine the technical concepts behind the internet, focusing primarily on the programming of web pages (websites), which are the form of the internet the end user most often sees.

Big data

This refers to the efficient and targeted processing of large volumes of data.

1.1 Structure and History of the Internet

The internet, whose name is a portmanteau of "interconnected network," is best understood as a network that connects many computers (Pomaska, 2005). The classic home network, which connects individual computers in a house, apartment, or company locally, is extended by the internet to a global scale, enabling data to be exchanged across national and continental borders. The computers in a local network are connected to each other via **switches**.

Switches

These are components of a network that establish a connection between different actors in the network.

Router

A router, like a switch, has the ability to establish a connection between different actors in a network. It can also connect a local network with the internet.

Since the internet is a global network, users cannot simply join it; organized access to the internet must be provided. This is typically handled by a provider (known as an internet service provider (ISP)), which provides the end user with both the hardware components needed to access the internet (e.g., a **router**) and the corresponding software for connecting a router to the internet. **Repeaters** make it possible to increase the range of internet access in a local wireless network by amplifying the respective signal.

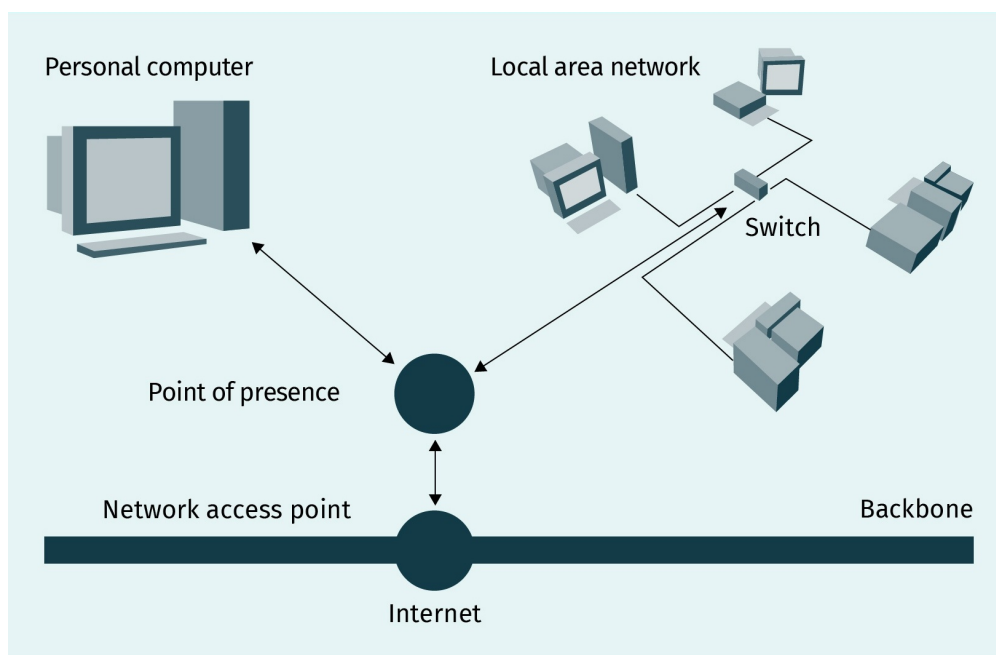
When dialing into the internet, each router receives an internet protocol (IP) address, which can be likened to a telephone number. If we want to call someone, we need their telephone number. The situation is very similar on the internet. Every router and server

needs a unique address assignment in order to be addressed via the internet: this is the function of the IP address. A provider's dial-in node is referred to as a point of presence (POP), which is connected to the provider's backbone network via a network access point (NAP) (Pomaska, 2005). This concept is shown in a simplified form in the following figure.

Repeaters

This is a network actor that has the ability to amplify a weaker signal, either through wires or wirelessly.

Figure 1: Structure of the Internet



Source: Pomaska (2005), p. 2.

The connection to the provider's **core network** is typically made via copper cable or, in more modern cases, via fiber optic connections. The IP address of a computer used on a local home network is converted into a public IP address by the router using a NAT service (network address translation). The local computer's request is then forwarded to the responsible servers (which can also be reached via IP addresses) by means of the public IP address. Once the responsible server has been found, the response (the requested data packet) is forwarded to the public IP address of the computer system that requested it. The router receives this packet, exchanges the public IP address for the local IP of the requesting computer, and forwards it the response.

Core network

The core network is the heart of the internet, which is not directly accessible to the user.

History of the Internet

Like many technologies, the emergence of the internet can be traced back to military applications and dates back to the end of the 1950s, when the internet first appeared in the form of the Advanced Research Projects Agency network (ARPAnet), which sought to create a network between computers (Bühler et al., 2019). Its main aim was to interconnect mainframe computers and enable them to establish data communication over long distances. Fail-safety was also a key consideration here: if a node fails, a large number of other nodes exist that can bridge its functionality.

In 1969, just four mainframe systems were connected in this way using special computers known as interface message processors (IMPs) (Böhringer et al., 2014). In 1971, the ARPAnet, initially intended as a military research project, was presented to the public. At that time, it contained about 15 network nodes. Since the use of the internet for private purposes was not yet feasible at this time, its use was initially limited to military and scientific applications. For security purposes, the military's internet was separated from the civilian one and continued as MILnet. In 1989, the ARPAnet consisted of about 100,000 connected host computers (Böhringer et al., 2014). Instead of continuing to use ARPAnet, the National Science Foundation network (NSFnet) was founded, which was also released for commercial use just one year later.

This network can be seen as the starting point of the internet being used across several countries. However, this version of the internet then did not come close to the data transmission rates that can be achieved today with modern fiber optic technology. At that time, the transmission speed was still limited to 9,600 bps (bits per second). Transferring a cell phone photo with a size of 3 MB (megabytes) would have taken about 43 seconds. The internet can also be traced back to scientific research. For example, HTML (Hypertext Markup Language) was developed by the British computer scientist Tim Berners-Lee at the European Nuclear Research Institute (CERN: *Centre Européen pour la Recherche Nucléaire*) (Böhringer et al., 2014).

The World Wide Web (WWW) service, which was based on HTML and published in 1993, was the first instance of the internet in the form we know today. The first browser to display HTML pages was Mosaic (Böhringer et al., 2014). The development of the internet began in the early 2000s. As of 2018, more than one billion host computers are connected to the internet, as well as approximately twenty billion internet-enabled devices. By comparison, the Earth's population is just about seven billion people. The daily average data volume in 2018 was approximately five exabytes (about five billion gigabytes), which is roughly 12,500 times the number of books that have ever been written (Bühler et al., 2019).

1.2 Internet Protocols and URIs

For individual users of the internet to be able to communicate with each other and exchange data, a common procedure for interacting must be established. This standardization is typically done through **protocols**. The most commonly used protocols on the internet are the internet protocol (IP) and the transmission-control protocol (TCP), the latter of which can be used not only on the internet, but also locally (in private networks). If, for example, a data packet is to be transmitted from Los Angeles to Hamburg, it does not make sense to transmit it in one piece, as this would take up an unnecessary amount of resources for too long (even if the data packet is very small). Instead, the data packet is divided into several smaller packets and sent separately.

A data packet created as a TCP/IP segment typically has a maximum size of 1,500 bytes. IP packets generally allow a maximum size of 65,535 bytes, but are still limited to 1,500 bytes because they are typically transmitted over an Ethernet connection, which operates

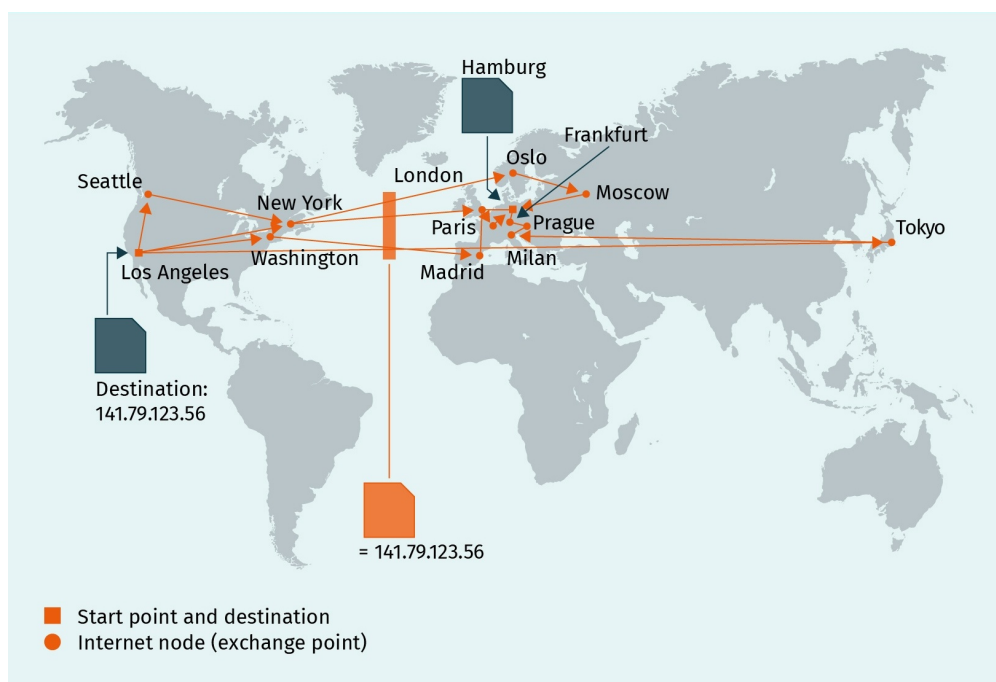
Protocols

These are general rules on how communication between network actors should proceed.

according to the specifications of TCP (i.e., a maximum size of 1,500 bytes). Twenty bytes are reserved for the header of the data packet for both TCP and IP, so that 1,460 bytes are available for the user data of a TCP/IP segment (Wendzel, 2018).

Since each of these data packets must arrive at a computer in Hamburg, it is necessary to provide each data packet with the computer's destination address. This destination address is the IP address and typically has an external form such as 10.155.124.214, i.e., it is divided into four blocks of three digits each, whereby individual blocks assume numerical values from 0 to 255. Leading zeros can be neglected. This is called a 32-bit address. This is an IPv4 address (internet protocol version 4). The forwarding of data packets via different nodes is also referred to as routing. Once they reach the recipient (e.g., 141.79.123.56), the individual data packets are combined back into the original message and displayed to the end user. This procedure is illustrated in the following figure, where the individual data packets are each provided with their destination address (141.79.123.56).

Figure 2: Routing of Data Packets through IP



Source: Bühler et al. (2019), p. 5.

IPv4 versus IPv6

IPv4 addresses are divided into four blocks of three digits each, with leading zeros neglected. Therefore, a total of 4,294,967,296 possible IP addresses can be assigned. This may seem like a large number at first, but consider that more than 20 billion internet-capable devices were in use in 2018 (Vailshery, 2021). Each of these devices is assigned a new IP address when, for example, connecting to the internet, so the supply of possible addresses is exhausted quickly.

Internet of Things

This refers to the networking of physical and virtual objects and their interaction through information and communication technologies.

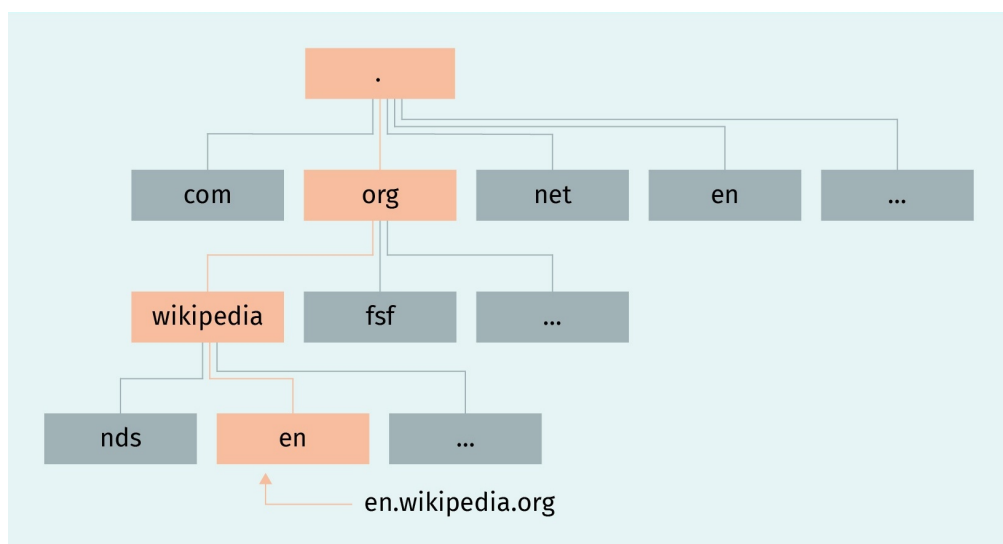
Moreover, **Internet of Things** (IoT) technology remains in its infancy, with many more internet-capable devices being put into operation as home automation and intelligent networking in production become more widespread (this is referred to as industry 4.0). To work around this limited number of IP addresses, it was decided early on to expand the IP space from 32 bits to 128 bits. We are then dealing with IPv6 addresses, which, in contrast to the decimal representation of IPv4, are now specified in a hexadecimal representation, e.g., 2001:0db8:85a3:08d3:1319:8a2e:0370:7344. The address space then increases from 232 addresses (IPv4, 32 bit) to 2128 possible addresses (IPv6, 128 bit). This makes it possible to assign an IP address to a coffee machine or a vacuum cleaner robot. IPv6 allows for 3.4×10^{38} (approximately 340 sextillion) individual addresses (Meyers, 2012).

Domain Name System (DNS)

Each computer and server is given an IP address through which it can be addressed or identified. Since it is somewhat unwieldy to address each server (and the web page made available for query on it) via 32-bit or even 128-bit addresses, we have the domain name system (DNS), which assigns a domain to an IP address. These are the typical “internet addresses” that we enter in the address line of our browser, such as “www.iu.org.” Since this address ends with .org (and could also have endings such as .de or .com), it is referred to as a top-level domain (TLD).

The second-level domain (SLD) is “iu.” The address in this form, “iu.org,” is called a domain. The translation of the entered domain into the IP address of the server on which the website is hosted is handled by the domain name system (DNS), which can be thought of as a kind of database or table with the corresponding assignments. Domain addresses follow a tree structure, as shown in the following figure.

Figure 3: Tree Structure of Domains



Source: TilmannR (2018).

Corresponding subdomains, which are located on a lower hierarchy level, can be assigned to their respective top-level domains, such as “com,” “org,” and “net” in the form en.wikipedia.org.

Since IP addresses can change, especially for private users, dynamic DNS (DDNS) makes it possible to update the assignment database described above so that even a server that does not have a statically assigned IP address can still be associated with a static web address (Bühler et al., 2019). A typical application of this is the operation of a private server at home. This can also be connected to the internet and is accessible via its IP address, but a problem arises from the fact that most privately accessible providers regularly briefly interrupt the internet connection. This results in a reassignment of the IP address, so the server is now no longer accessible under its original IP address. Using DDNS, this dynamic IP address is always mapped to a static IP address under which the server is permanently accessible. However, from a security standpoint, this is where problems arise: it is now easier for attackers to find and attack a server that may have configuration weaknesses.

Uniform Resource Locator (URL), Uniform Resource Name (URN), and Uniform Resource Identifier (URI)

In most cases, entering the domain is still not sufficient, as multiple services (and websites) can run on one server. An address entry of the generic form is referred to as a uniform resource locator (URL). An example of this is as follows:

```
protocol://server.domain-name/folder/file
```

A frequently used protocol is HTTP (hypertext transfer protocol), which also exists in encrypted form: HTTPS. This specifies that the desired call is a request for a hypertext. It would also be possible to make a request via file transfer protocol (FTP), which transfers files from a computer to a server. In the address above, “server” typically takes “www” and the domain could be “iu.” A server is organized like a local computer, so it has a folder structure. Calling the domain first directs us to the server’s “home” directory, which may contain other folders, each separated by a “slash” (i.e., “/”), as follows:

```
https://www.iu.org/bachelor/computer-science/
```

Here, we go through the folder “bachelor” to the folder “computer-science.” When we land in the folder, a file stored there, “index.htm” or “index.html,” is output automatically and displayed in the browser. If we want to output a different file, this is specified in the URL (Bühler et al., 2019).

While a URL specifies a location, the uniform resource name (URN) identifies a resource or a scheme with a freely assigned name. These both fall under the generic term uniform resource identifier (URI) (Pomaska, 2005).

1.3 Web Application Architecture

In simple terms, a web page (which can be understood as one page of a website) displayed in a browser is located as a file on a server. A server is no more than a program that runs on a computer, such as “Apache” or “nginx.” The web page shown is typically found in the basic format for web documents: hypertext markup language (HTML). HTML is a markup language that, in simple terms, describes how a web page should be displayed. The appearance can be separated from the content of a web page by means of cascading style sheets (CSS) (Pomaska, 2005). A CSS is a file that contains different style sheets and determines, for example, how a heading or a paragraph (both written in HTML) should be displayed in the user’s browser. Extensible markup language (XML) can be understood as a metalanguage for the definition of markup languages. The syntax and meaning of individual expressions is defined by means of a document type definition (DTD). XML can also be used to store data that is simply and logically organized, and, if necessary, output it again (e.g., the inventory of a small online store). Data that are stored in XML can be converted into HTML using a blueprint and extensible style language (XSL) (“Pomaska, 2005”).

Web applications themselves can be programmed on the client side, i.e., calculations and the display of a web page are primarily executed in the end user’s browser. This approach supports static web pages in particular, i.e., those that are “fixed” on the server and are not generated and transmitted to the end user on a request-specific basis (dynamically). This contrasts with server-side web pages, which are based on the server performing calculations itself in order to render the web page and database queries. This approach is particularly suitable for implementing **dynamic websites**. Furthermore, hybrid approaches can also be implemented, which are a combination of the described approaches.

Dynamic websites
This refers to websites that are created on a request-specific basis and are not readily available on the server for retrieval.

Static Web Pages

Web pages can be static on a server and deliver exactly the same output to the end user’s browser every time they are called up. An example of this is a Word document stored on a hard disk. Changes to the appearance are only made if this is “hardcoded,” i.e., initiated by direct modification of the HTML file by a developer (Pomaska, 2005).

Client-Side Dynamics

An initially static web page can be extended with dynamics by implementing further technologies, such as JavaScript. JavaScript can be used, for example, to accept user input and, as a result, change the appearance of the HTML document or perform arithmetic operations and generate a return value or change the appearance of the web page. JavaScript is not executed on the server, but directly in the browser of the user who has called the respective web page. The corresponding JavaScript code is integrated directly into the HTML web page as a tag (Pomaska, 2005).

Server-Side Dynamics

In addition to client-side dynamics, it is also possible to have the dynamics generated on the server side. A distinction can be made here between script languages (which extend the range of functions of the server) and programs on the server (which communicate with the server via the common gateway interface (CGI)). CGI can be written in different programming or script languages and describes the communication rules between server and application.

Besides CGI, it is also possible to implement a server extension using PHP. PHP is also executed on the server. It is integrated into the HTML files, calculates a desired output, and generates the HTML output, which is returned to the client or browser of the end user (Pomaska, 2005).

Web Application Quality

An important criterion when evaluating web applications is their quality. In principle, every developer and user has different expectations regarding the quality of a web application. ISO 25010 defines standardized criteria that can be used to assess the quality of a web application (International Organization for Standardization, 2011). This standardization defines various aspects as quality criteria for software. Here, we will consider just the main categories.

Table 1: Quality Criteria of a Web Application According to ISO 25010

Quality criterion	Description
Effectiveness	Accuracy and completeness with which users achieve specified goals
Efficiency	Amount of resources expended in relation to the accuracy and completeness with which users achieve goals
Satisfaction	Degree to which user needs are satisfied when a product or system is used in a specified context
Freedom from risk	Degree to which the quality of a product or system mitigates or avoids potential risks to economic status, human life, health, or the environment
Context coverage	Degree to which a product or system can be used with effectiveness, efficiency, satisfaction, and freedom from risk in both specified contexts of use and in contexts beyond those explicitly identified

Source: Sturm (2020), based on International Organization for Standardization (2011).

ISO 25010 further elaborates on the distinction between functional and non-functional quality criteria. Functional quality criteria primarily focus on whether an application fulfills the function that it was originally meant to perform. Aspects such as **response times**, **latency**, and **availability** are considered under non-functional quality criteria. These include a web application's freedom from errors and the runtime (i.e., performance) of a web page. With regard to freedom from errors, reference can be made to the standardization of web applications by the World Wide Web Consortium (W3C) under the chairmanship of Tim Berners-Lee. This standardized many programming concepts that are impor-

Response time

This is the amount of time it takes to receive a response from a server from the moment the request is sent.

Latency

The time span it takes to complete a command is called the latency.

Availability

The accessibility and usability of an application is referred to as its availability.

tant for the internet, such as HTML, CSS, and XML. The W3C also offers validators (W3 Validator, n.d.), which can be used to check the conformity (and, thus, freedom from errors) of a developed web application with regard to W3C standardization. Compatibility with different browsers must also be considered during evaluation.

The usability of an application is another criterion. Usability describes the extent to which an application can be used by its end users for an intended application context in order to achieve desired goals.

Even if the aspects of quality assurance mentioned above do not seem very complicated, and, in some cases, even seem self-evident, their actual implementation represents a considerable challenge. The framework conditions for implementation are set, for example, by the fact that a web application must function with a large number of different operating systems (Microsoft's Windows, Apple's macOS, Linux, and others). This is complicated further by the fact that each of these operation systems uses a different browser (e.g., Safari, Internet Explorer, Chrome, or Firefox) that places specific demands on the respective web application. A web application must typically also be compatible with different base systems, i.e., with stationary desktop computers with different screen resolutions and sizes, with cell phones (Apple's iOS, Android), and possibly with other consumer devices (e.g., the Amazon Echo).

1.4 Current Trends

As technology evolves, the diversity in choice of web development techniques grows. Whenever we start work on a web application, a key decision in its success will be the choice of which design or architecture pattern to use. Today's web applications have different expectations and requirements than before; there is higher demand for increased availability, flexibility, fault tolerance, and reliability. Scalability and ease of deployment have become requirements for organizations wanting to increase their agility in keeping their web applications tightly aligned with their business goals and market opportunities.

In this section, we will consider some of the most effective architecture trends used recently and that are still pertinent. We will highlight the strengths and weaknesses of these architectural patterns, and we show how to determine which pattern is suitable for solving different types of problems. It is also necessary to focus on the latest trends, updates, techniques, and approaches, since knowing trends and monitoring what is happening in the area is essential for web development.

The web industry has experienced architectural styles ranging from monolithic and service-oriented to microservice and serverless. Let us start with the traditional architectures, multi-page application (MPA) and single-page application (SPA).

Multi-Page Application (MPA)

Multi-page applications (MPAs), also known as server-side rendering, are web applications that work in the traditional way (in PHP, Java, Python), i.e., each time the user requests data from the server, it renders a new page which is sent back to the browser. This means that with each page change, the HTML code, scripts, styles, and templates are reloaded, and it is the server which returns all the content (Lau, 2020).

However, most of the time, we only want the new data that come with a new request. We do not need to regenerate the HTML base (e.g., the navigation bar and the footer), yet we always get a new rendering of these base elements. We can use **AJAX** (Asynchronous JavaScript and XML) to render only a particular component, but this makes development more difficult and complex.

AJAX

This is a technique that allows a web page to be updated asynchronously from the server without the necessity of reloading the entire page.

One of the main advantages of MPAs is the fast loading of the initial page. Since the pages and content are rendered by a server-side scripting language, such as PHP, before being sent to the client's browser, the rendered page is quickly received on the client's end.

However, MPAs also have a major drawback: the required server-side round trips tend to make them quite expensive in terms of bandwidth. Fortunately, most of these issues have been alleviated over the years, thanks to various browser features and technology improvements such as AJAX requests. Still, these techniques add more complexity to the development and deployment phases.

Single-Page Application (SPA)

Unlike server-side rendered applications, SPAs are client-side and render content in the browser using JavaScript, Angular, or Bootstrap, without requiring new pages to be reloaded during use (De Sanctis, 2021). In other words, SPA loads once on a browser and does not require the page to be reloaded and re-rendered throughout the life of the application.

Examples of SPAs include Gmail, Google Maps, AirBNB, Netflix, Pinterest, PayPal, Facebook, and GitHub, all of which use SPAs to build fluid, scalable experiences (Lawson, 2021; Themeselection, 2020). The main purpose of this type of web application is the ability to update parts of an interface without sending or receiving a full-page request. The SPA trend means we can avoid long communication time with the server, offering better page performance and high level of data protection.

One drawback of SPAs is the slow initial load time. Most resources, such as CSS styles, JavaScript code, and HTML templates, are loaded only once at the beginning (De Sanctis, 2021). By doing this, the application usually slows down from its initial load time, especially in a large SPA. However, SPA is quick when rendering content after the initial load. Only data are sent back and forth; the basic HTML and layout remain unchanged.

Microservices

Microservice architecture emerged as the result of the shortcomings and drawbacks of the traditional monolithic architecture and service-oriented architecture (SOA), as we will now discuss.

Monolithic architecture

In a monolithic architecture, a software application is designed to function as a single piece of code and can be deployed on a single machine. Monolithic architecture components are interconnected and interdependent, resulting in tightly coupled code. We present the most important advantages and disadvantages of monolithic architecture (W3 Lab, 2021; Valuy, 2020).

If an application is relatively small, there are some advantages to using a monolithic architecture:

- Better performance
- Easier to deploy (due to the simplicity of the high-level architecture)
- Easier to test and debug (in spite of the tightly coupled logic)
- Easier to scale (only need to run multiple instances of the same application)

As the size and complexity of an application grows, there become serious drawbacks to monolithic architecture:

- Difficult to maintain (tightly coupled components mean a change in one part of the application is more likely to affect other parts of the application)
- Difficult to introduce different types of technologies and programming languages (since the application is written as a single unit)
- Difficult to distribute functionality and responsibility among multiple development teams (changes made by one development team may affect another development team)

As such, large and complex software applications should not be monolithic. Microservices and serverless architecture are alternatives to monolithic architectures and address some of their issues and limitations.

Service-oriented architecture

Service-oriented architecture (SOA) is the foundation on which microservices are constructed. With SOA, we try to visualize and design the application as a combination of services that communicate with each other and the end user and meet user requirements. For example, if we have an email application, we may have services (e.g., sending and receiving emails) each running a specialized task.

There are many important advantages to using a service-oriented architecture (Singh et al., 2020), including

- flexibility. The application is split into different services, making it easier for teams to work on different services without affecting each other.
- loose coupling. Services are independent so that updates in one service will not affect other services. As a result, we can design, implement, and deploy a service independently, regardless of any other service. Therefore, each microservice can be developed using a programming language and framework that best suits the problem it is designed to solve (meaning they can be developed with different programming languages and use different frameworks). In addition, we can maintain each service without referencing any other service, and if one service goes down or encounters a problem, the whole application will keep working.
- easy testability. Each service can be tested independently, so you do not have to wait until all the code of the application is finished.

One drawback to service-oriented architecture is that we are still dealing with large services. If there is a problem with the deployment of these services, since the services are not yet small enough, we may be confronted with the deployment of a monolithic application. Thus, we experience similar difficulties to those of monolithic application, such as improvement and maintenance. Another major challenge with SOA is scalability, as scaling a single service is difficult.

Understanding microservices

After SOA, microservices are the next logical architecture, where we extend the SOA and start to consider dividing services into a fine-grained web service.

Most microservice design principles were already available through SOA. So where is the novelty? The prefix “micro” in microservices refers to the granularity of its internal components, meaning that the service components within microservice architecture are usually single-purpose and do one task really well and in an optimal manner. In contrast, SOA services usually include many more tasks and are often implemented as complete subsystems. For example, if we have an authentication service, we can have microservices like filter id, validation id, etc., each handling a separate task.

The next most important aspect is that microservices are loosely coupled (i.e., independent entities), and each microservice should be developed, tested, deployed, and managed as a complete sub-application. While these features are not achievable with SOA and are highly desirable, they are becoming the accomplishment and necessity of a microservices-based architecture.

There are many advantages to using a microservices. The following advantages are a result of the loose coupling (Garrett, 2017; Singh et al., 2020):

- Services are easier to scale because each service can be deployed and scaled independently.
- Services are smaller and independent of each other, and therefore they can be deployed quickly and easily. In addition, we do not need to deploy the entire application each time a microservice is modified.

- Service updates and new features can be deployed more quickly and easily, making continuous deployment possible. (Since each microservice can be deployed independently, developers do not have to coordinate the deployment of local changes specific to their service.)
- Services can be tested separately without affecting each other (Singh et al., 2020).

Despite the many advantages, there are drawbacks to using a microservice architecture. The following are some of the main challenges with microservice-based architecture (Garrett, 2017):

- A distributed application architecture comes with distributed—and new—responsibilities. As a distributed system, a microservice architecture introduces complexity simply not found in a monolithic application. When multiple services work together in a distributed system and something goes wrong, there is added complexity in figuring out what failed and where.
- Decomposing a complex system into the right set of microservices can be difficult. You want a system with microservices that are fine-grained, but this results in a large number of microservices. As the number of services increases, management of those services becomes increasingly complex.
- A dependency on **DevOps** can occur. As we need to maintain multiple services that interact with each other through messages, we need to make certain all services are available and properly monitored.

DevOps

This is a portmanteau of the words “development” and “IT operation.” This set of agile methods tries to achieve a short and continuous software development life cycle.

Serverless Architecture

The term serverless refers to a way of building and running applications and services without needing to manage infrastructure (or administer servers). The application still runs on servers, but all server management is done by cloud service providers. The idea is that a developer should only focus on developing code, and the cloud providers provide the necessary IT infrastructure. Cloud providers typically use on-demand computing models, and scaling functionality is handled dynamically by the cloud services.

With serverless architecture, you do not have to deal with physical servers, and the complexity of how compute resources are delivered is hidden from you. Software applications use as much or as little compute ability as they need.

Serverless architecture is maturing, and its use is increasingly common. It may be the ideal architectural choice for some web applications. A number of cloud providers, including Amazon Web Services, Microsoft Azure, Google cloud platform, and IBM, provide compute services (Dignan, 2021).

Cloud computing has innovated the way we consume and operate IT and application resources and make them available to as many people as possible. It refers to the storing and accessing of data in the cloud (a massive of interconnected networks of servers or even individual PCs) rather than on a computer hard drive (Miller, 2008). It opposes the notion of local storage, which consists of storing data and launching programs from the hard drive.

There are many important advantages to using serverless architecture, which is why it is becoming increasingly popular. These advantages (Ingeno, 2018; Singh et al., 2020) include

- freedom from infrastructure planning and a focus on building core products. Serverless architecture lets us focus on code while the service provider takes care of the infrastructure and management of the servers. This increases productivity and reduces the time to market. A development team that wants to build an application quickly with less concern for the operational side will find serverless architecture attractive.
- cost effectiveness. With serverless architecture, code is only executed when needed. You benefit from utility billing because you are only billed for the actual compute resources used. Since you only pay for actual usage or actual traffic, you do not have to worry about maintaining minimum infrastructure levels. Organizations moving to serverless architecture will reduce hardware costs because there are no servers and network infrastructures to support. In addition, organizations do not have to hire staff to support so much infrastructure.
- scalability and flexibility. Serverless architecture can scale up and down compute capacity as demand changes. You avoid the situations of not having enough servers during peak capacity periods or having too many idle servers during off-peak periods.
- development technologies. With serverless architecture, developers have the flexibility to select the best languages and runtimes based on the required functionality. It also gives teams the ability to easily experiment with different technologies. Cloud providers are expanding their selection so that different languages can be used for different functions under development.

Serverless architecture of course has some drawbacks. The challenges of serverless architecture (Ingeno, 2018; Singh et al., 2020) include

- vendor-based limitations (when the function requires a long time to complete). Some providers may impose limitations when providing functions as a service. The functions are dedicated to simple and fast tasks. The execution of each function must respect a time limit. In the case of Amazon, there is a limit of five minutes of execution per invocation of a function. As a result, the serverless model is not suitable for all use cases, especially for heavy and complex tasks like video processing, or where the application requires constant traffic. This limitation can be overcome by reviewing the application to break up a task into smaller functions.
- vendor lock-in. Moving from one serverless environment can actually be quite complicated. Because each supplier has specific formats and deployment methods, some degree of refactoring will be necessary if you want to change providers. Ideally, your software application will not depend on a particular cloud provider. One way to overcome this drawback is to use a framework that packages your application in such a way that it can be deployed to any cloud provider.
- difficulties with debugging and monitoring. The loss of control associated with the concept of serverless makes it more complex to diagnose and monitor applications, especially in terms of execution performance and resource use. The “pay as you go” system requires you to have a good monitoring of the time and resources consumed since you will be charged for these items. However, this essential aspect is gradually improving with the integration of better monitoring tools into platforms.

- multitenancy issues. Whenever applications for different clients are running on the same server, it is possible that one client affects the other. Examples include security, e.g., a client being able to access another client's data (especially when the application processes sensitive or personal data), or performance, e.g., a client experiences a heavy load that affects the performance of another client. It is therefore necessary to have guarantees on data security and the management of multiple tenancy of servers.
- immaturity. Serverless technologies evolve very quickly. Therefore, the standards and best practices for serverless architectures are not as well established as those for other types of software architecture. However, some organizations and development teams do not hesitate to use innovative technology if they can use it to their advantage.



SUMMARY

The origin of the internet can be traced back to the development of the ARPA network, which was used by the military to connect computers. Scientific efforts led to the further development of the internet and the form in which it is available to private users today. Web pages are formulated, in their most elementary form, as HTML documents (markup language) and can be extended with JavaScript (client-side), PHP, or CGI (server-side) to have functions beyond their static character. With IP, it is possible to exchange data packets via the internet and address them to a recipient computer. The quality of web applications can be assessed by many criteria, such as the script's freedom from errors (conformity with the W3C standard), performance, and browser compatibility.

We also discussed various architectural styles, including monolith, service-oriented, microservices, and serverless. First, we introduced the MPA and SPA approaches, along with the pros and cons of each. Then, we discussed monolith architecture and challenges that make it unsuitable for larger applications. After that, we presented the SOA, which can be considered the basis for microservices. Microservice-based architecture, as well as its characteristics, advantages, and challenges, was then introduced. Finally, we presented the modern trend of serverless architecture and its advantages and drawbacks.

An important question becomes which architecture to use, given the range of choice. However, there is no definite answer as to which is better; it depends on the features and complexity of the application being developed. The choice should be based on the project at hand.

It is particularly important to note that these architectures are complementary and not mutually exclusive. It is highly recommended to start with a monolithic design and separate the service into microservices step by step, based on a vision for future functionality. This makes it pos-

sible to choose which services to split into smaller services (microservices). Finally, some of these microservices can be implemented as serverless functions (Singh et al., 2020).

UNIT 2

TOOLS OF WEB DEVELOPMENT

STUDY GOALS

On completion of this unit, you will be able to ...

- utilize key tools (including Visual Studio Code) to develop a web application.
- use version management tools (or version control systems) to manage different versions of one or more files of a web application.
- install the necessary software and packages using package managers.
- describe upload and deployment and use deployment tools.

2. TOOLS OF WEB DEVELOPMENT

Introduction

Web application development tools allow you to create web applications that users can access on the internet via web browsers. One of the key steps in setting up an application is choosing the technologies and tools you will use. There is a wide variety of tools available for developing web applications, so selecting the best ones can be a challenge. Several languages and frameworks have become available to developers, and considerable effort has been made to create tools that increase developer productivity and support the complete developmental life cycle.

The first and most essential tool we will need is a development tool, which will be used to write and edit our code. Some developers get by with a simple editor, such as Notepad or Vi, but time is often wasted doing everything from the command line. Another option is a graphical user interface (GUI) editor, which comes with features such as syntax highlighting, easy navigation, customizable interfaces, and search and replace options. There are many GUIs, such as Atom, Eclipse, IntelliJ IDEA, Microsoft Visual Studio, NetBeans, Sublime Text, WebStorm, and Visual Studio Code.

Version management tools (also called version control systems) are used in web development to keep the source code relating to the different versions of software. A version management tool is a software that makes it possible to store a set of files while keeping a chronological record of all modifications made to them. This allows access to the various versions of a batch of related files. Usually, each new version of a file is called a revision, and its version number is incremented by one from the previous.

There are also software and services for decentralized version control (or DVCS, for distributed version control system). Git and Mercurial are two examples and are available on most Unix and Windows systems (GeeksforGeeks, 2020). There are many VCSs, such as AWS Code Commit, Bazaar, Beanstalk, Bitbucket, GitHub, GitLab, and Microsoft Team Foundation Server.

A package manager is another indispensable tool for web application development. A package manager (or package-management system) is a collection of software tools that automates the installation, upgrading, configuration, and uninstallation of computer programs for a computer's operating system in a consistent manner (Burrows, n.d.). Examples of package management systems include Chocolatey, Homebrew, NPM, pkgadd, apt-get, Windows installer, and Yarn.

Once the development of the web application is complete, the next step is uploading and deploying it to its production environments. There is currently an impressive number of high-quality software deployment tools available for this, such as Jenkins and GitLab CI/CD.

2.1 Development Tools

Once we begin developing our web application (using markup languages such as HTML, CSS, and JavaScript), we will need a development tool to edit, review, and manage our code. A development tool is indispensable when developing **websites**. These tools can be categorized into code editors and integrated development environments (IDEs).

A code editor is a fundamental tool for developers, since it is designed with the writing and editing of website source code in mind. Code editors are similar to text editors but have more features and extensions. For reference, examples of text editors include Emacs on macOS, WordPad on Windows, and Vim on Linux. The additional features of code editors are designed with the aim of making editing code easier for developers. These include features such as smart code completion (IntelliSense), automatic highlighting, and extended support for additional languages, frameworks, and libraries. Before using a code editor, however, we will need to install proper plugins and packages for each specific language. There are many code editors on the market, such as Sublime Text, Atom, Visual Studio Code, Komodo, and PHP Storm.

An integrated development environment (IDE) is a set of integrated tools used together in order to increase developer productivity. These include text editors, compilers, and debugging tools. Some IDEs are dedicated to a particular programming language: Visual Studio for ASP.NET applications, RubyMine for Ruby, Eclipse for Java development, Code::Blocks for C/C++, and XCode for Apple technologies.

As code editors are used most by developers, we will now look at Visual Studio Code (Microsoft, 2021).

Visual Studio Code

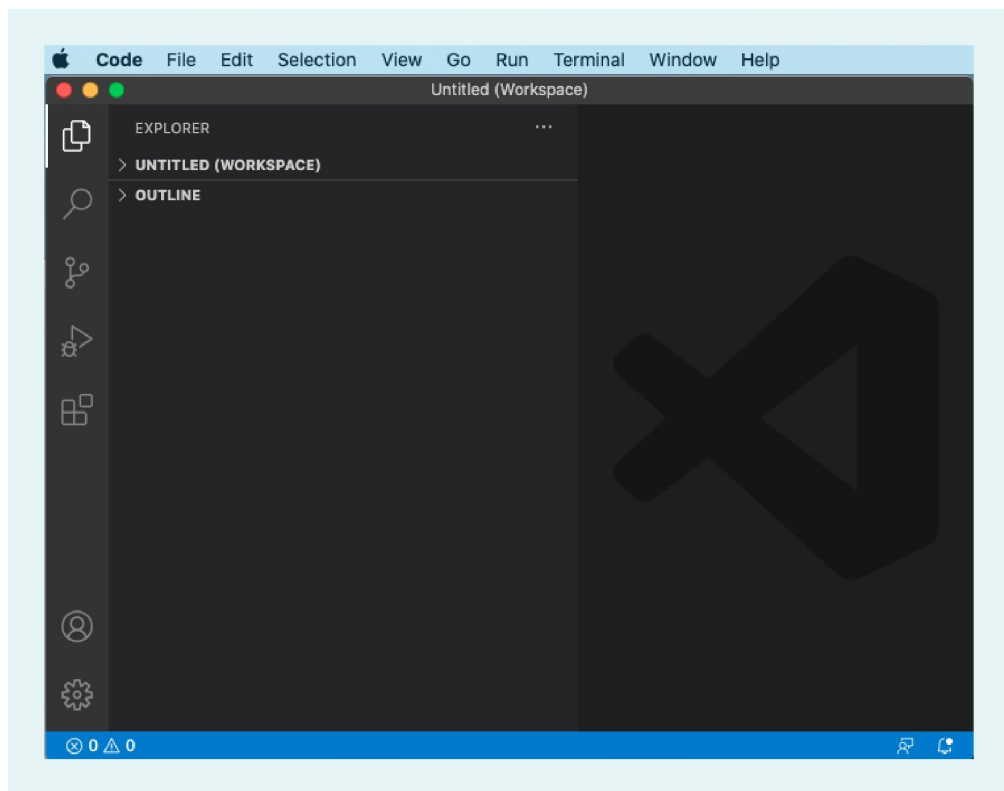
Visual Studio (VS) Code is a multi-platform code editor that runs on Windows, macOS, and Linux. It is free and can be downloaded from the Visual Studio website. VS Code is one of the most popular coding tools used today (Eduonix, 2019). It focuses on speed and extendibility and can build up to the needs of a wider developer community.

VS Code is a cross-platform editor and supports Windows, macOS, and Linux OS. In this sequel, we will explore VS code on the macOS operating system. Once you have installed VS Code and launched it, you will see the default screen below. On the left-hand side, you have an activity bar. We will now start exploring some basic features of VS Code.

Website vs. web page

The term “website” refers to the entire web presence (e.g., that of a company or a project), while the term “web page” describes a single page of a web presence (e.g., the entry or contact page).

Figure 4: Visual Studio Code User Interface

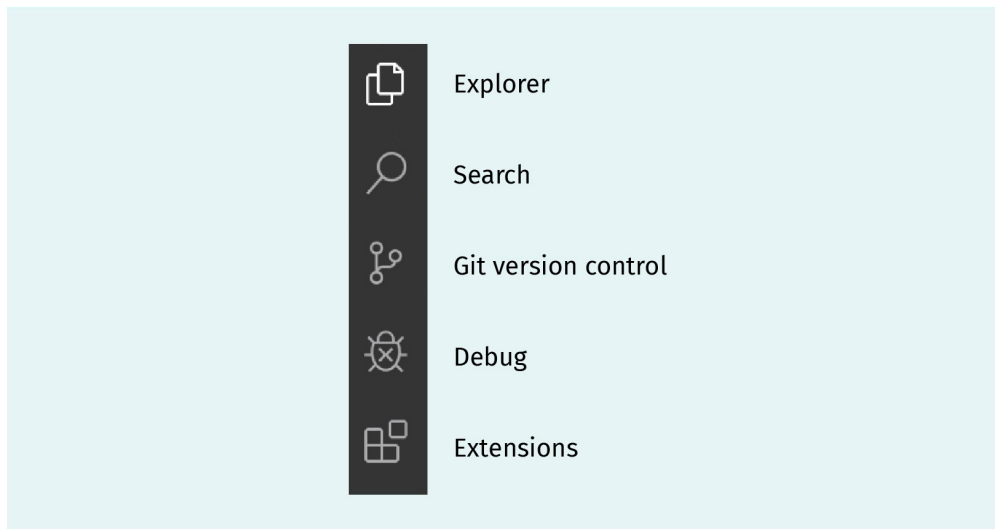


Source: Hadid (2021).

Activity bar

The toolbar on the left is the activity bar. The activity bar provides quick access to the key features of VS code.

Figure 5: Activity Bar Features



Source: Hadid (2021).

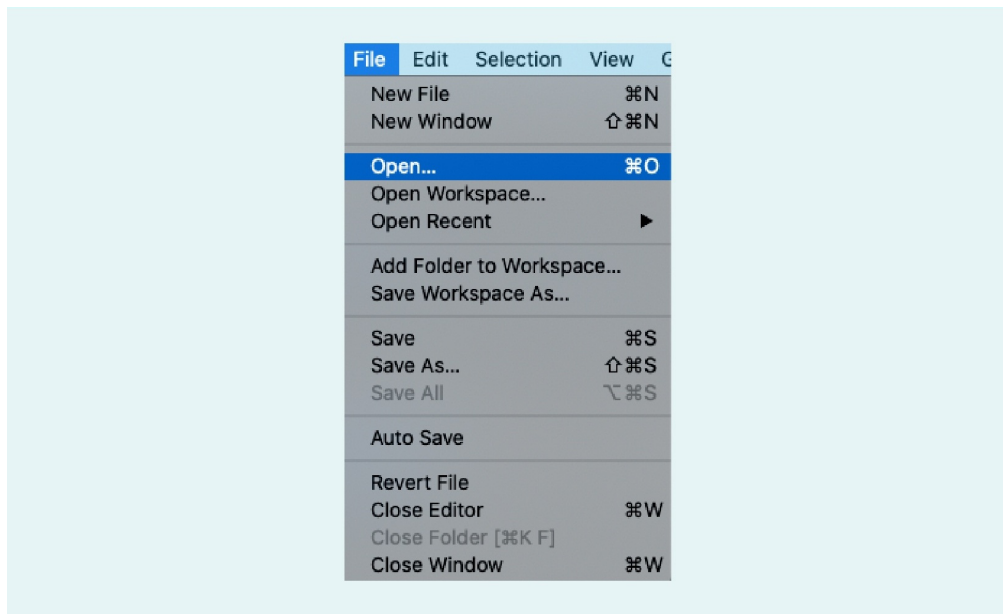
The first button is the explorer, where you can see the folders and files that have been loaded into VS Code. The search button allows you to search for files and folders. The next button is for **Git** version control. To run and debug code, we use the debug button, which allows us to execute a program. To manage your extensions in VS Code, you can use the extension button.

Files and folders

Unlike with an IDE, VS Code does not depend on creating a solution or a project file. The project is the folder, with subfolders and files in it. You can use the “File” menu and “Open” option to either open or create a folder, or to open a file and begin editing it.

Git
Git is a decentralized version control software for tracking changes in a given set of files (Chacon, 2014).

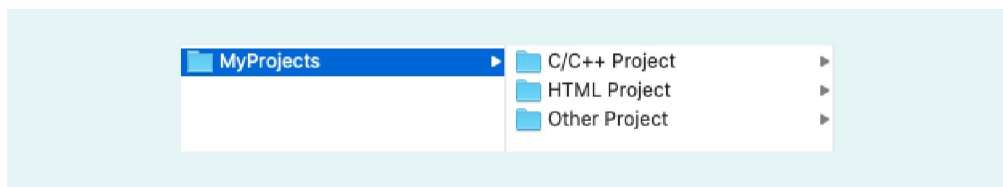
Figure 6: File Menu in VS Code Editor



Source: Hadid (2021).

To practice, we will create the folders seen in the figure.

Figure 7: MyProjects Folders and Sub-Folders

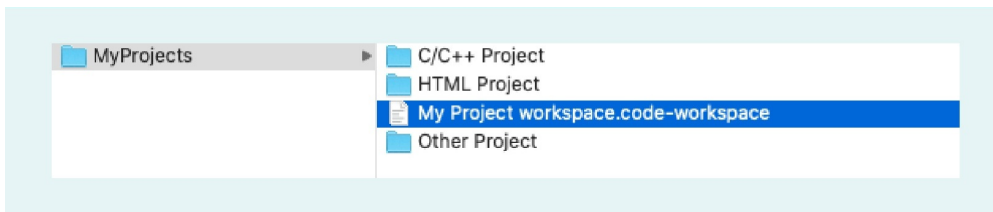


Source: Hadid (2021).

Workspace

A VS Code workspace can contain one or more files and folders. To create a new workspace, select the “Add Folder to Workspace” option and then “Save Workspace As...”. This routine generates a file with the extension “.code-workspace” that contains all the folder names included in this workspace.

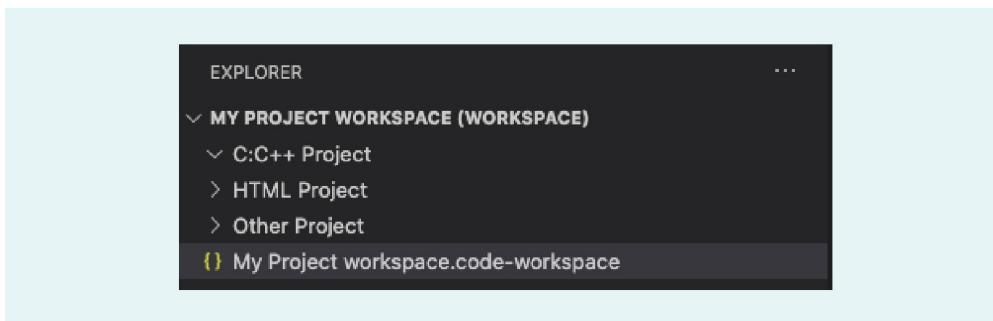
Figure 8: MyProjects Folders and the Workspace File



Source: Hadid (2021).

Opening the workspace in the editor will also open the linked project folders. The following figure shows how the workspace and folders are linked in VS Code.

Figure 9: MyProject Folders as Part of the Workspace in the VS Code Editor



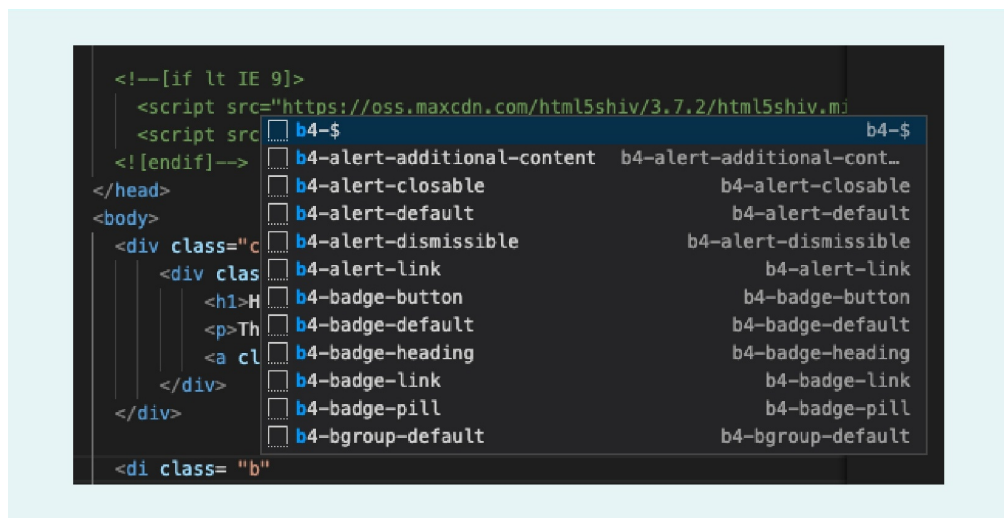
Source: Hadid (2021).

The workspace file “My Project.code-workspace” stores the path of folders linked in the workspace.

IntelliSense

IntelliSense is a useful feature for developers and includes smart code completion, content assist, and code hinting. These come in handy when the development environment prompts a list of keywords matching with what you are typing, as shown in the following screenshot.

Figure 10: An Example of Code Completion



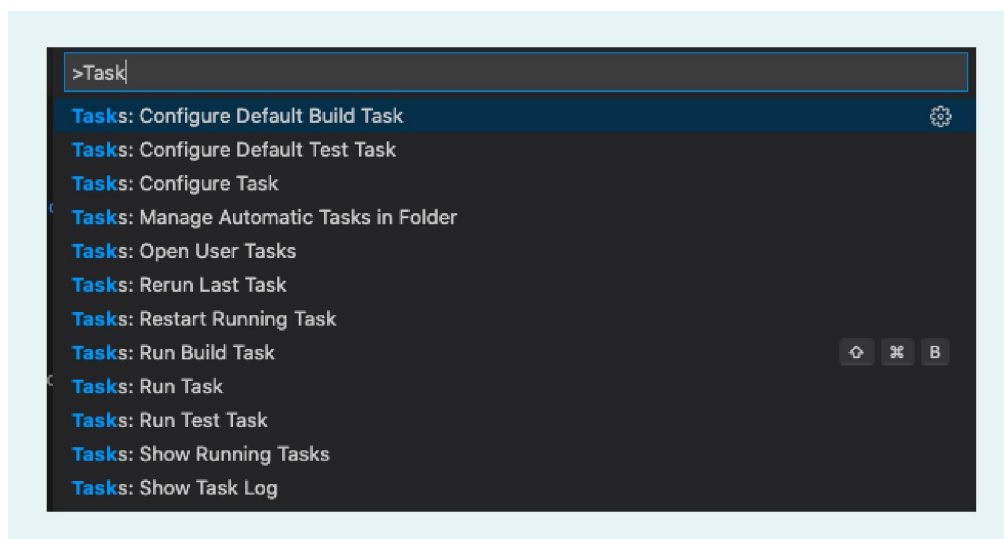
Source: Hadid (2021).

VS Code provides IntelliSense for JavaScript, TypeScript, JavaScript Object Notation (JSON), HTML, CSS, and Bootstrap. It is also possible to add additional language extensions from the Visual Studio marketplace (Eduonix, 2019).

Tasks

IDEs provide automated and well-integrated processes that increase the productivity of a developer. VS Code provides similar functionality through its Tasks framework, illustrated here.

Figure 11: Tasks Option



Source: Hadid (2021).

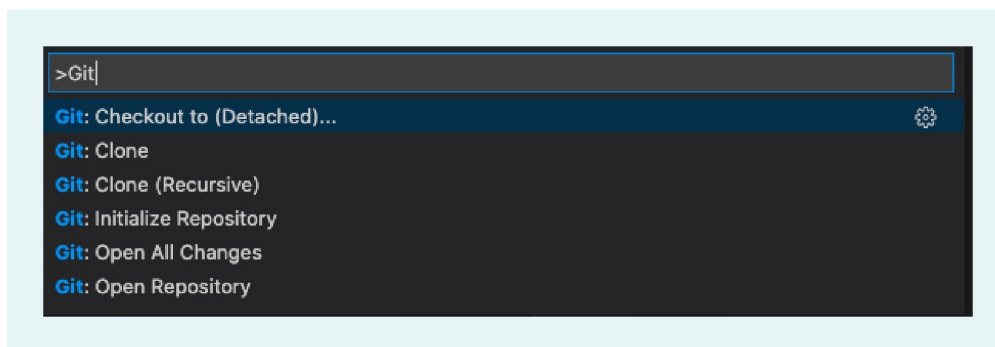
Debugging

Another important feature of VS Code is the ability to debug code. Using VS Code, you can debug Node.js, JavaScript, and TypeScript.

Version control

VS Code integrates Git for version control purposes. You can quickly initialize your Git repository by calling it via command palette and selecting “Git: Initialize Repository.”

Figure 12: Using Git for Version Control Through the Command Palette

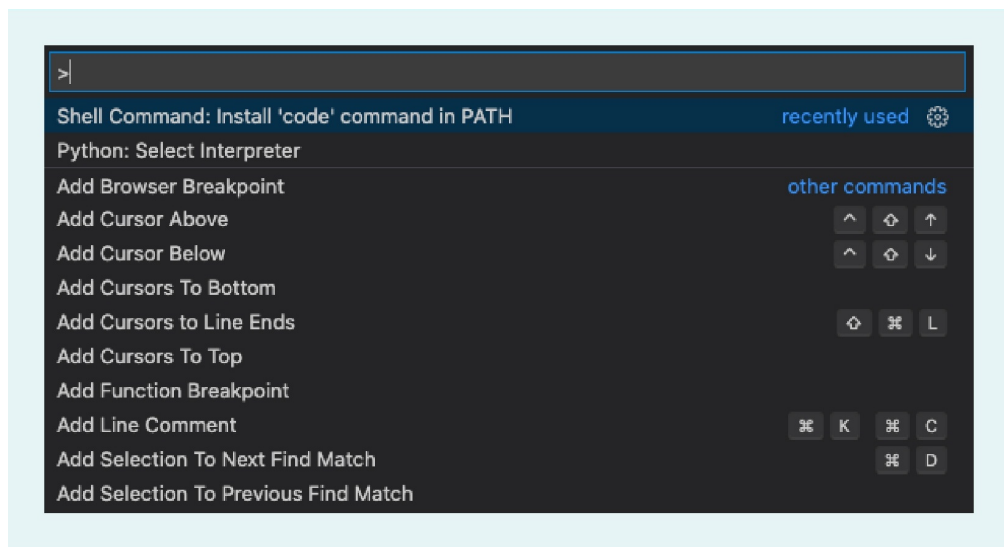


Source: Hadid (2021).

Command Palette

The command palette is the centerpiece of the VS Code editor. You can call the command palette by pressing Ctrl + Shift + P on Windows or Cmd + Shift + P on macOS. A window will appear at the top of the editor, initialized with a > sign.

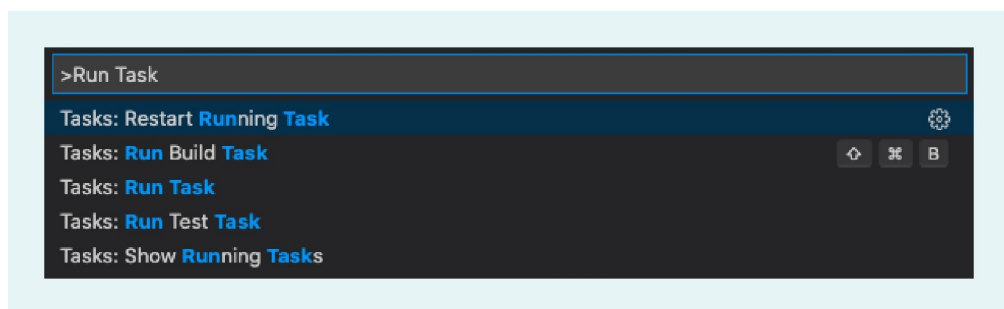
Figure 13: Call a VS Code Function Using the Command Palette



Source: Hadid (2021).

From there, you can run any command (for example, running a task).

Figure 14: Quick Way to Call a VS Code Function Using the Command Palette

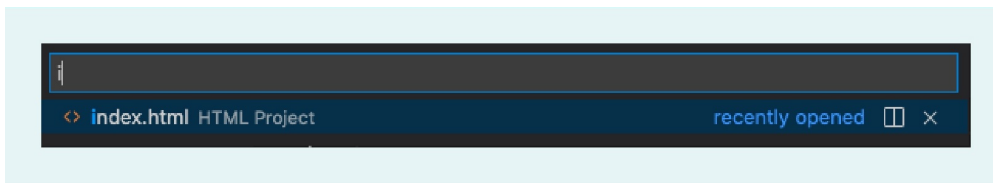


Source: Hadid (2021).

Search option

After you have opened the command palette, remove the > sign. You will now be able to search for files in the project.

Figure 15: Quick Search for Files Using the Command Palette

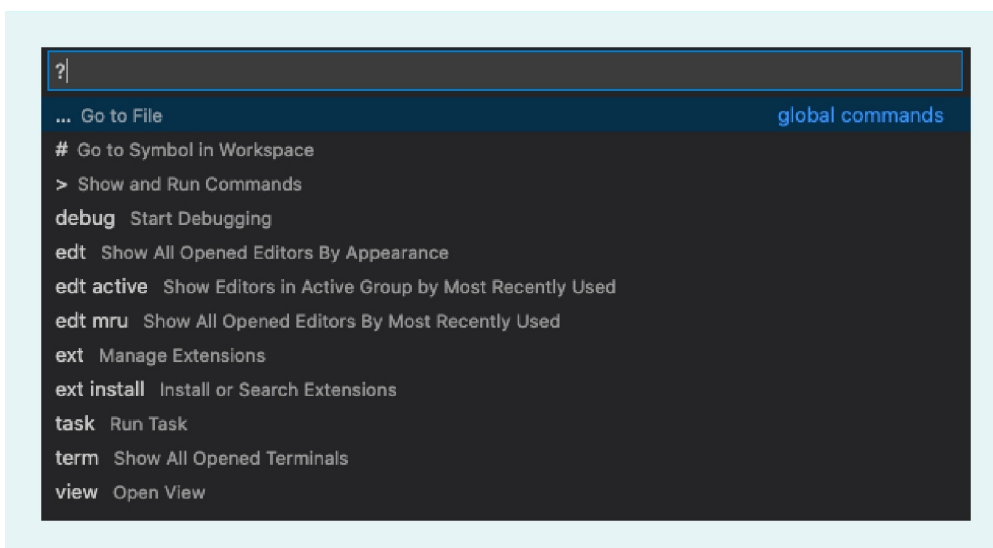


Source: Hadid (2021).

Help (?)

After opening the command palette, type the question mark sign. You will see a list of commands you can run.

Figure 16: Listing Possible Commands Using the Command Palette



Source: Hadid (2021).

debug

Use the debug command to launch the debugger or add a debug configuration.

Figure 17: Debugging Options in the Command Palette

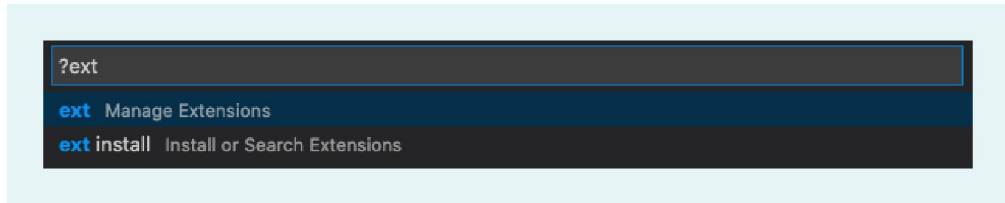


Source: Hadid (2021).

ext

Use the ext command to manage and install extensions.

Figure 18: Using the Command Palette to Call Extensions



Source: Hadid (2021).

Integrated Terminal

VS Code has an integrated terminal window inside the editor. This is a useful feature, as it means you do not need to open a terminal window separately to run command-line instructions. From the command palette, use the “View: Toggle Terminal” command. You will see a window appear at the bottom of the editor.

Figure 19: Option to Select the Default Shell from the Integrated Terminal



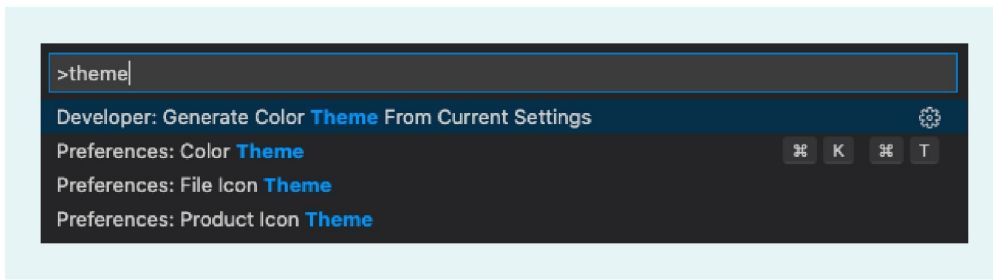
Source: Hadid (2021).

Themes

Another important customization feature of VS code is the ability to set themes. Some developers like a dark colored environment, while others prefer light colors. VS Code offers a list of themes to choose from.

You can set your preferred theme by calling the command palette, typing theme, and choosing “Preferences: Color Theme.”

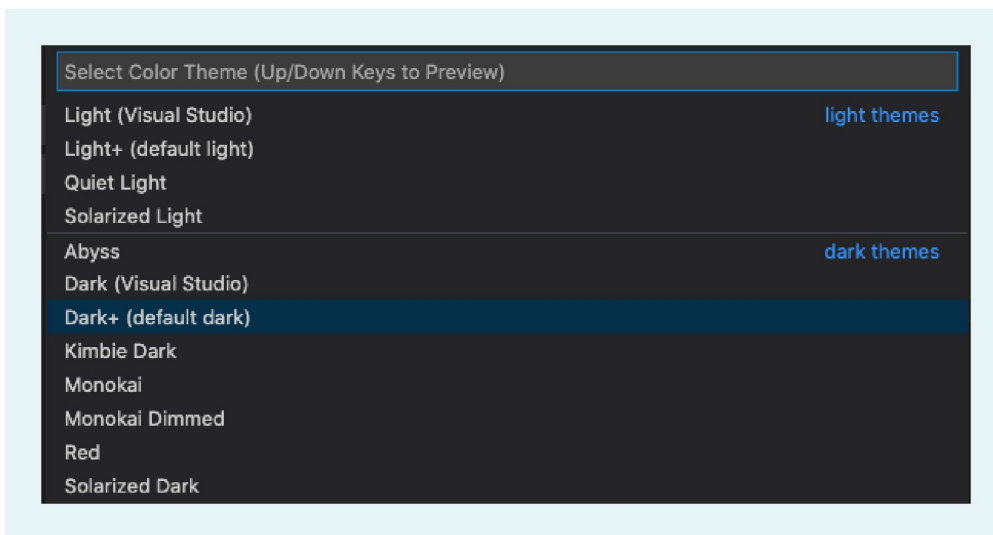
Figure 20: Calling Theme Settings from the Command Palette



Source: Hadid (2021).

The following figure shows a sample list of the themes available.

Figure 21: List of VS Code Themes



Source: Hadid (2021).

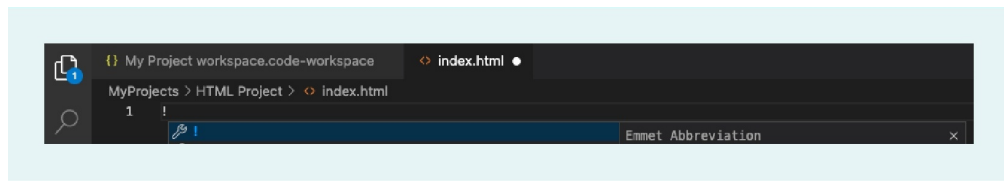
Basic Editing in VS Code

VS Code offers various features that allow developers to code faster. One of these is Emmet.

Emmet

The Emmet extension is built into the VS Code editor and helps to quickly write HTML code. Let us create an HTML file in Visual Studio Code, type “!” or “doc,” and press enter.

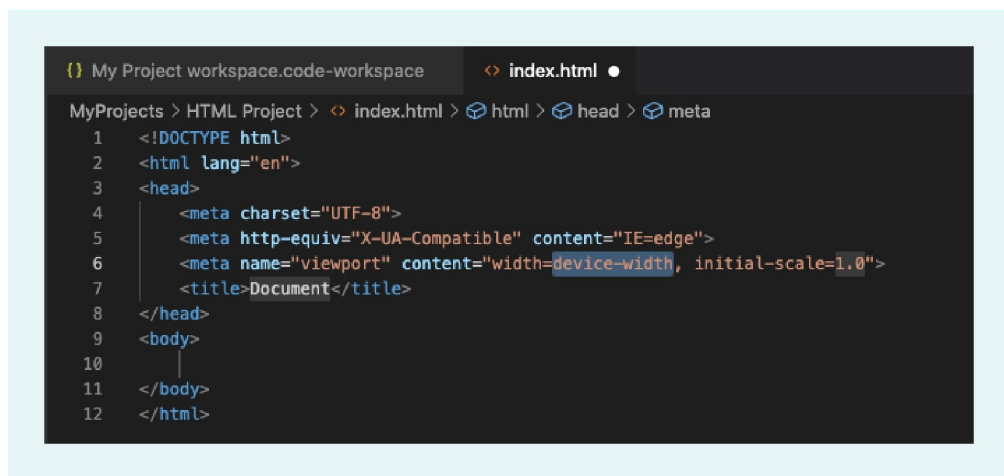
Figure 22: Using the Emmet Extension to Create HTML Code



Source: Hadid (2021).

VS Code will then automatically create a basic HTML template.

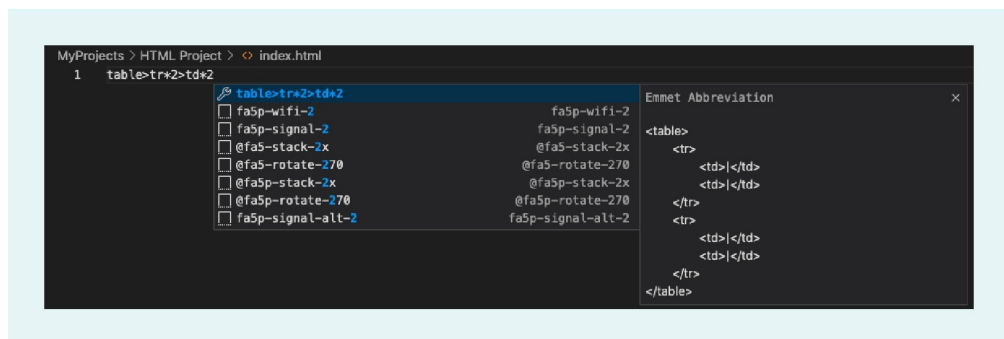
Figure 23: HTML Template Generated by the Emmet Extension



Source: Hadid (2021).

In Emmet, you can generate an element by typing its name and pressing the Tab key. For example, if you type “div” and press Tab, Emmet will expand that into “<div> </div>”. To create a table with five rows and two columns in each row, you can use the following statement: `table>tr*2>td*2`. This will produce the following table.

Figure 24: HTML Code Generated by the Emmet Extension



Source: Hadid (2021).

As shown, you will notice that VS Code shows you a preview of the HTML it will generate. Pressing Enter generates the HTML code.

The > operator can be used to create a child element. With the * operator, you can define how many times a particular element should be output.

A class attribute is added to a tag by using a dot (.) after the tag name, as in the following code snippet: `table>tr*5.myclass>td*2`.

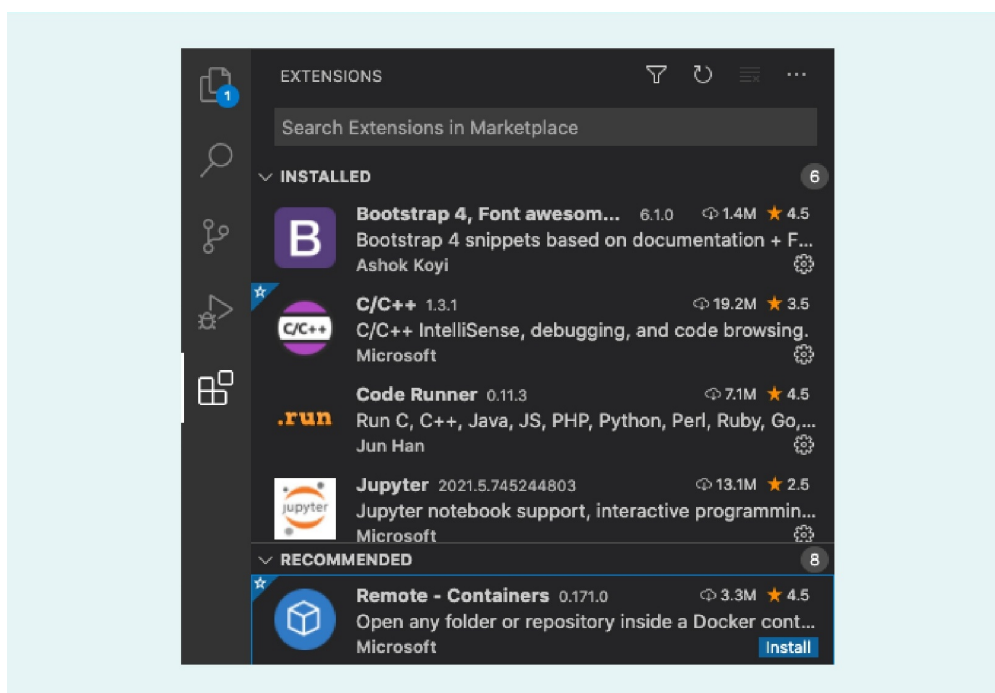
Extensions in Visual Studio Code

One of the key features of VS Code is extensions. With extensions, a developer of any platform can use VS Code with any language and framework. Extensions can be installed from VS Code's extension gallery in your editor and used to perform the desired operation.

Examples of extensions include support for Emmet, an integrated terminal, IntelliSense support for various markup languages (such as JavaScript, TypeScript, JSON, HTML, and CSS), and version control support.

VS Code makes it easy to browse and install extensions. You can install extensions through the extensions pane (which can be called with Ctrl. + Shift + X on Windows or Cmd. + Shift + X on macOS).

Figure 25: Extensions Pane



Source: Hadid (2021).

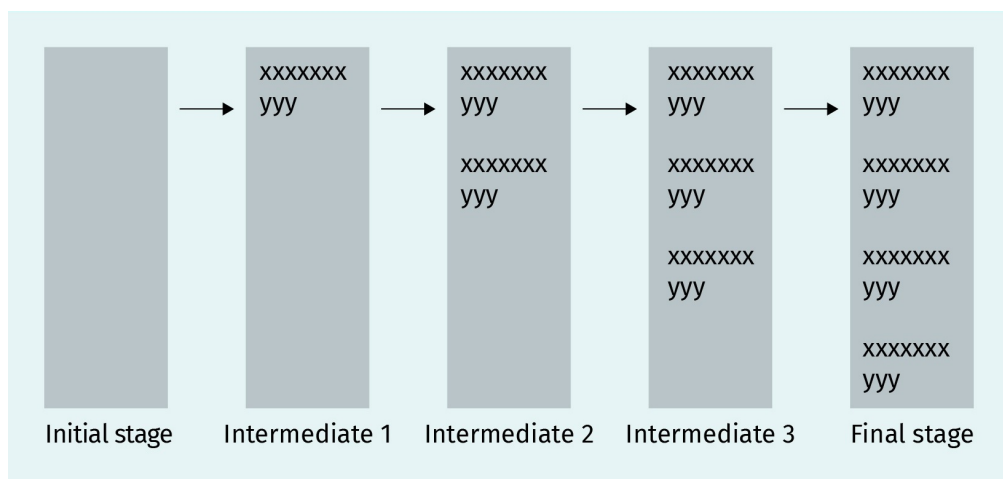
This pane sorts extensions by “Installed,” “Recommended,” and “Disabled.” Currently installed extensions are displayed under the Installed tab, recommended extensions will be listed under the Recommended tab, and the Disabled tab shows the extensions currently disabled.

2.2 Version Management

The version management tool (or version control system (VCS)) is used to manage multiple versions of one or more files. It is used primarily in the field of web and software development and mainly involves managing the source code.

The version control system is the software responsible for the long-term management of changes made to a file (i.e., the source code of a program, document, or website) in such a way that we can go “back in time” to invoke a specific version of that file. The following two figures illustrate the evolution of content creation with and without a version control system.

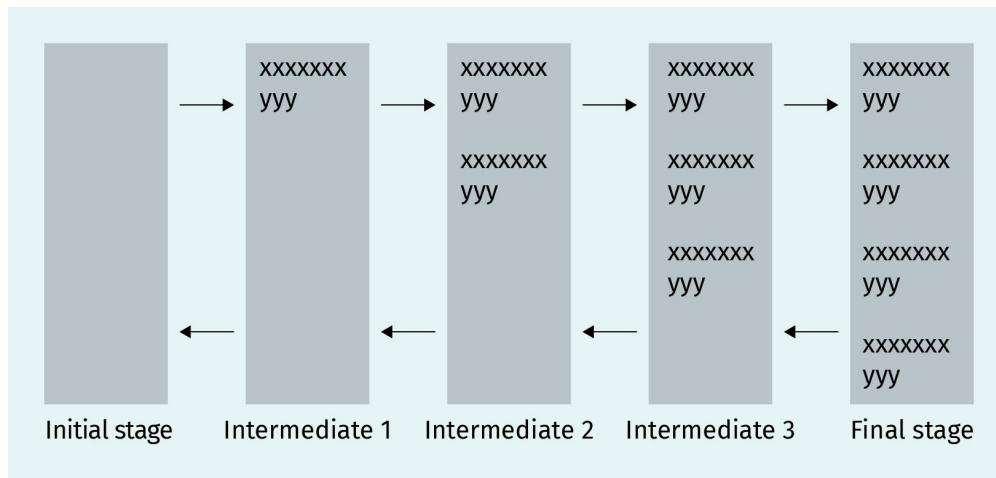
Figure 26: Flow of Content Creation Without a VCS



Source: Hadid (2021), based on McGlone (2020).

In the previous figure, we see that a normal evolution of a text file flows from left to right, meaning that content creation progresses in one direction (when it comes to construct content in different time periods). In this flow, you cannot return to an intermediate stage to make edits without losing data. (You also cannot use the undo feature, as you cannot undo something once you have saved and closed a file).

Figure 27: Flow of Content Creation with a VCS



Source: Hadid (2021), based on McGlone (2020).

However, when you use a version control system, the flow is in both directions. Every modification to the content is considered a new version, meaning that we can return to a previous version without any data loss.

Types of Version Control Systems

There are two types of version control systems:

1. Centralized version control systems (CVCSs), e.g., concurrent version system (CVS) and Apache Subversion (SVN)
2. Distributed version control systems (DVCSs), e.g., Bazaar, Mercurial, and Git

The key difference is that CVCSs keep the version history on a central server, from which every developer can request the latest version of the work and then apply the changes to it. This means the work of everyone on the server is shared. A drawback of CVCSs is that problems can arise if the central server becomes unresponsive.

On a DVCS, everyone has a local copy of the entire project history. This means it is not necessary to be online to change or make revisions to a project. As there is not a central entity in charge of the project history, team members can synchronize with each other at will. This helps avoid problems should the central versioning server become unresponsive. Git is the most popular DVCS, growing quickly from a niche tool to mainstream.

Introduction to Git

Linus Torvalds, the developer of the Linux Kernel, initiated the Git project in 2005 (Chacon, 2014). Git manages files in repositories. Repositories track changes made to files over a period of time and store the history of what was changed, when it was changed, and who made the changes.

Initializing the Git repository

To start using Git, the first step is initializing the Git repository. A repository is centralized and organized storage for your entire project and can consist of one or more folders, sub-folders, and files.

Running the command `git init` in your project folder will create a `.git` folder in the project directory. This will store the complete history of your project and is referred to as the local repository.

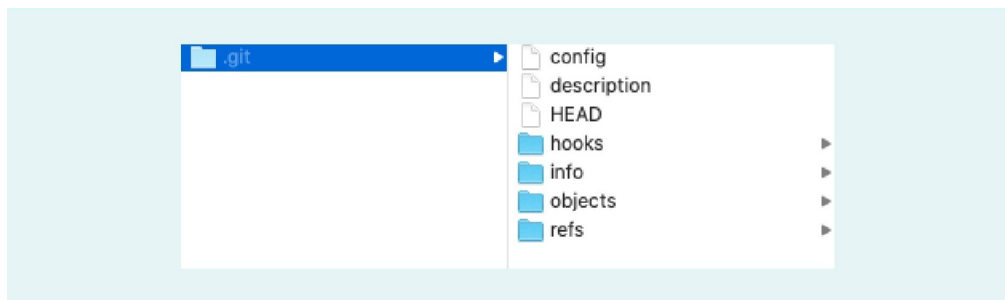
Let us then begin by creating our first repository. `git init` is a single-use command used during the initial setup of a new repository. We will run this command in our folder (“MyFirstRepo”), as shown.

Code

```
MyFirstRepo % git init
Initialized empty Git repository in /Users/IU/Documents/MyFirstRepo/.git/
MyFirstRepo %
```

Git has now created a subfolder with the extension “.git.” This subfolder (which is normally hidden) contains some other files and folders.

Figure 28: Files and Folders Created after Initializing a Repository



Source: Hadid (2021).

To allow you to make changes, Git needs to know your identity. This is because, in Git, every modification made in a repository must be registered to the name and email of an author. Use the following commands to set the name and email of an author:

- `git config --global user.name "Your Name"`
- `git config --global user.email your_email@domain.com`

We will now explain the life cycle of a file inside Git. A file will go through many states, with the Git command line moving this file from one state to another. We will look at each state and its command line now.

Staging files

Initializing a repository in your project folder will not automatically tell Git to track changes. Only files added to the staging area are tracked, meaning they are in the Git repository. To test this, let us create a dummy text file in the project directory (for example, "file1.txt"). Now, in the terminal window, run the command `git status`. As shown here, Git has located the newly added file but it is currently untracked.

Code

```
MyFirstRepo % git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  file1.txt
nothing added to commit but untracked files present (use "git add" to track)
```

To track a file, you must execute the following command: `git add File1.txt`. Running `git add` will stage all files in the folder. We can now check if this was successful. Using the `git status` command again, we can see the status of the repository.

Code

```
MyFirstRepo % git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new files: file1.txt
```

Your file is now tracked by Git. The file is currently staged and ready to be committed inside the repository.

Committing a file

Once files are modified and in a satisfactory state, you can commit them in order to update their content. To do this, use the `git commit` command. When making a commit, it is essential to write a message accompanying the commit using the `-message` (or `-m`) subcommand. This message should describe the nature of the changes you are committing.

Code

```
MyFirstRepo % sudo git commit --messag "First commit"
Password:
[master (root-commit) 9eddb13] First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt
```


After executing the first commit, the repository has a master branch. This branch is also called the root-commit and can be seen in the previous figure. A branch is like a path in a repository, with a repository having multiple paths that often intersect.

Executing the `git log` command will show the list of commits. Since we have only done our first commit, only one will be displayed.

Code

```
MyFirstRepo % git log
commit 9eddb13ea8c84d909655ffa797b515d3317c0107 (HEAD -> master)
Author: John <john.doe@web.com>
Date: Sun Apr 25 10:01:55 2021 +0200
First commit
```

The code above shows the unique commit hash, the author who committed the change, and the timestamp of when these changes were committed to the local repository. The file is in an unmodified state because you just committed the change.

Modifying a committed file

In our demo repository, we modify the file “file1.txt” and add some content to it. For example, we can add the comment “First modification” using the echo command, as shown.

Code

```
MyFirstRepo % echo "First modification" >> file1.txt
MyFirstRepo % git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file1.txt
no changes added to commit (use "git add" and/or "git commit -a")
```

Calling `git status` shows Git is aware of the changes made to the file. The `git status` command is used to view changes to the working directory. They are displayed in red with a “modified” prefix, informing us that the file has been modified. `git status` also informs us that we need to commit the file to save this change. The last message displayed by `git status` is “no changes added to commit.” This lets us know which files we should include in the next commit. For example, if we have modified more than one file, but only want to commit one of them, we can add only that file. However, if we try to commit but skip the step, nothing will happen.

Code

```
MyFirstRepo % sudo git commit
Password:
On branch master
```

```
Changes not staged for commit:
  modified: file1.txt
no changes added to commit
```

Let us add the file again for the purpose of preparing our next commit.

Code

```
MyFirstRepo % sudo git add file1.txt
MyFirstRepo % sudo git status
On branch master
Changes to be committed:
  (use „git restore --staged <file>...” to unstage)
   modified:   file1.txt
```

Now we can make another commit.

Code

```
MyFirstRepo % sudo git commit -m "Second commit"
[master 887b914] Second commit
 1 file changed, 1 insertion(+)
```

After executing the `git log` command, we see the following:

Code

```
MyFirstRepo % git log
commit 887b914bf147e2a54f080f9c1151ef6bcd5fb390 (HEAD -> master)
Author: John <john.doe@web.com>
Date: Sun Apr 25 19:19:26 2021 +0200
    Second commit
commit 9eddb13ea8c84d909655ffa797b515d3317c0107
Author: John <john.doe@web.com>
Date: Sun Apr 25 10:01:55 2021 +0200
First commit
```

After multiple commits are made, we will probably want to review the thread. `git log` lists the commits performed in reverse chronological order, meaning the most recent commits appear first. In our demo, we have made two commits.

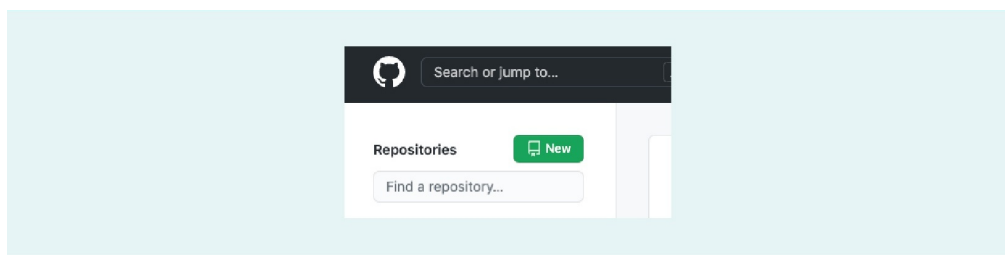
We will now look at how a remote repository is created in GitHub and how a local repository can be pushed to it. GitHub is a Git-based platform that allows developers to store and share, publicly or not, the code they create.

Creating a New Repository in GitHub

The aim of this section is to create a repository in GitHub. To get started, you will need to have the Git command line tool installed on your computer. You will then need to have an account on the GitHub website and log in.

To begin, we create a local repository using the same process explained in the previous section (the “MyFirstRepo” repository). Then, we go to the GitHub website and create a new repository by clicking the “New” button (this is found in the left-hand pane, as seen below).

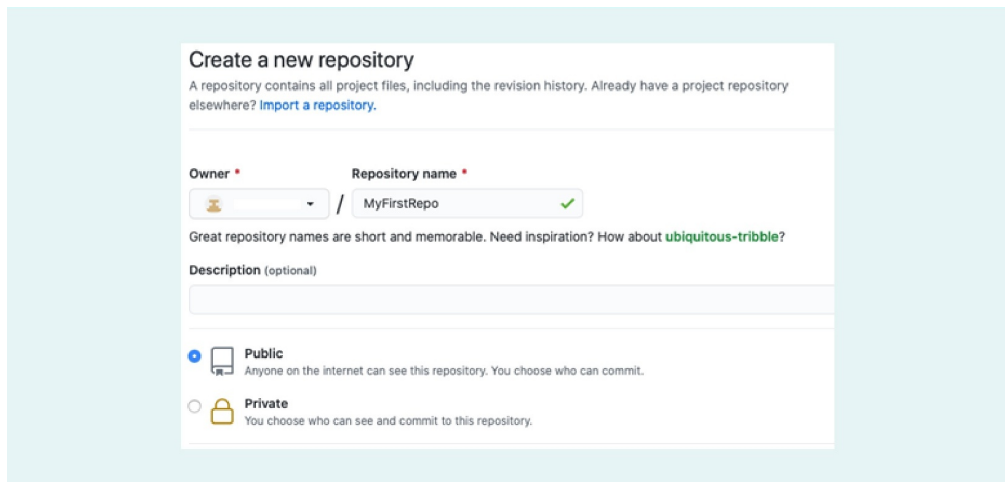
Figure 29: Creating a New Repository in GitHub



Source: Hadid (2021).

The repository must be given a name, so we will use “MyFirstRepo” again, as shown.

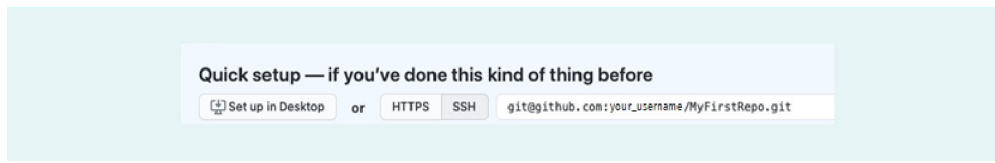
Figure 30: Defining the Name of New Repository in GitHub



Source: Hadid (2021).

We then click “Create” to create our repository. We can click on the “SSH” button to obtain the SSH URL for our repository. This button is found below the text that reads “Quick setup – if you have done this kind of thing before.”

Figure 31: Location of the SSH URL of the New Repository



Source: Hadid (2021).

The SSH URL of our repository should look something like the following: `git@github.com:yourname/MyFirstRepo.git`. We will now set the remote URL on the local repository using the following code:

Code

```
MyFirstRepo % git remote add origin git@github.com:yourname/MyFirstRepo.git
MyFirstRepo %
```

We create a README file (with some content) using the following code: `echo 'IU Web Application Development project' > README.md`

Create a `.gitignore` file using the following command: `echo '.DS_Store' > .gitignore`

Add the README file and the `.gitignore` file to the index using the following command: `git add README.md .gitignore`. Then, commit the README and `.gitignore` files as shown.

Code

```
MyFirstRepo % git commit -m "initial commit"
[master (root-commit) 9295da9] initial commit
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
```

Push the files to the remote repository (GitHub) by using the `git push -u origin master` command, as shown.

Code

```
MyFirstRepo % git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 305 bytes | 305.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:hadidrachid/MyFirstRepo.git
```

```
* [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
MyFirstRepo %
```

A repository has been created on your GitHub account with README.md and .gitignore files.

Navigating Git

Let us now take a closer look at the objects Git stores in the repository.

Commits

When you run the `git log` command in this project's directory, you should get a result that looks like the following:

Code

```
MyFirstRepo % git log
commit 9295da9184eb4989f129e334e6f29c3be060b147 (HEAD -> master, origin/master)
Author: John <john.doe@web.com>
Date: Mon Apr 26 16:11:43 2021 +0200
    initial commit
```

Notice that the `git log` command displays the commit we did in this repository; we have only one commit for now.

The hash

The first line contains the code (or hash) of the commit, a sequence of 40 hexadecimal numbers. Each commit has a unique code within the repository, and we refer to it when we want to do certain actions.

The commit messages

The message we attached to the commit we made is displayed under the author and date. By using the `git log` command with the `-format=fuller` option, we can display the committer and the committing date.

Code

```
MyFirstRepo % git log --format=fuller
commit 9295da9184eb4989f129e334e6f29c3be060b147 (HEAD -> master, origin/master)
Author:   John <john.doe@web.com>
AuthorDate: Mon Apr 26 16:11:43 2021 +0200
Commit:   John <john.doe@web.com>
CommitDate: Mon Apr 26 16:11:43 2021 +0200
    initial commit
```

Deep analysis

We can go deeper into our analysis to display more information using the `git log` command with the `--format=raw` option.

Code

```
MyFirstRepo % git log --format=raw
commit 9295da9184eb4989f129e334e6f29c3be060b147
tree e57f977bd6c515fc1d67d36ccb5958ac10e60674
author John <john.doe@web.com> 1619446303 +0200
committer John <john.doe@web.com> 1619446303 +0200
    initial commit
```

As before, we have the author and committer displayed, but in a more compact form, and there is the commit message. But we have a new message: tree.

Trees

The tree more commonly represents a directory or folder of your application, which also contains files and other subfolders.

By using the `git cat-file -p` command, we can display the contents of the tree.

Code

```
.git % git cat-file -p e57f977b
d6c515fc1d67d36ccb5958ac10e60674
100644 blob e43b0f988953ae3a84b00331d0ccf5f7d51cb3cf
    .gitignore
100644 blob 28db6070ba3f29124f7e2df09423c21f30f6ee17
    README.md
```

This tree contains some additional objects (files), called blobs.

Blobs

At the right of the previous command output, we have `README.md` and `.gitignore`, which makes us guess that Git blobs represent the files. As before, we can verify its contents by looking inside.

Code

```
.git % git cat-file -p
28db6070ba3f29124f7e2df09423c21f30f6ee17
IU web Application Development project
```

Its content is exactly the content of our `README.md` file. To confirm, we can use the `cat` command, which allows us to see the contents of a file.

Code

```
MyFirstRepo % cat README.md
IU web Application Development project
```

As you can see, the result is the same. Blobs are binary files.

In the sequel, we need a new commit. So, let us now proceed by modifying the README.md file.

Code

```
MyFirstRepo % echo "IU Networking project" >> README.md
MyFirstRepo % git add README.md
MyFirstRepo % git commit -m "Second commit"
[master 41475de] Second commit
 1 file changed, 1 insertion(+)
```

Use the `git log` command to check the new commit; the `--oneline` line option allows us to see the log in a more compact way.

Code

```
MyFirstRepo % git log --oneline
41475de (HEAD -> master) Second commit
9295da9 (origin/master) initial commit
```

We have a second commit, with its hash. Let us see what it is inside.

Code

```
MyFirstRepo % git cat-file -p 41475de
tree 5af1c427727d260a944abb20f8429501d848639f
parent 9295da9184eb4989f129e334e6f29c3be060b147
author John <john.doe@web.com> 1619523800 +0200
committer John <john.doe@web.com> 1619523800 +0200
Second commit
```

Observe that there is something new: the parent `9295da9184eb4989f129e334e6f29c3be060b147` row.

A parent of a commit is simply the commit that precedes it. In fact, the `9295da9` hash is the hash of the first commit we made. So, every commit has a parent (except the first commit), and following these relations between commits, we can always navigate from a random commit down to the first commit (i.e., the root commit).

2.3 Package Manager

A package manager is a system or set of software tools that automates the installation of new software and tracks what software is installed on your system. The package manager allows you to easily upgrade, configure, or remove software previously installed (Chacon, 2014). Most package managers are designed for discovering and installing developer tools. Think about the Windows app store, the Mac app stores, or the various Android app stores.

Packages are file archives that hold all the files that make up software (such as the application itself, shared libraries, development packages that contain the files needed to build the application). Examples include Visual Studio or Git.

An important aspect of packages is the relationships they contain. Effectively, packages link files to other packages, as well as applications that need a runtime environment (other tools, libraries, etc.) to make the application work properly.

A “package” or “packet” is a coherent block of source code (including meta-information and other data) made up of different functions that allows a task to be performed. Using packages means we must not code everything ourselves: If other developers have already developed a feature that interests us, why not reuse it?

Package dependencies are used to express these relationships. Package dependency means a package depends on another package as a prerequisite. Some packages require other packages to function. In Debian, packages can depend upon, recommend, suggest, break, or conflict with other packages (Burrows, n.d.).

Package managers solve the problem of dependency by ensuring that a package is installed with all the packages it needs. Various required packages (dependencies) are automatically installed when installing a package.

Assume, for example, that you want to install software that requires many other applications to function properly, such as Java, Microsoft C ++, or Dot Net Framework. You are therefore required to download and install this other software to resume installation of the one you really need. This is commonly referred to as software dependency. An application manager would solve this problem by automatically installing the dependencies of any software you want to install.

In a Linux environment, this principle already exists by default, making it easy to install software or packages. On a Mac, Homebrew also allows you to do this, which greatly facilitates the task. Even more popular, we have the app stores, which allow the installation and uninstallation of controlled applications.

Package Management Tools

Today there are many package managers that work on different operating systems such as macOS, Unix/Linux, and Windows.

A package manager provides a subset of tasks that can be performed from the command line: list, search, install, update, and remove packages. A package manager can also be run through a graphical user interface (GUI). Thus, some package managers use graphical interfaces, others use command line interfaces. Some examples of command line package manager tools are Scoop, chocolatey, and AppGet for Windows. Similarly, there are many package managers for OS X and Unix, like Yum, Npm, Yarn, Homebrew, and apt-get (Bowman, 2020).

Node.js

As a JavaScript runtime environment, Node.js is cross-platform, open source, and can run within a browser and on a server.

Next we will discover the use of NPM, the package manager for **Node.js**. NPM is an essential tool if you work on Node, Vue, React, or Angular. NPM works on different operating systems, including Windows, Linux, and Mac OS (Giraud, 2019).

NPM Package Manager

NPM (Node Package Manager) is the official package manager for Node.js (NPM, n.d.). It allows you to download and install packages for use on a project or, on the contrary, to share packages so that other users can use them.

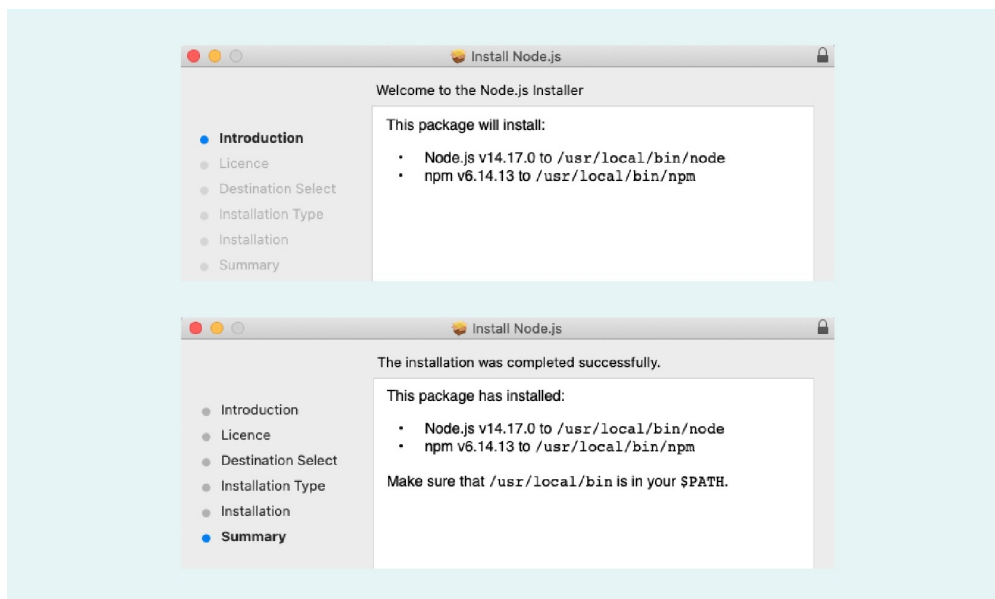
NPM was only designed to serve as a package manager for Node. However, as it is open source and was quickly adopted by a large community of developers, it has evolved into a more global package manager for all of JavaScript and its environments, thus going beyond the framework of Node.

NPM is now the world's largest package registry with over one million packages available. As developers, we will therefore often use NPM to install packages that interest us, such as React, Express, Moment, Request, Lodash, Async, Chalk, Sass, or even Debug (Giraud, 2019).

Install NPM on your macOS

The easiest way to install NPM is to install Node.js because NPM is integrated with it. To do this, go to the Node website and download the latest stable version (LTS). The installation of NPM on macOS or Windows systems is quite easy.

Figure 32: Installing Node.js and NPM on macOS



Source: Hadid (2021).

To ensure NPM is installed properly and working correctly, you can open your terminal and type `node -v`, which should return the version of Node.js installed and `npm -v`, which should return the version of NPM installed.

Code

```
Last login: Wed Jun 2 15:26:25 on ttys000
bin % npm -v
6.14.13
bin % node -v
v14.17.0
bin %
```

Using NPM

Once NPM is installed, you can install any package available from the registry from the command line. There are two ways to install a package with NPM: locally or globally.

To install a package locally, use the `npm install package_name` command, obviously replacing “package_name” with the name of the package you want to install.

To install a package globally, use the `npm install` command with the `-g` option. To install Sass globally, for example, we will type `npm install -g sass`.

Finding packages

To find a package to install, use the following command: `npm search package_name`. `package_name` is the name of the package to install, e.g., Hadoop. You can use the command below to verify if there is a git package to install:

Code

```
bin % sudo npm search git
NAME | DESCRIPTION          | AUTHOR   | DATE           | VERSION
git  | A node.js library...   | =christkv| 2013-06-24    | 0.1.5
```

From the result shown here, there is a git package that may be installed.

Find information about a package

If you want to get more information about a git package, use the following command, illustrated in the code box (`npm view package_name`).

Code

```
bin % sudo npm view git
Password:

git@0.1.5 | Proprietary | deps: 1 | versions: 6
A node.js library for git

dist

.shasum: 9ef62df93f851c27542143bf52d1c68b1017ca15

dependencies:
mime: 1.2.9
```

Notice it displays that a git package requires mime 1.2.9 to run properly (package dependency). Also, it shows some links where you can find more information on the git package.

List all installed packages

NPM has a simple command to display all packages installed on your system: `npm list`.

Code

```
bin % sudo npm list
Password:
/usr/local/bin
(empty)
```

Installing packages

The first main purpose of the NPM is to install new packages. For example, if we want to install git on your computer, we use the `npm install git` command. Then, after running the instruction, we can again display the list of installed packages. Note that git is installed along with its dependency package mime.

Code

```
bin % sudo npm list
/usr/local/bin
  git@0.1.5
  mime@2.3.9
```

Updating packages

To obtain a list of installed packages that are not updated (packages that have a newer version than the one installed), we can type the command `npm outdated` for local packages or `npm outdated -g` for global packages. To update out-of-date packages, we use the `npm update` command for locally installed packages or `npm update -g` for globally installed packages.

Removing packages

To remove a package, we use the `npm uninstall package_name` command.

Package dependencies

When installing a package, NPM automatically installs all the packages required (dependencies) for its operation. It is possible to view all the dependencies of a package in the form of a formatted dependency tree, using the command `npm list`.

Formatting installed packages in JSON format

To format the output of the installed packages in JSON format, you use the `npm list` command with the `--json` parameter:

Code

```
bin % sudo npm list --json
{
  "dependencies": {
    "git": {
      "version": "0.1.5",
      "from": "git",
      "resolved": "https://registry.npmjs.org/git/-/git-0.1.5.tgz",
    }
  },
  "dependencies": {
    "mime": {
```

```
    "version": "1.2.9",  
    "from": "mime@1.2.9",  
    "resolved": "https://registry.npmjs.org/mime/-/mime-1.2.9.tgz"
```

2.4 Upload/Deployment

Uploading

The file upload consists of transferring a file from the user's computer to the web server via networks (this is the reverse of the download operation).

To transfer files to a remote server, you will most often have to use the File Transfer Protocol (FTP). FTP is used for downloading and uploading files between a server and a client. The uploading operation is done using the address of an FTP server, and you must have the permission in advance to access the server.

Many applications operate in a client-server model, which means that a client (process that is part of the network) contacts a server (a process generally powerful in terms of input-output capabilities), which provides it with services. These services are referred to as client-to-server uploading. For example, the FileZilla application allows you to upload or download files onto a remote server; it is free and available for macOS, Linux, and Windows.

One of the most common uses of peer-to-peer (P2P) is file sharing. A peer-to-peer network model allows each part to function as both client and server. A client who owns a file (such as a film) makes it available to other clients by uploading it onto a download platform. In this situation, the client becomes a server. The same client can simultaneously also download other files shared by other clients (here, also servers). The practice has spread via applications such as Emule, BitTorrent or MTorrent, for the purpose of sharing multimedia files, such as movies or music albums.

Deployment

Web application deployment is considered the final and most important stage of the software development life cycle (SDLC). This stage, which occurs after the software design and testing, consists of implementing the application on each of the machines interconnected in the network. However, to be able to deploy and run applications, it is necessary to have servers targeted by the deployment. After the deployment process, the application is available to its intended users.

Deploying a software system is a process that takes place on multiple servers in a network. Today, almost all the actions required to deploy application servers and other libraries are still manual. It is necessary to connect to remote machines, send commands to transfer files, launch executables, or even set environment variables. It is therefore essential that

the deployment described is consistent so it does not corrupt the host servers concerned. This need for verification becomes more important as the size of the system to be deployed increases, such as in computing grids or ubiquitous environments.

Deployment Life Cycle

The deployment life cycle includes the following steps (Sun Microsystems, 2004):

1. Installation is responsible for transferring the software packages (libraries, executables, configuration files, etc.) to the concerned servers.
2. Configuration is the next step, in which the parameters of the installed software are defined according to the properties of the host server.
3. Activation (or startup) runs the installed and configured software properly. Once this step is completed, the software is ready to run.

Deployment Methods

There are several methods for deploying an application.

Manual deployment using FTP

Consider the following practical example: You must develop a website, but the client does not yet have hosting. Therefore, the development will be done on your local machine and then transferred by FTP to your server to test the site “under real conditions” and allow the customer to test it and give final feedback. Finally, after validation, the entire file will be uploaded to the client’s hosting.

In this example, the development environment is your local machine, the test environment is your server (also known as a staging environment), and the production environment is the customer’s hosting.

An application service provider (ASP) is a company that provides software or computer services to its customers through a network (the internet in general). There are many hosting providers, and the greatest benefit of this model is access to different levels of computing and network resources at different prices. ASP mode allows faster deployment in the company, saving the costs and delays associated with setting up a technical infrastructure (servers, network, software, etc.).

There are many hosting provider platforms. Providers offer developers software and hardware tools on their own infrastructure, which are typically those needed for application development and deployment. Thus, a provider frees developers from the obligation to install hardware and software in-house so they can develop, test, deploy, and run their applications. There are many examples of providers providing the tools and services needed to build applications. Major providers include Google, Microsoft, Amazon Web Services (AWS), IBM, Pivotal, Oracle, and Heroku (Chai et al., 2021). There are providers that offer free services or limited services, like Heroku. Amazon Web Services, Google Cloud, and Microsoft Azure do not offer free services, but they offer free credit when you subscribe to join. Some providers offer a very attractive monthly or yearly price.

Deployment automation

In contrast with the manual deploy, deployment automation allows you to deploy your whole website to testing and production environments with the push of a button. Deployment automation is extremely useful: it saves time and helps ensure the process is followed 100 percent of the time. In addition, some deployments are much more complex than an FTP transfer.

Deployment Tools

There are many tools for deployment automation that are not hard to find on the internet. Let us look at some tools available for automated deployment (Cooper, 2021).

Jenkins

Jenkins is the most popular open-source continuous integration tool. It was originally created as a build automation tool for Java applications. Since then, it has evolved to have more than 1400 plugins to easily integrate with other platforms and tools. Developers use Jenkins to build, integrate, test, and deploy their code automatically. It is intended to automate all construction and deployment tasks for software development projects of any size. You can install Jenkins on major operating systems like Windows or Linux because it runs on Java.

Octopus Deploy

Octopus is a software deployment tool, recognized for its rapid and reliable deployments. Octopus is a deployment automation tool for .NET developers. The Octopus system is free for very small projects (up to 10 deployment targets). For bigger projects, there is a monthly fee for Octopus Cloud and a yearly fee for Octopus Server (Cooper, 2021).

AWS CodeDeploy

AWS CodeDeploy is a service from Amazon that automates software deployments to various compute services like Amazon EC2 or your on-premise servers. It is a cross-platform application and may run on Microsoft Windows, Linux, and macOS. AWS CodeDeploy is free for the users of AWS EC2 and Lambda services. Otherwise, AWS CodeDeploy charges a small amount per update (pay-as-you-go model) (Eduonix, 2019).

GitLab CI/CD

GitLab CI/CD is a platform that covers all stages of the software development process, including building, testing, deploying, and monitoring the code. This platform works on different operating systems like Windows, UNIX, and Mac. Developers can use different programming languages like C, Java, Ruby, and PHP. A GitLab free trial is available for 30 days (Cooper, 2021). GitLab has four offers: free (but limited), bronze, silver, and gold, each with their own pricing (Eduonix, 2019).



SUMMARY

In this unit, we introduced the various web application development tools that allow developers to build web applications. We started by presenting the different types of development tools: code editors and IDEs. As an example, we looked in detail at the VS code editor.

Then, we present another essential tool, version management (also known as Version Control System). This tool is used to store and track the different versions of files. In order to clarify this tool, we presented the most used VCSs: Git and GitHub.

Next, we presented the package manager tools and package dependency. A package manager handles the automatic installation of a package and its dependencies. Some examples of package management tools used in different operating systems were given, such as Node Package Manager (NPM), one of the most popular package manager tools today.

Finally, we defined the web deployment tools that are considered the last stage of the software development life cycle. Web deployment is the process of making the web application available to its intended users. We also presented the manual and automation deployment and some example tools for automation deployment.

UNIT 3

STATIC WEB PAGE DEVELOPMENT

STUDY GOALS

On completion of this unit, you will be able to ...

- implement a web page using HTML.
- give your website a consistent layout using CSS.

3. STATIC WEB PAGE DEVELOPMENT

Introduction

In the early days of the internet, websites were kept statically on a server and transmitted to a user when called. Here, static means that the respective web page was transmitted to the end user in the same state it was available on the server, regardless of when, how often, or by whom the respective web page was called. Web pages could not be adjusted individually. If the content of a web page changed, it had to be reprogrammed accordingly and transmitted again as a static web page to the respective end users. Initially, this caused few problems because the amount of information that could be published via the web was very small by today's standards.

As the amount of information being published increased, it became necessary to develop simpler and faster ways of publishing information, which ultimately led to dynamic programming. Through dynamic programming, web pages were no longer static, but generated when the user requested them and tailored to their specific request. The data transmitted by a server is displayed in a special program running on the end user's computer: the browser. At first, browsers could only output and display text, e.g., the browser World-WideWeb (W3.org, n.d.). This was until the browser NCSA Mosaic was developed at the National Center for Supercomputing Applications (Mosaic Communications Corporation, n.d.), which could also display and output embedded elements such as graphics and interactive elements. In this unit, we will focus on the basic structure of static web pages, such as the HTML description language, which can be used to create a basic framework for web pages. From there, we will move on to more in-depth concepts such as CSS.

3.1 Fundamentals of HTML5

Hypertext markup language (HTML) structures documents such as web pages by formatting the content available on the web pages and converting pure information into a graphically descriptive representation (Robbins, 2014). HTML was developed to a significant degree by Tim Berners-Lee (Walter, 2008) at the European Nuclear Research Center CERN in Geneva (European Organization for Nuclear Research, n.d.). HTML can be described as the language that ultimately gives web pages (and websites) the appearance and structure that is seen by end users.

HTML's origin can be traced back to the development of the World Wide Web in the late 1990s (selfHTML, n.d.-b). One of the main goals of HTML was to create a description language that could run on any platform. At CERN, a way to efficiently distribute measurement data and research results was sought, so that, for example, the measurement data collected in Geneva could be analyzed by scientific institutions all over the world. For this, it was necessary to provide the raw data and a classical transmission protocol for ade-

quate interpretation of the obtained results. This led to the birth of the internet, which resulted in the first version of the HTML specification on November 3, 1992 (selfHTML, n.d.-b).

HTML differs from the programming languages discussed so far in that it knows no logical command flow structures (such as loops), but describes how the result (the web page) should ultimately look. To describe the respective commands, we work in tags, which begin in each case, for example, with a tag in the form `<h1>` and end with a tag in the form `</h1>`. This syntax is very similar to the syntax used in XML. For example, an HTML document is always introduced as follows:

Code

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title> Test Website </title>
5 </head>
6 <body>
7 <h1> Hello world </h1>
8 <p>This is my first HTML document </p>
9 <a href=http://www.iu-university.org/>I am a link</a>
10 
11 </body>
12 </html>
```

An HTML document always begins by declaring the document type. This ultimately serves to tell the browser what the document in question is about. This is an example of an HTML document, so the declaration can look like this: `<!DOCTYPE html>`. Then, the actual HTML elements follow. An HTML element always begins with a start tag in the form `< . . . >`, followed by the content of the respective tag and an end tag in the form `</ . . . >`. In addition, an HTML document can have a header element, in which, for example, the title of a web page is defined. This will be displayed in the browser or the tab label of the browser. The actual content of an HTML document is listed in a body element.

For example, `<h1>` is used to create a bold heading, after which `<p>` is used to introduce a paragraph. An `<a href>` tag leads to the definition of a link (the tag here is actually just “a,” “href” is, strictly speaking, an attribute that describes the characteristics of the “a” tag: here, around the exact link). An `` tag is used to embed an image into the HTML web page. The modern advancement of HTML to HTML5 has also enabled targeted animation control, the embedding of videos, and much more. If web pages need to be extended by logical structures and arithmetic operations, or, for example, database connections are desired, this cannot be realized with HTML alone and requires the use of PHP. The HTML tags `<a> . . . `, i.e., opening and closing tags, are also called HTML elements. Selected HTML tags, along with their functions, can be seen in the following table (selfHTML n.d.-b).

Table 2: Overview of Selected HTML Commands

Command/tag	Description
<h1>	Headline (highest level)
	Link
	Embeds an image
<head>	Metadata
<title>	Website title
<table>	Table

Source: Sturm (2020).

Attributes of HTML Tags

As mentioned above, attributes in HTML tags describe a more precise function. Through this, it is possible to assign further values to an HTML tag. Attributes are assigned within the tag, starting with the respective attribute and followed by the respective value of the attribute (separated by "="), which is typically specified in quotation marks.

 element

With an tag, it is possible to embed an image into an HTML web page: . "img" is an HTML element for controlling or including images in an HTML web page. With the "src" attribute, it is possible to define the exact image (via the exact file name and, if applicable, the address path to the respective file, whereby both local addresses and addresses on the internet can be considered here). If we want to define the size of the image to be loaded, we can use the additional attributes "width" and "height": . In this case, the image to be loaded, "IU.png," would be displayed with a width of 500 x 600 pixels. If an image cannot be displayed, it is often helpful to offer a user a text that roughly reflects what the image is supposed to represent. One way we can achieve this is by using an "alt" attribute. For example, if we want our website visitors to see the text "IU Logo" in case the image "IU.png" cannot be loaded, we ensure the alternative output of the text using . Some selected attributes of the tag are shown in the following table.

Table 3: Selected Attributes of the Tag

Attribute	Description
src	Path to image element
old	Text that is displayed when the desired image cannot be loaded.
width	Width of the image to be implemented

height

Height of the image to be implemented

Source: Sturm (2020).

Headings

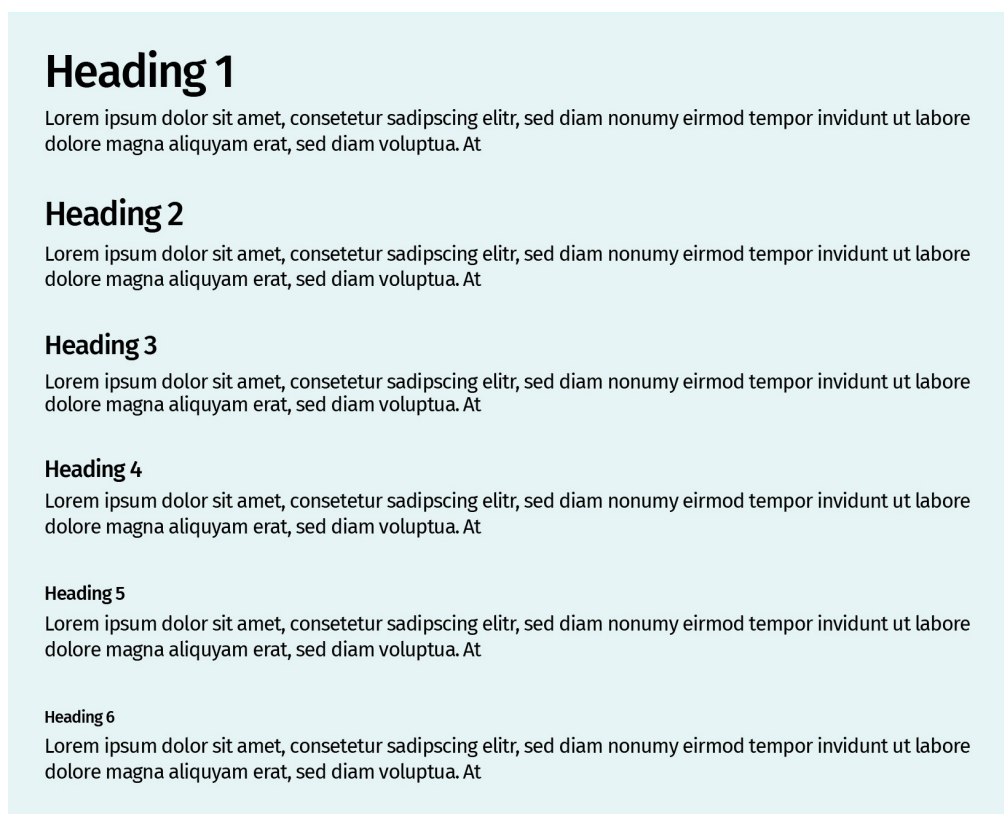
To make a web page, especially a text, appear structured and clear, we can use headings. HTML has, like most word processing programs, different hierarchy levels for headings. The `<h1>` tag is used to create the first hierarchy level of a heading. A heading lower in the hierarchy is then introduced by means of an `<h2>` tag. Let's look at the following example to illustrate this:

Code

```
<html>
<body>
<h1>Header 1</h1>
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At
<h2>Header 2</h2>
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At
<h3>Header 3</h3>
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At
<h4>Header 4</h4>
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At
<h5>Header 5</h5>
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At
<h6>Header 6</h6>
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At
</body>
</html>
```

This results in the following output on the user's screen.

Figure 33: HTML Headings Output



Source: Sturm (2020).

Lorem Ipsum

The example text used here (“Lorem ipsum...”) is a computer-generated filler text, which is often used as a placeholder in web design to give the external appearance a structure. This text is later replaced. This allows developers and designers to work on a project in parallel. “Lorem ipsum” appears to be written in Latin, but has no semantic or syntactic meaning.

Comments

Often, when writing an HTML document, it can be useful (or even necessary) to leave comments in the document, e.g., as a note to oneself or a colleague regarding how something was implemented and why. This is achieved in HTML as follows: `<!-- - I am a comment - - >`. The character combination at the beginning of a comment is interpreted by a browser such that it does not display the text contained in the comment in the user’s browser. This text remains in the source code, hidden from the user.

Paragraphs

HTML is also capable of inserting paragraphs. This is advantageous in that continuous text, which is formulated coherently but differs slightly in the subject matter, can be separated for better readability. We can start a new paragraph with a `<p>` element followed by the content of the paragraph. Once the paragraph is finished, the paragraph is closed with the `</p>` tag:

```
<p> Here is a paragraph. </p>
```

```
<p> Here is another paragraph. </p>
```

Links

A very common tool in web design is the use of links. Links ultimately serve to redirect a user, e.g., with a mouse click, from one page to another. Links are created in HTML using an `<a>` element followed by an “href” attribute:

```
<a href="https://www.iu-university.org/">IU</a>
```

Here we encounter another challenge: we may not want the link to be displayed as the actual URL, as this would affect the readability of the web page, especially when dealing with very long URLs. Instead, we can close the angle brackets after the “href” attribute and specify the text with which the included link will be labeled. The link element is then closed with the `` tag.

Lists

It is often necessary to enumerate individual elements and give them an external structure. One way to achieve this is using lists. HTML is capable of generating both unordered and ordered lists.

We define an unordered list with an `` element. Elements (the respective “list items”) are input with the `` tag. With the following short HTML implementation, we can create an unordered shopping list:

Code

```
<html>
<body>
<h2>Shopping list</h2>
<ul>
<li>Coffee</li>
<li>Water</li>
<li>Milk</li>
<li>Ice cream</li>
</ul>
</body>
</html>
```

In the browser, this unordered list is then displayed as follows.

Figure 34: HTML List Output



Source: Sturm (2020).

The unordered list shown here can be quite easily converted to an ordered (numbered) list in HTML by creating the list using an `` (“ordered list”) tag instead:

Code

```
<html>
<body>
<h2>Shopping list</h2>
<ol>
<li>Coffee</li>
<li>Water</li>
<li>Milk</li>
<li>Ice cream</li>
</ol>
</body>
</html>
```

This then leads to the following output.

Figure 35: HTML Ordered List Output



Source: Sturm (2020).

Tables

It is also possible to insert tables in HTML, e.g., in order to make information clear and comprehensible. A table is defined in HTML with the `<table>` tag. The `<tr>` tag then specifies the individual rows of the table. The respective header, i.e., the column heading of a table, is created with the `<th>` tag. The respective entries (or table data) in a cell are embedded as `<td>` tags. An HTML table is structured in such a way that the description of the table ultimately becomes increasingly specific, i.e., from the table declaration itself to the content of a specific cell. The formatting HTML uses is already stored in the HTML standard, so entries in the `<th>` element will always appear bold and centered.

In the following example, we will try to compile the stock of a small warehouse in a HTML table. The hypothetical inventory system should have options for entering an item description, an item number, the price, and the quantity available in the warehouse. Three articles (1-3) are then to be entered into the table as an example. A possible implementation of such a table is as follows:

Code

```
<html>
<body>
<h2>stock</h2>
<table>
<tr>
<th>Article</th>
<th>Article no</th>
<th>Price</th>
<th>Quantity</th>
</tr>
<tr>
<td>Article 1</td>
<td>1234</td>
<td>50</td>
<td>94</td>
</tr>
<tr>
<td>Article 2</td>
<td>4567</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>Article 3</td>
<td>5794</td>
<td>10</td>
<td>45</td>
</tr>
```

```
</table>
</body>
</html>
```

The corresponding output in the browser would look like this.

Figure 36: Example Table: Stock

Stock			
Item	Item no.	Price	Quantity
Item 1	1234	50	94
Item 2	4567	25	20
Item 3	5794	10	45

Source: Sturm (2020).

ID of an HTML Element

Similar to how a name identifies a person, we can also assign names to individual HTML elements in order to identify them again later. For example, if we have a heading that is described by an `<h1>` element, we can assign a name to it using the “id” attribute. In concrete terms, an implementation would look like this:

```
<h1 id="FirstHeader">I am a header</h1>
```

The IDs of an HTML document are not displayed in the browser. They are only used for internal reference. The declaration of an ID for an HTML element is case sensitive. This means that even identical names will result in correspondingly different IDs once different upper- and lowercase letters are used.

So, what are IDs useful for? With IDs, it becomes possible to access or refer to the element with the desired ID at (almost) any place in the HTML document (this is typically achieved by means of a pound sign (#) placed in front of the ID). For example, we can introduce a style element into the header of an HTML document, which is opened with `<style>`, followed by the respective content, and closed with `</style>`. For example, if we want to set the background color of the heading to blue, we would use the following code.

Figure 37: HTML IDs

```
<!DOCTYPE html>
<html>
<head>
<style>
#FirstHeading {
  background-color: blue;
}
</style>
</head>
<body>

<h1 id="FirstHeading">I am a heading</h1>

</body>
</html>
```

Source: Sturm (2020).

Here we have also used a style definition, which actually follows the syntax of Cascading Style Sheets (CSS). In the style element, the heading with the respective ID is addressed by “#FirstHeading” and applied to the commands that are listed in curly brackets that follow. In this case, this refers only to the background color, i.e., “background-color,” to which we assign the value “blue.” This is confirmed with a semicolon. In the user’s browser, this would lead to the following output.

Figure 38: HTML IDs Output



Source: Sturm (2020).

Classes

In the same way IDs can be assigned to HTML elements, it is also possible to aggregate several HTML elements into a class. A class definition is introduced by the “class” attribute. If this is accessed again in a later step, the same operation is applied to all class elements. While accessing IDs takes place by means of a pound sign, this is done for classes by means of a prefixed period (.). Let’s look at another example.

Figure 39: HTML Classes

```
<!DOCTYPE html>
<html>
<head>
<style>
#FirstHeading {
  background-color: blue;
}
.SecondHeading {
  background-color: yellow;
}
</style>
</head>
<body>

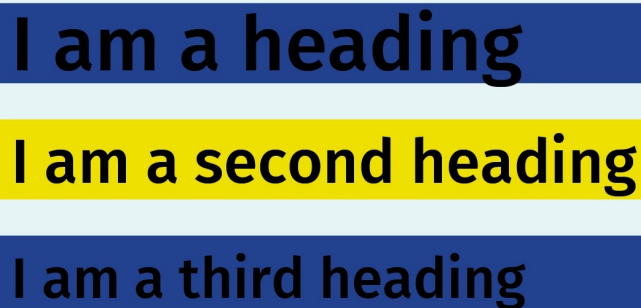
<h1 class="FirstHeading">I am a heading</h1>
<h1 class="SecondHeading">I am a second heading</h1>
<h1 class="FirstHeading">I am a third heading</h1>

</body>
</html>
```

Source: Sturm (2020).

In this example document, we have embedded some headings that are assigned to the classes “FirstHeading” and “SecondHeading.” We have now specified a style definition for each of the two classes in the style element. All elements that belong to the class “FirstHeading” have a blue background. All elements of the class “SecondHeading” have a yellow background. This leads to the following output in the browser.

Figure 40: HTML Classes Output



The image shows three lines of text, each on a separate line, representing the output of the HTML code. The first line, "I am a heading", is on a blue background. The second line, "I am a second heading", is on a yellow background. The third line, "I am a third heading", is on a blue background.

Source: Sturm (2020).

div class

A very commonly used class is the div class. By means of div elements, it is possible to group text, namely the text that is inserted inside the div element:

```
<div> Here comes the text </div>
```

We will see below, especially in the context of CSS, that the use of such div classes offers some advantages when it comes to setting the design for an HTML document.

Creation of a Form Using HTML

Forms in web applications are typically still generated using HTML. A form can generally have the following form elements:

- single-line and multi-line text fields for input
- elements for file upload
- selection lists
- text fields by means of which the arrow keys can be used to increase or decrease the corresponding values
- radio buttons for either/or selection
- checkboxes for both/and selection
- dates
- buttons

For these elements, HTML has `<form>` tags, which allow the creation of forms, or, in essence, input fields. A form is placed between `<form>` and `</form>`. Everything that is entered inside this tag is recognized as form content when the HTML code is later processed by the browser. We can generate the first input field using the command below.

Figure 41: HTML Input Field

```
<label for="fname">First name:</label><br>
<input type="text" id="fname" name="fname" value="John"><br>
```

Source: Sturm (2020).

This has the heading “First name.” Using `
` moves us to the next line and we use the `<input>` tag to define what form and input line should accept. Using the “type” attribute, we define a “text,” i.e., an input line is created into which text can be entered. Using “id,” we can assign a unique identifier to the input text line. This will become important later, for example, when we want to collectively read the data from the form to forward it by e-mail or store it in a database. We can also assign a name to the input line. This is done using “name.” Here, we assign the name “fname.” Using “value,” we can define which default value should be present in the input field when the form is first called. This can help the end user or the customer know in which format an address, for example, should be entered. Ultimately, however, assigning “id,” “name,” and “value” is an optional step. We can now generate an entire form to record a customer’s address. A possible implementation in HTML would look like the following figure.

Figure 42: HTML Input Form

```
<html>
<body>

<h2>Order form</h2>

<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Smith"><br>
  <label for="address">Address:</label><br>
  <input type="text" id="address" name="address" value="25 Long Road"><br>
  <label for="city">City:</label><br>
  <input type="text" id="city" name="city" value="Munich"><br>
  <label for="zipcode">ZIP code:</label><br>
  <input type="text" id="zipcode" name="zipcode" value="81673"><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

Source: Sturm (2020).

Using the value attributes, the corresponding input cells are pre-filled with a default value. However, these can be changed by the user. Just as in the procedure already described, we now implement the individual form fields in the `<form>` tag, which can be found in the HTML body, and extend our previous line for entering the first name with a field for the last name, the address, the place of residence, and the postal code. As we would expect from an input form on the internet, a form ends with a “submit” button, which is responsible for sending the form (submitting an order or a contact form). This is achieved in HTML by means of an input tag of the type “submit.” Using the value option, it is possible to specify a label text for the button. In this example, we label it “Submit.” In a browser, the form implemented in HTML above would appear as follows.

Figure 43: Order Form

Order Form

First name:
John

Last name:
Smith

Address:
25 Long Road

City:
Munich

ZIP code:
81637

Submit

Source: Sturm (2020).

Admittedly, the form does not yet look particularly appealing. This form is a purely functional and technical implementation. We can visually polish the form further through the additional use of CSS stylesheets or even special JavaScript classes, but this is outside the scope of this introductory course.

Drop-down menus

Forms in HTML are not purely limited to the text input fields seen above. We can also create drop-down menus that, for example, allow a customer to select a specific color of the item they wish to purchase. A possible implementation for a drop-down color selection menu could look like this.

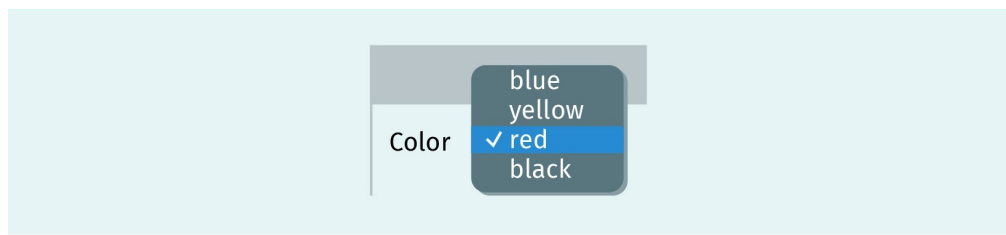
Figure 44: HTML Drop-Down

```
<form>
  <label for="color">Color:</label>
  <select id="color" name="color">
    <option value="blue">blue</option>
    <option value="yellow">yellow</option>
    <option value="red" selected>red</option>
    <option value="black">black</option>
  </select>
</form>
```

Source: Sturm (2020).

We create a drop-down menu using `<select>` tags, where the individual options available for selection in the drop-down menu can then be specified. The value attribute is the value that is assigned to the selected field. For example, it is possible to assign a number when selecting the color blue, which can be useful if, for example, different colors of a t-shirt have different prices. For example, we can provide the selection option “blue” to the end user but use the price of the shirt internally for invoicing. If we look at the selection option for the color “red,” we notice the additional option “selected.” This is the value of the drop-down menu that is listed as the default when the form is called. In the browser, the above form would be displayed as follows:

Figure 45: Menu Options



Source: Sturm (2020).

For the actual reading and sending of the form data, however, we need help from more advanced concepts, such as PHP.

3.2 Fundamentals of CSS3

CSS is the abbreviation for the term “cascading style sheets” and describes how HTML elements should be displayed (selfHTML, n.d.-a). CSS allows for a better, more formative conversion of information: A major advantage of defining the appearance of HTML pages using CSS is that the appearance of HTML elements only needs to be defined once. This can then be applied to the entire HTML document without the need to further redefine the appearance for each HTML element used.

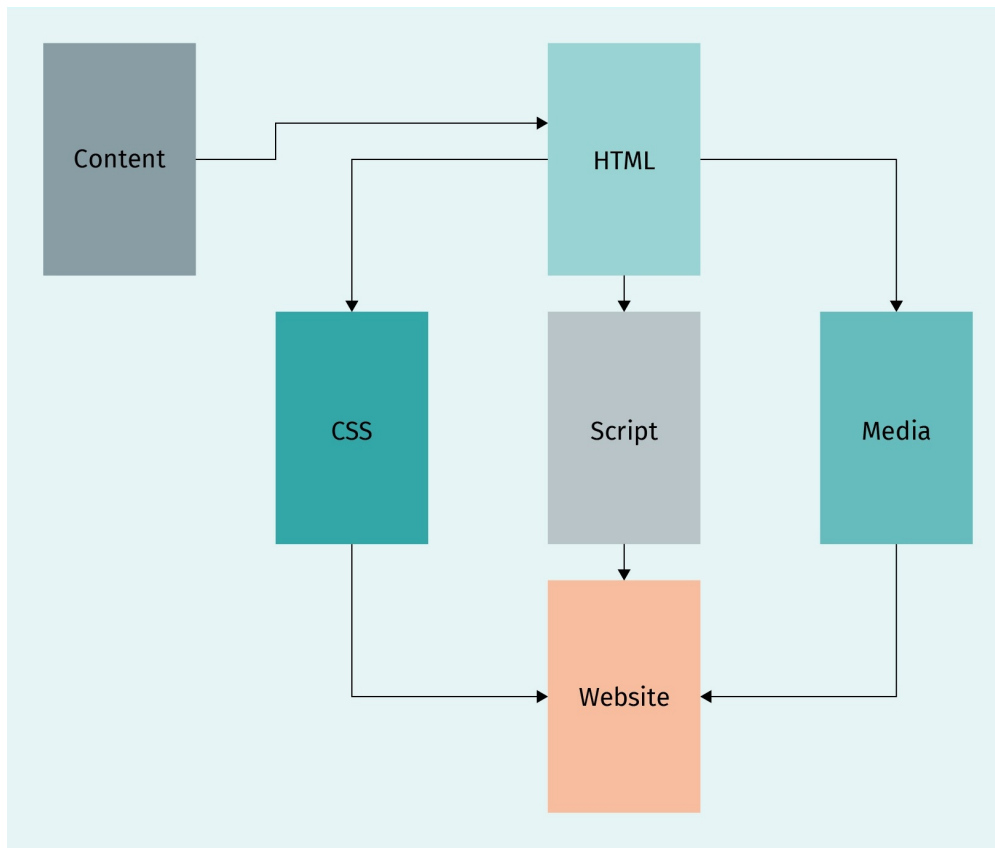
CSS Structure

Selector
A selector commands the selection of a specific HTML tag, which should be covered by the CSS formatting defaults.

Declaration
A declaration defines the appearance of the HTML tag it covers.

CSS is always structured in the same way: a CSS rule consists of a **selector** and a **declaration**. A selector is an element that specifies which HTML elements the declaration following the selector should refer to. The declaration then determines how the HTML element in question is to be formatted. The declaration can be made either in a separate CSS file, which can be stored together with the HTML document on a server, or stored in the style element directly in the HTML document. The interaction between HTML, CSS, and other media is described in the following figure.

Figure 46: Interaction Between CSS and HTML



Source: Sturm (2020), based on Vonhoegen (2018), p. 414.

Content is inserted into an HTML document, the subsequent presentation of which can be described using HTML. The exact formatting of the respective descriptive HTML tags is determined by a CSS, which interacts with the HTML document and is ultimately aggregated to the final web page. Based on the HTML document, further functionality can be added to the web page by integrating additional scripts (e.g., JavaScript). The integration of media (e.g., images) is also possible.

Let's look at this again using an example in which we want to have all HTML elements declared as `<p>` elements left-aligned and displayed in blue. The CSS description required for this would be as follows.

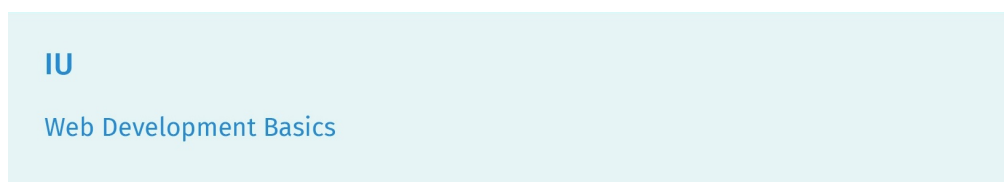
Figure 47: CSS in HTML

```
<html>
<head>
<style>
p {
    color: blue;
    text-align: left;
}
</style>
</head>
<body>
<p>IU!</p>
<p>Web Development Basics</p>
</body>
</html>
```

Source: Sturm (2020).

As already mentioned, the CSS description in the style element takes place directly in the HTML document. If a <p> element is then defined in the further course of the HTML document, it is displayed according to the CSS description. The following output would appear on the user’s screen.

Figure 48: CSS in HTML Output



Source: Sturm (2020).

Comments

For better clarity and comprehensibility, it is also possible to store comments in CSS. A comment can be introduced by “/*,” followed by the actual comment text, and closed again with “*/.” For example, we can introduce a comment as follows:

```
/* I am a comment */
```

Background Color

In CSS, it is not only possible to change the font color, but also to set the background color of a web page or HTML element. Using the CSS command “background-color,” it is possible to change the background color of a web page. Again, as explained above, the structure of the CSS statement is identical (built from selector and declaration). Using the following command, the background color of the “body” area is set to blue.

Figure 49: CSS Background Color

```
<style>
body {
    background-color: blue;
}
</style>
```

Source: Sturm (2020).

The body area is generally understood to be the background of a web page.

Margins

Using margins, it is possible to define an HTML element's distance from the edge of the screen or browser window. Let's look at this again with an example.

Figure 50: CSS Margins

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    border: 2px solid black;
    margin-right: 150px;
    margin-left: 80px;
    margin-top: 100px;
    margin-bottom: 100px;
    background-color: yellow;
}
h2 {
    color: blue;
    margin-left: 80px;
}
</style>
</head>
<body>
<h2>IU</h2>
<div>Web Development Basics</div>
</body>
</html>
```

Source: Sturm (2020).

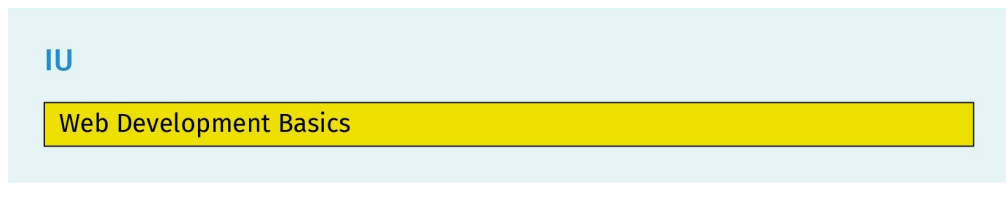
For the div element, we use the margin commands to define the respective margins, e.g., margin-left defines the left margin in pixels. Using border, it is possible to draw a frame around the respective HTML element. In this example, the border should be 2 pixels wide, solid, and black. The background color for the div element is set to yellow in this example.

For the heading h2 we want it to be blue in color and flush (relative to the left side) with the div element.


The body element is followed by the implementations of the h2 element and the div element.

This results in the following output on the user or developer's screen.

Figure 51: CSS Margins Output



Source: Sturm (2020).

 **SUMMARY**

Using static web pages, it is possible to store a hard-coded web page on a server. This has the exact same appearance and retains the same information each time it is called. This also means that the web page itself must be modified directly if its content is to be changed.

The basis for the use of static web pages is HTML as markup language. HTML can be used to specify how a web page should look and be formatted through description and in tag notation. The web page itself is rendered and displayed in the user's browser.

More specific formatting of HTML can be achieved using cascading style sheets (CSS). Using CSS, HTML tags can be addressed by selectors and formatted by declarations. This allows for a consistent appearance throughout the HTML document. The use of CSS also avoids excessive code and, thus, redundancies, which may make the code in question appear confusing.

UNIT 4

ADVANCED DESIGN TECHNIQUES

STUDY GOALS

On completion of this unit, you will be able to ...

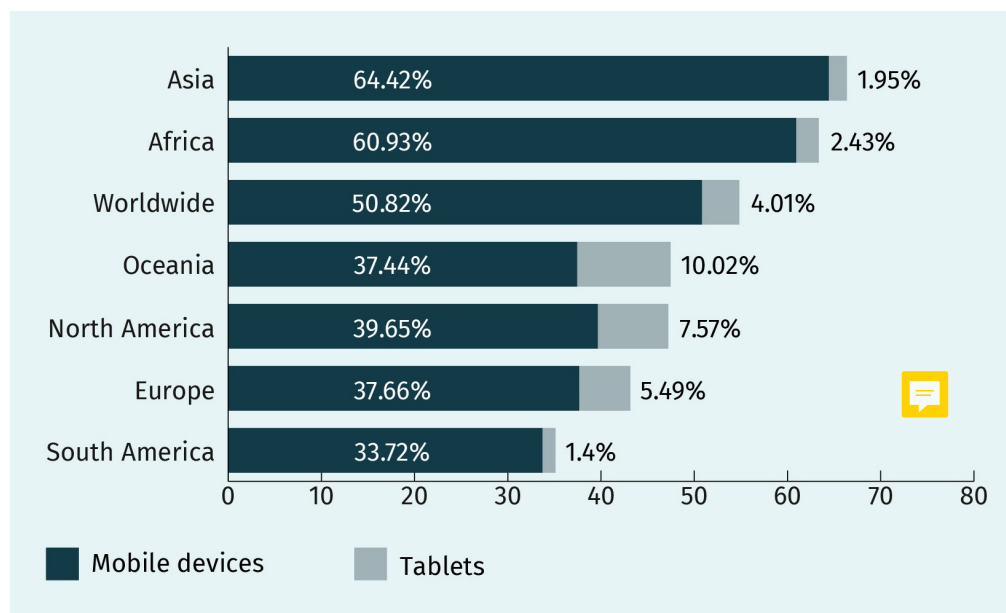
- describe the purpose of responsive user interfaces.
- discuss the advantages of responsive web user interfaces.
- consider what is important when designing responsive web user interfaces.
- explain how behind layout types, breakpoints, media queries, and layout patterns function.
- use the Bootstrap framework to build fast and responsive, mobile-first websites.

4. ADVANCED DESIGN TECHNIQUES

Introduction

As the internet becomes more widespread, the number and variety of the devices through which users access the internet has also grown. While websites were once accessed primarily via desktop computers, the proportion of website page views via mobile devices is increasing. In Asia and Africa, for example, more than 60 percent of website calls were made via mobile devices in 2018, while in Europe the share was just under 38 percent (Statista Research Department, 2019).

Figure 52: Share of Page Views from Mobile Devices in 2018



Source: Statista Research Department (2019).

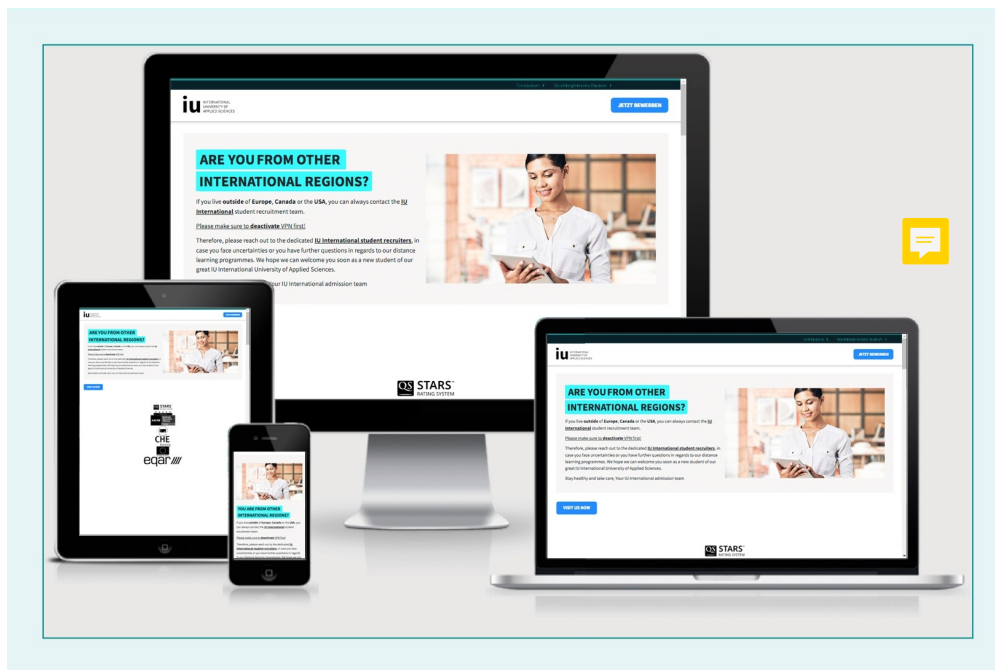
Responsive user interfaces offer a way to present website content in a user-friendly and visually pleasing way on various screen sizes. We will look at the basics of responsive user interfaces, some of their challenges, and what should be considered when designing responsive websites.

4.1 Responsive Web Design

The term responsive web design refers to a collection of measures that aim to design a website in such a way that it is optimally adapted for various visual output devices. For example, a smartphone in landscape or portrait mode can serve as the visual output device for the website. Alternatively, a smart watch, tablet, laptop, large monitor, or even a

high-resolution television screen could be used for this purpose. Thus, screen sizes can vary greatly. A responsive website is designed so that its content can be optimally adapted to different display sizes, which is illustrated below with the example of the IU website. Users often have different devices through which they want to access website content. Thus, it is necessary that the mobile variant communicates the same content as a desktop variant. The visual design and structural layout must also not change too much so that users do not get lost.

Figure 53: The IU.org Website on Different Devices



Source: Avery (n.d.).

Responsive websites enable optimal adaptation to different screen sizes by relying on three core elements: a fluid layout, adaptive content, and layout breaks through media queries. These elements are explained in detail below.

A fundamental challenge in responsive web design is the understanding of, as well as the approach to, design. In the past, websites were designed mostly for fixed window sizes, but, in responsive web design, it is necessary to make the content adaptable.

Wroblewski (2009) coined the “mobile first” design approach. Previously, the design of a website began with the desktop version, followed by the creation of a mobile version for smaller screens by means of “graceful degradation.” This means that the corresponding desktop web page will be adapted, or in most cases, reformatted to fit mobile requirements as gracefully as possible. Wroblewski (2009) cited three main reasons for a mobile-first approach, which is becoming increasingly widespread:

- Page views via mobile devices have grown strongly in percentage terms in recent years. Starting with the design of the mobile version ensures that companies have developed a well-conceived design solution for the rapidly growing market.
- The design of the mobile version, with their small screen sizes in mind, forces designers to limit themselves to the essentials. This focus on the key functions of the website means that users can reach the information quickly and without detours. This results in a streamlined interface with high user satisfaction, which has a positive impact on both the success of the site and the user experience.
- Mobile devices, such as smartphones or tablets, offer additional possibilities compared to a classic website. For example, by including position data via GPS or acceleration data via the sensors of the mobile device, additional information can be added to provide even more personalized (additional) offers. For example, when searching for a restaurant via a mobile device, the user's location can be included in order to present the search results prioritized according to distance. These enhanced functionalities can, in turn, increase usability.

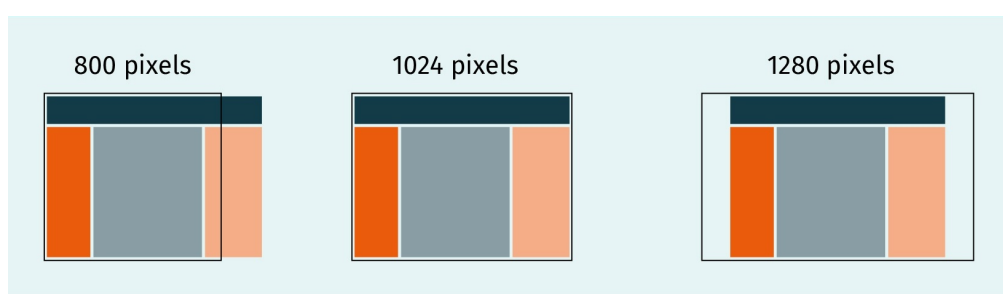
4.2 Page Layout

The layout is the first component of a responsive website. To explain the difference between a responsive layout and other layout approaches, we will first introduce the different variants.

Fixed Layout

A fixed layout cannot be adapted to different screen sizes. The dimensions of the fixed layout are given in pixels. If the screen is too small, scroll bars are displayed. For mobile devices, the complete layout is sometimes scaled down linearly to a default value (Ertel & Laborenz, 2017).

Figure 54: Fixed Layouts Cannot Adapt to Their Environment



Source: Ertel & Laborenz (2017), p. 29.

Fluid Layout

The fluid (or flexible) layout is defined in percentages in relation to the display window. The dimensions of the layout then change with the size of the window. Content elements, such as images and texts, remain at their original size (Ertel & Laborenz, 2017).

Figure 55: Fluid Layouts Always Assume the Same Relative Dimensions with Respect to Their Environment

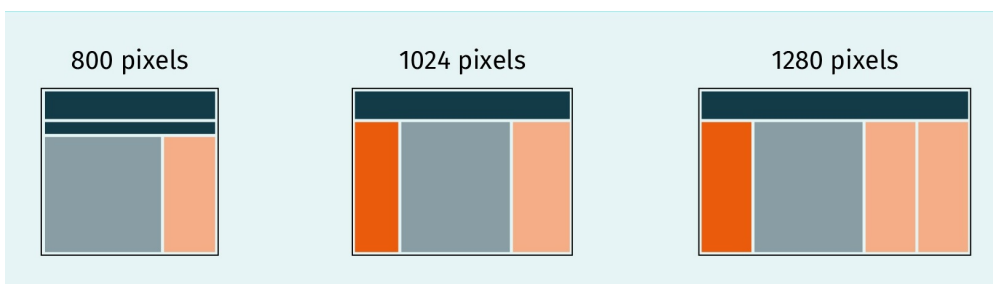


Source: Ertel & Laborenz (2017), p. 29.

Adaptive Layout

The adaptive layout is basically a fixed layout that exists in different variants. If the window size is changed, the layout “jumps” between the variants. Consequently, the adaptive layout adjusts itself between several fixed display sizes. The change between the fixed layouts takes place at defined break points. Between individual break points, on the other hand, the layout behaves like a fixed layout (Ertel & Laborenz, 2017).

Figure 56: The Adaptive Layout Offers Jump Adjustments between Different Fixed Display Sizes

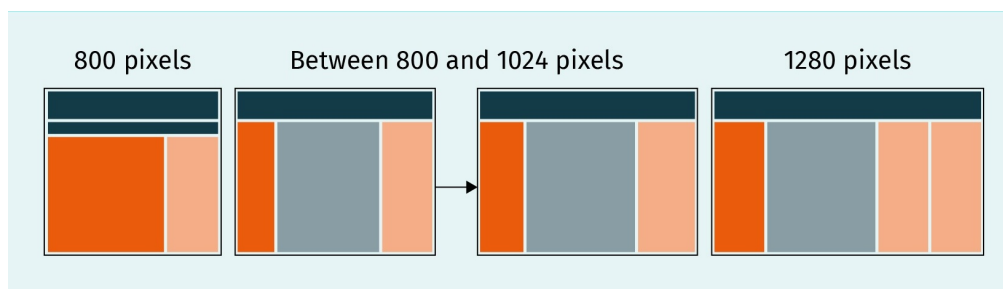


Source: Ertel & Laborenz (2017), p. 30.

Responsive Layout

The advantages of both the adaptive and the fluid layout are combined in the responsive layout. It contains breakpoints like the adaptive layout, outside of which the layout changes significantly. Between the breakpoints, however, it behaves like a fluid layout. The responsive layout is also equipped with scalable content: images, for example, adapt to the available space (Ertel & Laborenz, 2017).

Figure 57: The Responsive Layout Combines Positive Aspects from the Adaptive and Fluid Layouts



Source: Ertel & Laborenz (2017), p. 30.

Grid Systems

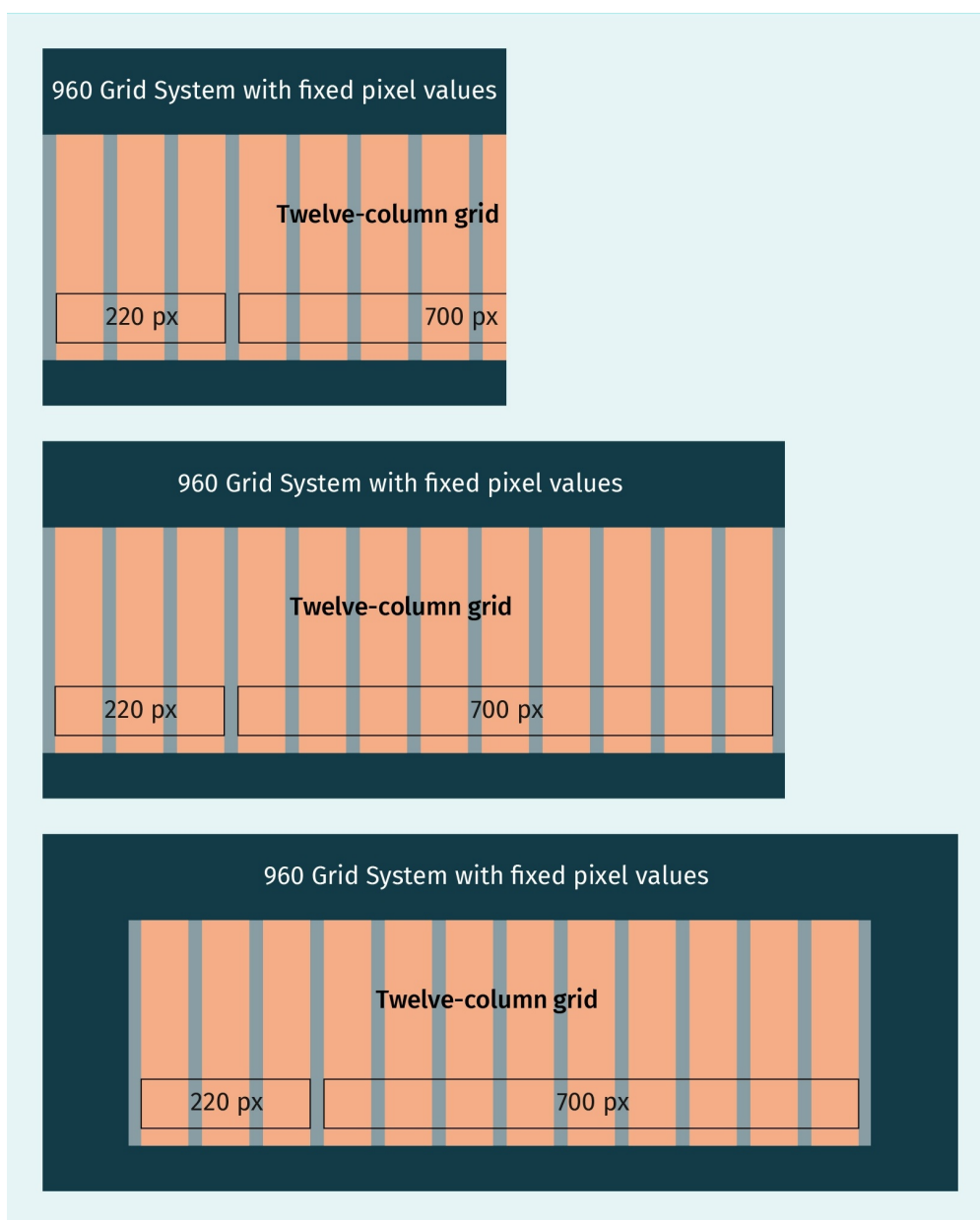
Grid system
By using a responsive grid system, the web page is divided into columns. These columns will have a total width of 100 percent. They will shrink or expand if the browser window is resized.

What are the functions of grids in web design? Modern websites are based on a **grid system**. A grid system is an invisible system of lines along which various design elements such as images and buttons can be aligned. This supports the web designer and leads to a structured and harmoniously aligned layout. If a grid is used in the background, web designers also have the possibility to deliberately allow individual elements to break out of the grid in order to let them come strongly into the foreground of the viewer. Thus, grids serve as a useful design tool for web designers. In contrast to graphic and print design, however, grids are not derived from the design, but are constructed first. Therefore, the layout of the website is based on the previously established grid. This is necessary because there are no fixed areas for display on the web. These vary greatly depending on the window size of the browser and the user's terminal device (Hellwig, 2015).

In the past, the 960 grid system was very popular for static layouts. It is useful for us to learn about the 960 grid system, as it serves as the basis for understanding more complex grid types. As the name suggests, the static 960 grid system is based on a total width of 960 pixels. This width has been used to create a workspace that can, in turn, be easily subdivided. The 960 pixels can be displayed easily in both twelve-column and sixteen-column grids. Based on this grid, the various design elements of the website can be aligned (Hellwig, 2015).

The disadvantage of this static grid system, with its fixed layout and width of 960 pixels, is that content can be "lost" on smaller screens or can only be accessed via a horizontal scroll bar. With 960 pixel-wide screens, the layout fits exactly, while space is wasted with larger screens (Ertel & Laborenz, 2017).

Figure 58: Behavior of a Static Twelve-Column Grid System with a Fixed Layout and a Width of 960 Pixels

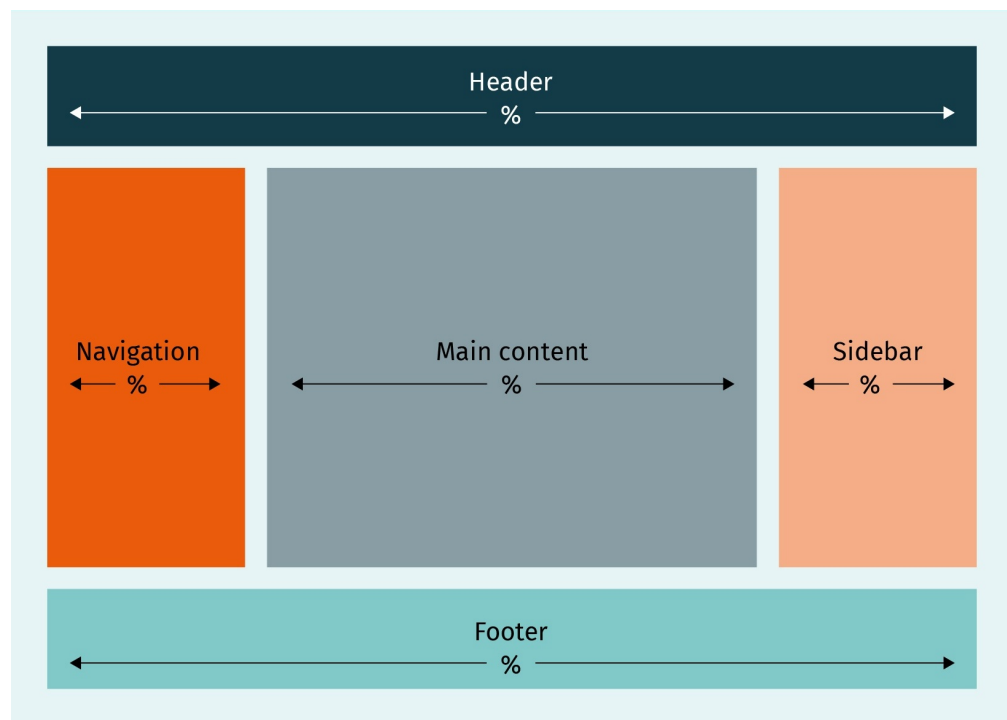


Source: Ertel & Laborenz (2017), p. 30.

To design a responsive web page (or website), the creation of an adaptive grid system is necessary. Therefore, grid systems with fixed pixel values must be converted into a fluid grid with percentage values (Ertel & Laborenz, 2017). Web frameworks such as Bootstrap can assist in the development, as existing templates already specify fluid grid systems.

A widely used grid system based on the fluid layout uses a header area followed by a three-column layout with navigation, main content, and sidebar areas, followed by a footer. In contrast to the fixed layout, the column width of the fluid grid system is specified in percentages, and thus adapts smoothly to different screen sizes (Hellwig, 2015).

Figure 59: Example of a Grid System Based on the Fluid Layout

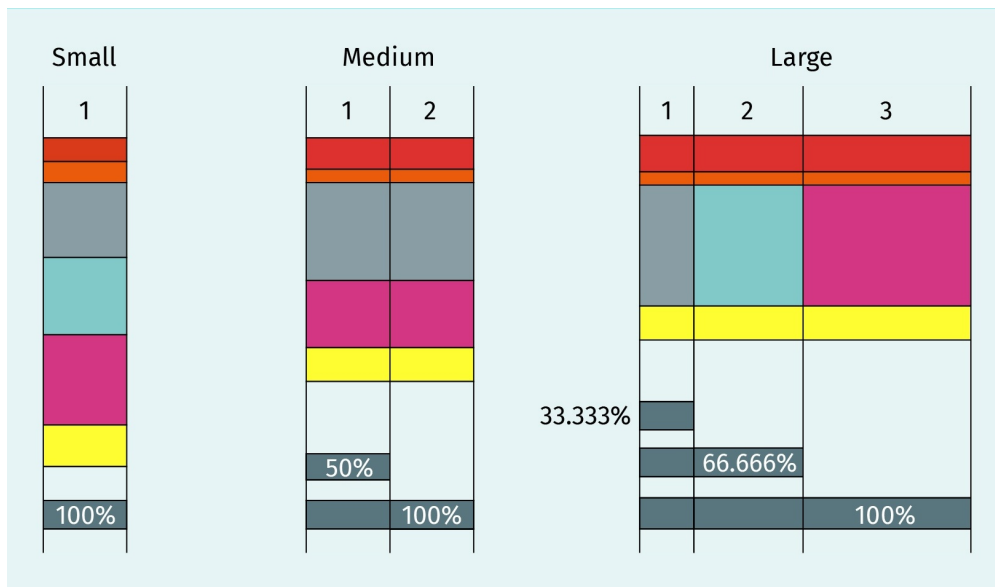


Source: Hellwig (2015).

Breakpoints

As explained above, the responsive layout is based on the fluid layout. At predefined thresholds, also called breakpoints, the layout switches between different grid systems. Breakpoints therefore allow even more flexibility when designing for different screen sizes by creating layout breaks. For example, a single-column grid system can be used for small screens (e.g., smartphone screens). At a predefined breakpoint, for example, a window width of 640 pixels or more is switched to a two-column grid. For larger screens, a three-column grid is used. At the breakpoints, the system switches between different grid systems depending on defined window sizes, which are known as viewports. The breakpoints ensure the flexibility of responsive layout systems.

Figure 60: Example of a Responsive Grid System: One Column on Small Screens, Two on Medium Screens, and Three Columns on Large Screens



Source: Hellwig (2015).

Both adaptive and responsive layouts work with breakpoints. The key difference between the two systems is that the responsive layout is more flexible, as the content is scaled fluidly between the breakpoints. The advantages and disadvantages of adaptive and responsive layouts are as follows (Ertel & Laborenz, 2017):

- Both layout types adapt for different screen sizes. Thus, they function better than a fixed website layout on mobile devices.
- With adaptive layout, **pagination** only occurs at assigned breakpoints and the page remains unchanged in between. The website is therefore only optimized for the previously designated screen sizes. In addition to the disadvantage of jumpiness, space is sometimes not optimally utilized.
- The adaptive layout gives the designer more detailed control compared to the responsive layout.
- The responsive layout offers more flexibility for all screen sizes.
- Whether implementing either the adaptive or responsive layout is a higher requires more effort depends on the layout pattern, the number of break points, and the extent to which the design elements of the web page (e.g., images) should change.

Pagination

The process of splitting the content of a web page into discrete pages to simplify viewing is called pagination.

4.3 Media Queries

Along with flexible layout grids and adaptive content, media queries are one of the three central components in responsive web design. Media queries are the core technology used for switching between different layouts at breakpoints. This allows different cascading

style sheets (CSS) to be assigned to different output media, such as smartphones, tablets, or large screens. This allows us to take the different properties of the output media to be taken into account.

In simple terms, the web browser on the respective end device receives the website as an HTML document after a web page call. The “hypertext” in HTML means that documents are linked. Linked documents allow interaction, unlike linear documents such as books, whose format cannot change. Hypertext documents are a foundation of the World Wide Web and are rendered by web browsers. Web browsers create a user interface that is then experienced by the user. HTML is used to create, structure, and link information on a website. Multimedia content can also be integrated. However, the visual presentation is not specified by means of HTML, but is determined by the web browser with the help of design templates such as CSS. CSS was developed as a standard for the presentation of the contents of a website. Thus, design features such as colors, layout, or typeface are specified independently of the HTML structure. Only by changing the style sheet can websites be displayed with a completely different design. This is impressively demonstrated by Dave Shea (n.d.) on the CSS Zen Garden website, for example.

With responsive websites, it is possible to switch between different layouts at defined breakpoints. For example, a screen width of 768 pixels or more can be switched to a multi-column layout using media queries. Different properties can be assigned to different screen sizes using CSS. Between the defined breakpoints, responsive websites behave like a fluid layout. When the breakpoint is reached, however, media queries are used to switch to an adapted layout. This is how the high flexibility and adaptability of responsive websites to different visual output devices is achieved. Web frameworks such as Bootstrap also support the development of dynamic websites and offer solutions for defining breakpoints and controlling them via media queries.

4.4 CSS Frameworks (Bootstrap)

In this section, we will understand how to use Bootstrap to format our HTML files and explore the CSS styles applied directly to our elements. We will discuss responsive grid systems and mobile-first support with Bootstrap, as well as the display order of the various extensive prebuilt components.

Brief history of Bootstrap

Bootstrap, originally named Twitter Blueprint, was created in 2011 by Mark Otto and Jacob Thornton, two employees of the Twitter company (Bootstrap Team, n.d.). Initially, Bootstrap aimed to offer a library of standard components that would speed up and standardize the development of user interfaces. Bootstrap was designed to reduce redundancy and increase adaptability and reusability among different internally developed web applications. This speeds up the development process, as it is no longer necessary to repeatedly create basic foundations. For example, usually a website will have a sign-in/sign-up system, so it makes sense to reuse existing code for this.

After using the Twitter Blueprint framework within **Twitter**, contributions from internal developers towards the project (it was renamed “Bootstrap”) increased. Shortly after, Twitter decided to release Bootstrap 1 as an open-source project to the community. Its code has been available since August 19, 2011. Since then, a team led by Otto and Thornton, as well as a community of developers, **have helped** accelerate Bootstrap’s growth.

After many contributions from the core team and the community, Bootstrap 2 appeared in early 2012. This version provides developers with new graphics components: progress bars, button groups, carousels, etc. (Hussain, 2017). The most important novelty of Bootstrap 2 **is** the implementation of a system that allows the creation of fully responsive websites (using a 12-column grid system) and web applications that adapt to mobile media, such as smartphones and tablets.

Bootstrap 3 was released in 2012, and its major development is the integration of the response as a core functionality of the library. It is designed mobile-first with a grid system with 12 columns, making it easier to build a site for mobiles and then adapt it for larger screens. This version is suitable for most developers, and it is used by many websites **today**.

Bootstrap 4 was finalized on January 18, 2018 (Otto & Thornton, 2018). Bootstrap 4 is practically a complete rewrite from Bootstrap 3 and is considered easier to use than previous versions.

What is Bootstrap?

Bootstrap is an open-source CSS framework that makes the construction of websites and applications easier. A framework corresponds to a set of libraries of components, grouped together for a specific purpose and with internal rules that users must follow, that is reusable (as opposed to re-creating every component repeatedly). In other words, Bootstrap is a collection of CSS and JavaScript files containing predefined rules that work together and can be used to create complex designs in a straightforward way. These rules are enclosed in classes, so only the classes of interest are used to apply a set of styles to a particular HTML element.

Bootstrap is overall one of the most popular systems for developing responsive, fast, and reliable websites and applications. Bootstrap has earned its popularity because of its responsiveness and mobile-first orientation.

Bootstrap Response Grid System

Bootstrap is designed with a responsive grid system, as well as supporting classes that form the columns and rows. Bootstrap grids are based on the flexbox, and a grid will always be made up of 12 basic columns, scaling from 1 to 12. Bootstrap provides you with the following predefined prefix classes to define the width of each column and aim the column for a specific viewport size: `col-`, `col-sm-`, `col-md-`, `col-lg-`, and `col-xl-`. The following table resumes more detail on the prefixes (Firdaus et al., 2016).

Table 4: Five classes grid system in Bootstrap on multiple devices

Prefix	Description
col-	extra small devices—screen width less than 576px
col-sm-	small devices—screen width equal to or greater than 576px
col-md-	medium devices—screen width equal to or greater than 768px
col-lg-	large devices—screen width equal to or greater than 992px
col-xl-	xlarge devices—screen width equal to or greater than 1200px

Source: Hadid (2021).

To define a new grid line, we use the `.row` class. For each new row, we decide the number of columns the row will have out of a possible 12, thanks to `.col-*` classes. Here it is important to understand that Bootstrap grids automatically have 12 base columns, which are implicitly defined. We then can “group” some of these 12 basic columns together in each row to form new customized columns.

The classes above can be combined to create more dynamic and flexible layouts. Each class scales up, so if you wish to set the same widths for `sm` and `md`, you only need to specify `sm`.

Basic structure of a Bootstrap grid

In the following example, we create a row by using `<div class="row">`. Then, add the number of columns you want by using the tags with suitable `col-**-*` classes (in our example we created 3 columns). The first star `*` represents the responsiveness (`sm`, `md`, `lg` or `xl`), while the second star represents a number between 1 and 12.

Code

```
<!-- Control the column width and how they should appear on different devices -->
<div class="row">
  <div class="col-**-*"></div>
  <div class="col-**-*"></div>
  <div class="col-**-*"></div>
</div>
```

In the next example, instead of adding a number to each `col`, let bootstrap manage the layout. Thus, if a row has three elements to which we have passed a `.col` class, each of the three elements will automatically occupy a space with a width equivalent to $12/3 = 4$ base columns of the row.

Code

```
<! -- Or let Bootstrap automatically handle the layout -->
<div class="row">
  <div class="col"></div>
  <div class="col"></div>
  <div class="col"></div>
</div>
```

Creating a grid with columns of the same width

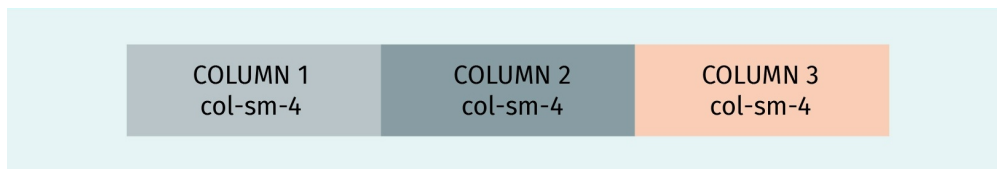
We start with a simple example consisting of a grid with one row and three columns of the same width, using `col-sm-4` class.

Code

```
<div class="row">
  <div class="col-sm-4"> col-sm-4</div>
  <div class="col-sm-4"> col-sm-4</div>
  <div class="col-sm-4"> col-sm-4</div>
</div>
```

Thus, each column has the same size and is reduced to the small size viewport defined by Bootstrap (≥ 576 px). In the following, we show how the previous code appears in the browser (for clarity, we added coloring):

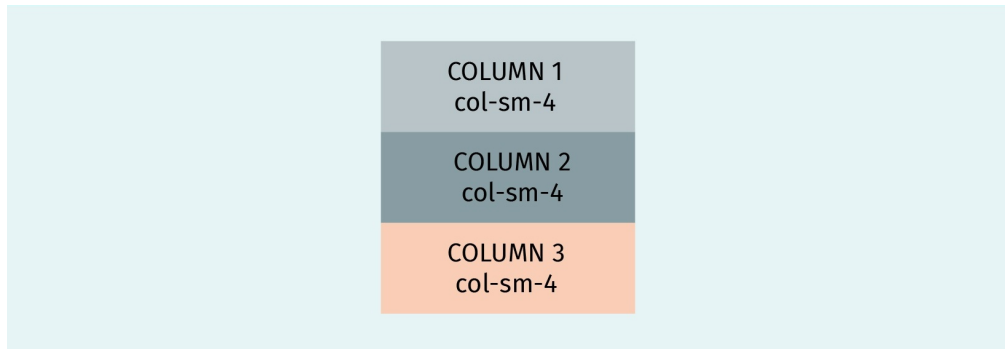
Figure 61: Grid with Three Columns of the Same Width with Viewport Size = 576 Pixels



Source: Hadid (2021).

If we view the previous code in the viewport size smaller than 576 px, all these columns will start stacking up: The first column stacks upon the second, which stacks upon the third, as shown in a shot of next screen:

Figure 62: Grid with Three Columns of the Same Width with Viewport Size < 576 Pixels



Source: Hadid (2021).

Creating a grid with columns with multiple viewport width sizes

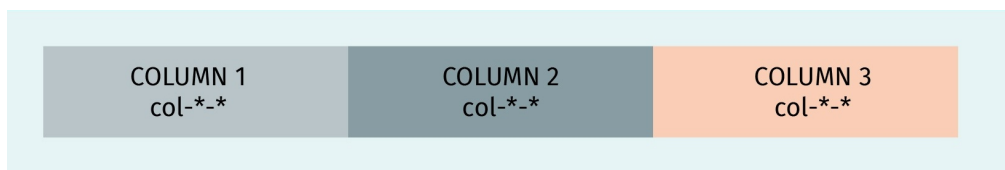
Moreover, we can set a specific column width based on the screen size, and we can use multiple classes within multiple viewport sizes, as follows:

Code

```
<div class="row">
  <div class="col-sm-6 col-md-2 col-lg-4"></div>
  <div class="col-sm-3 col-md-4 col-lg-4"></div>
  <div class="col-sm-3 col-md-6 col-lg-4"></div>
</div>
```

Considering the previous example, the columns will have the same size within the large viewport size $\geq 1,200$ px, as shown in the following screenshot:

Figure 63: Grid with Three Columns of the Same Width with Viewport Size = 1,200 Pixels



Source: Hadid (2021).

The proportion of the column starts to change when we visualize it in the medium viewport size according to the assigned classes on each column. The width of the first column will become smaller, the second column will keep the same proportion, and the third column will become larger, as the following screenshot shows:

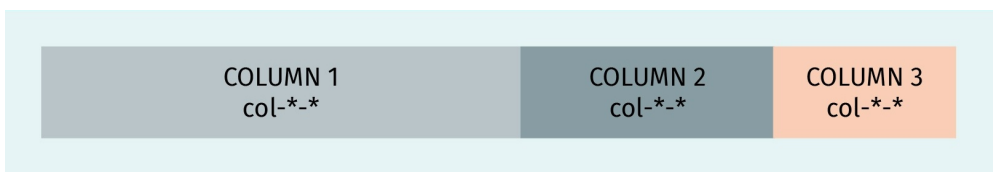
Figure 64: Grid with Three Columns of the Same Width with Viewport Size < 1,200 Pixels



Source: Hadid (2021).

The proportion of the column will change again when the web page is at the threshold of the medium and small viewport sizes defined by Bootstrap, which is around 991 pixels, as shown in the following screenshot:

Figure 65: Grid with Three Columns of the Same Width with Viewport Size < 991 Pixels



Source: Hadid (2021).

Buttons

Another important component we will integrate into the web page are buttons. To customize the style of our buttons with Bootstrap, we use the base class `.btn` and classes of type `.btn-*`, as shown in the following code:

Code

```
<button type="button" class="btn btn-*">Primary</button>
```

Apply a background color to a button

Bootstrap includes several styles of predefined buttons, each serving its own semantic purpose. The default appearance is gray with white text. To customize the color of our buttons with Bootstrap, replace the star (*) of `btn-*` class with `primary`, `secondary`, ..., `light`, or `link` to give the buttons the colors specified, as shown in the following code:

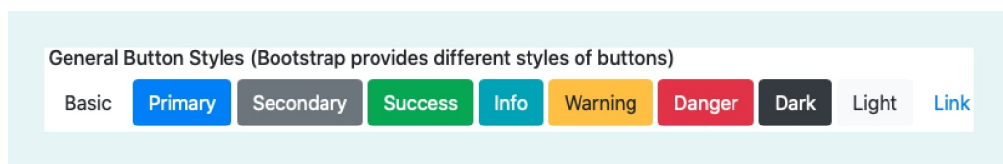
Code

```
<button type="button" class="btn">Basic</button>  
<button type="button" class="btn btn-primary">Primary</button>  
<button type="button" class="btn btn-secondary">Secondary</button>  
<button type="button" class="btn btn-success">Success</button>  
<button type="button" class="btn btn-info">Info</button>  
<button type="button" class="btn btn-warning">Warning</button>  
<button type="button" class="btn btn-danger">Danger</button>
```

```
<button type="button" class="btn btn-dark">Dark</button>  
<button type="button" class="btn btn-light">Light</button>  
<button type="button" class="btn btn-link">Link</button>
```

In the following, the button changes appearance when the previous classes are applied:

Figure 66: The Different Buttons Styles in Bootstrap



Source: Hadid (2021).

Contour button

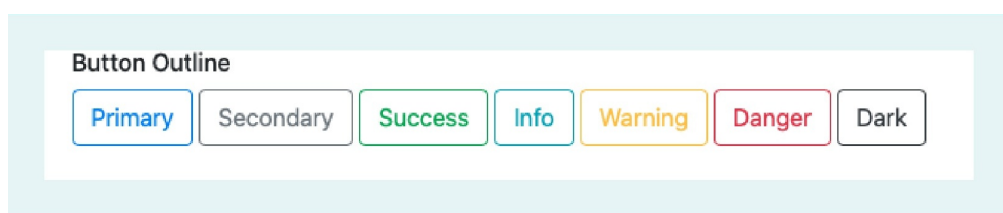
You can remove the background color on any button of any predefined style by replacing the modified default classes with `.btn-outline-*` styles, as shown in the following code:

Code

```
<button type="button" class="btn btn-outline-primary">Primary</button>  
<button type="button" class="btn btn-outline-secondary">Secondary</button>  
<button type="button" class="btn btn-outline-success">Success</button>  
<button type="button" class="btn btn-outline-info">Info</button>  
<button type="button" class="btn btn-outline-warning">Warning</button>  
<button type="button" class="btn btn-outline-danger">Danger</button>  
<button type="button" class="btn btn-outline-dark">Dark</button>
```

As explained previously, Bootstrap provides classes to apply the semantic color to the contour of buttons only; to apply it, use the class `.btn-outline-*`. In the following, we show how the button contours change appearance when the previous classes are applied.

Figure 67: Manage Button Contours in Bootstrap



Source: Hadid (2021).

Button size

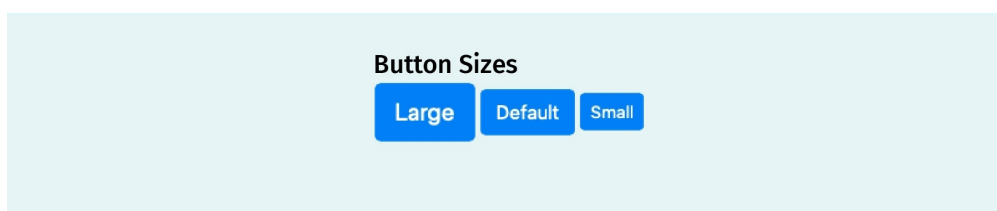
We can adjust the size of the button as shown in the code. Buttons may have small, normal, and big sizes, use `btn-sm` and `btn-lg` classes to make that happen:

Code

```
<button type="button" class="btn btn-primary btn-lg">Large</button>  
<button type="button" class="btn btn-primary">Default</button>  
<button type="button" class="btn btn-primary btn-sm">Small</button>
```

The following shows how the button sizes change when the previous classes are applied:

Figure 68: The Different Button Sizes in Bootstrap



Source: Hadid (2021).

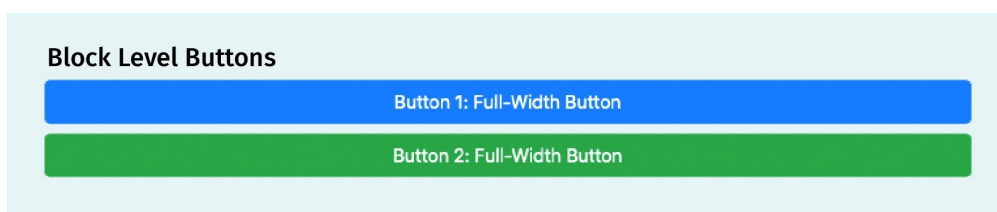
If you want to create block level buttons that span the full width, just add `btn-block` class, as shown in the following code:

Code

```
<button type="button" class="btn btn-primary btn-block">Button 1: Full-Width Button</button>  
<button type="button" class="btn btn-success btn-block">Button 2: Full-Width Button</button>
```

The following shows the button span changes when the previous class is applied:

Figure 69: Buttons Spanning Full Width in Bootstrap



Source: Hadid (2021).

The aim of this section is to introduce some of the important components of Bootstrap. However, for more learning or support on building other components such as navigation menus, form, inputs, carousel, and flexbox, head over to the official Bootstrap site.



SUMMARY

Responsive websites allow the best possible design for different visual output devices with different screen sizes. In contrast to a fixed or adaptive layout, responsive websites offer optimal adaptation to different display formats and sizes. An adaptive layout defines breakpoints at which layout properties can be changed significantly using CSS; however, within the individual breakpoints, the adaptive layout corresponds to a fixed layout.

With a responsive layout, breakpoints can also be defined, but the behavior between breakpoints is a fluid, adaptive layout. Responsive websites rely on adaptive content such as text, images, and icons.

The challenge in responsive website design lies in designing adaptable content. For example, different image sections for window-width key visuals can be defined for different screen sizes. Furthermore, a mobile-first design approach should be used. In responsive web design, different layout patterns can be used.

In this section, we started by discussing the history of Bootstrap. We underlined the advantages of this framework to give an idea of the cases where it will be interesting to use and those where it will be necessary to favor other solutions. Lastly, we presented the new features introduced in Bootstrap 4, the version with which we will be working and setting up our working environment.

UNIT 5

WEB PAGE DEVELOPMENT WITH JAVASCRIPT

STUDY GOALS

On completion of this unit, you will be able to ...

- identify the differences between client-side programming and server-side programming.
- describe the role that JavaScript and DOM play in client-side programming.
- send and receive data across platforms using JSON.

5. WEB PAGE DEVELOPMENT WITH JAVASCRIPT

Introduction

It often happens that a web page changes while a user is visiting it. This can happen, for example, when viewing a web page with a news ticker that updates with each new visit. It can also happen that we want to reload data from a server without reloading the entire web page, thereby, e.g., saving traffic that would be needed for the repeated transfer. It is also not always expedient to occupy server resources with small computations or any kind of server-side data manipulation. The execution of code and computations on a server is called server-side programming. Database inquiries fall under this classification.

The alternative is client-side programming, where the goal is to perform smaller calculations locally in the user's browser or make small design changes locally and independently, without requesting new data from a server. In the context of this unit, we will examine some important aspects of client-side programming. One disadvantage to client-side programming is that its range of function is somewhat limited, since databases, which may need to be queried, are not locally present and require communication with a server (Walter, 2008).

5.1 JavaScript History, ES5/ES6, and Typescript

JavaScript (JS) is a comprehensive, dynamic programming language able to provide dynamic interactivity (e.g., button click responses or data entered in forms, dynamic composition) on websites. JavaScript can implement complex mechanisms on a web page, such as games, interactive maps, 2D/3D animations, and scrolling video menus. Along with HTML and CSS technologies, JavaScript is one of the core technologies of the World Wide Web. Nearly every major website, including Google, Facebook, and Netflix, makes heavy use of JavaScript (Shute, 2019).

In 1995, the foundation for what would eventually become JavaScript was written by Brendan Eich, a programmer at Netscape. The first version of the language was written in 10 days and known internally as "Mocha" (later renamed "LiveScript" before finally becoming JavaScript) (Dubois & Georges, 2019). A language inspired by many languages (especially Java), it simplified syntax for beginners. JavaScript is a scripting programming language primarily used in interactive web pages for client-side rendering, although in recent years its use on the server-side has grown (with Node.js for example).

By late 1996, Sun and Netscape began working with the European Computer Manufacturers' Association (ECMA) to develop a standard for future versions of JavaScript (Timms, 2016). In mid-1997, the first standardized version of JavaScript, ECMAScript (EC), was released. It has many features currently used in JS, such as functions, objects, and prototypical inheritance class.

Fortunately, efforts to standardize JavaScript continued behind the scenes. Versions 2 and 3 of ECMAScript (ES2 and ES3) were released in 1998 and 1999, respectively (Timms, 2016, p. 4). ES3 was released with exception handling.

Work began in early 2000 on ECMAScript 4, which was to be a major new release. In 2000, many features were offered by ECMAScript 4, including classes, interfaces, optional types, and other mechanisms intended to meet the needs of large companies (Timms, 2016, p. 4). However, the standard did not attract many people to the community. The committee therefore decided to develop ECMAScript 3.1 (simple version) and ECMAScript 4 (enterprise version) in parallel. Unfortunately, this approach failed, and ES4 was never released.

For the next five years, JavaScript was unpopular and suffered from several problems. Incompatibility between different browsers on the market was one of the major problems with JavaScript at the time. Years passed without a new release of ECMAScript. However, the evolution of ECMAScript frameworks continued, and libraries such as jQuery, Prototype, Dojo, and Mootools concealed the main differences between browsers, making it easier to develop across browsers (Timms, 2016, p. 4). Meanwhile, the number of apps being developed using JavaScript increased significantly.

In 2006, JavaScript received a boost with the release of John Resig's jQuery library (York, 2009).

Node.JS was presented in 2009 by Ryan Dahl, giving rise to the JavaScript Everywhere paradigm (Training.com, 2016). The language gradually regained the interest of developers, and at the end 2009 a new version of JavaScript, ES5, was released. ES5 is the version most familiar to developers around the world. It includes strict mode, accessors, syntax changes, meta-programming, and, most importantly, support for JSON (Dubois & Georges, 2019).

Since 2010, new JavaScript frameworks and libraries have multiplied, including AngularJS (Google), Backbone.js (Jeremy Ashkenas), and ReactJS (Facebook) (Cardoso, 2020). Dozens of other frontends, backends, and full Stack frameworks have appeared, including Angular, Ember, Meteor, Sails, Vue.JS, Svelte, Mithril, Knockout, and Polymer (Cardoso, 2020). Several new features came with the release of ECMAScript 6 in 2015, changing the way JS developers write their code.

The specification now has a yearly release schedule, meaning a new version is released each year. The development of the language has continued to accelerate; ES7, ES8, ES9, ES10, and ES11 were released in 2016, 2017, 2018, 2019, and 2020, respectively (Dubois & Georges, 2019).

5.2 JavaScript Fundamentals

JavaScript (selfHTML, n.d.-d) is a client-based programming language. It is one of the most widely used programming languages worldwide and one of the most common programming languages of web pages (W3 Schools, n.d.). Although JavaScript is a programming language, it is often referred to as a scripting language. JavaScript can be understood as a means of modifying content formulated in HTML based on calculations performed locally on the user's device.

JavaScript can be included directly in an HTML document by opening a corresponding `<script>` element. A special feature of JavaScript is that it is executed in the user's browser and does not have to send any data to a server for calculation. JavaScript has the particular advantage of being able to manipulate HTML code, perform calculations, and, if necessary, influence the design of an HTML page. Using selectors, it is possible to specify where the output of the calculation should go. Let's look at an example in which JavaScript is used to output the text "Web Development Basics" in a `<p>` element.

Figure 70: JavaScript in HTML

```
<!DOCTYPE html>
<html>
<body>

<h2>IU</h2>
<p id="ptag"></p>
<script>
document.getElementById("ptag").innerHTML = "Web Development Basics";
</script>

</body>
</html>
```

Source: Sturm (2020).

Using `document.getElementById("ptag")`, we assign the text to be output in the `p`-tag with `id=ptag`. This then leads to the following output.

Figure 71: JavaScript in HTML Output

IU
Web Development Basics

Source: Sturm (2020).

Another advantage of JavaScript is that it can perform calculations. Let's assume that we want to add the variable `b = 1` to the value of the variable `a = 2`. This can be achieved as follows:

Figure 72: JavaScript Calculations

```
<!DOCTYPE html>
<html>
<body>

<h2>IU</h2>
<p id="ptag"></p>
<script>
var b=1;
var a=2;
var c=a+b;
document.getElementById("ptag").innerHTML = "Web Development Basics:" + c;
</script>

</body>
</html>
```

Source: Sturm (2020).

This results in the following output on the user's screen.

Figure 73: JavaScript Calculations Output

IU

Web Development Basics: 3

Source: Sturm (2020).

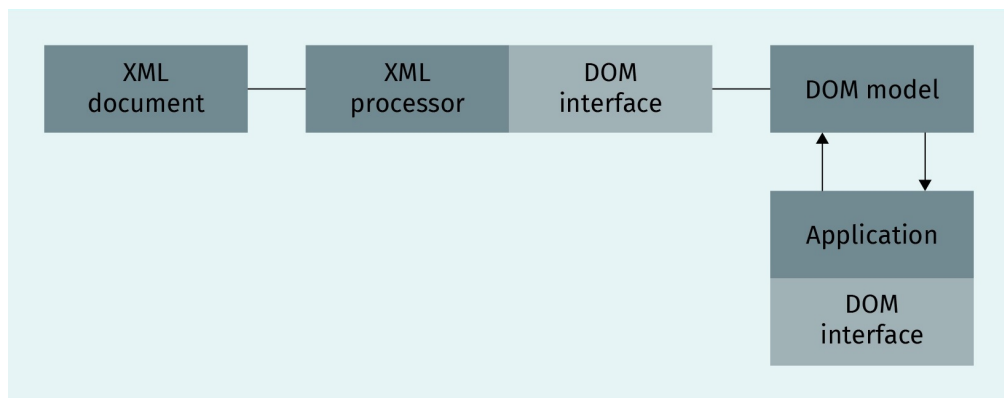
These may be very simple examples, but JavaScript is a completely self-contained scripting language (comparable to almost any other prominent language) that allows us to pass data across platforms, dynamically change the appearance of a web page, display graphs, and much more.

DOM

In addition to JavaScript, we can also make use of the document object model (DOM), which can influence the content of HTML pages on the client side. It does so by means of XML.

The concept behind DOM is to create a uniform programming interface using XML in order to exchange data in an automated and standardized way. An XML processor prepares an arbitrary XML document over generally recognized interfaces as a tree, whose elements can then be accessed via different markup or programming languages. The typical structure of a DOM document is shown in the following figure.

Figure 74: XML Document with DOM Interface

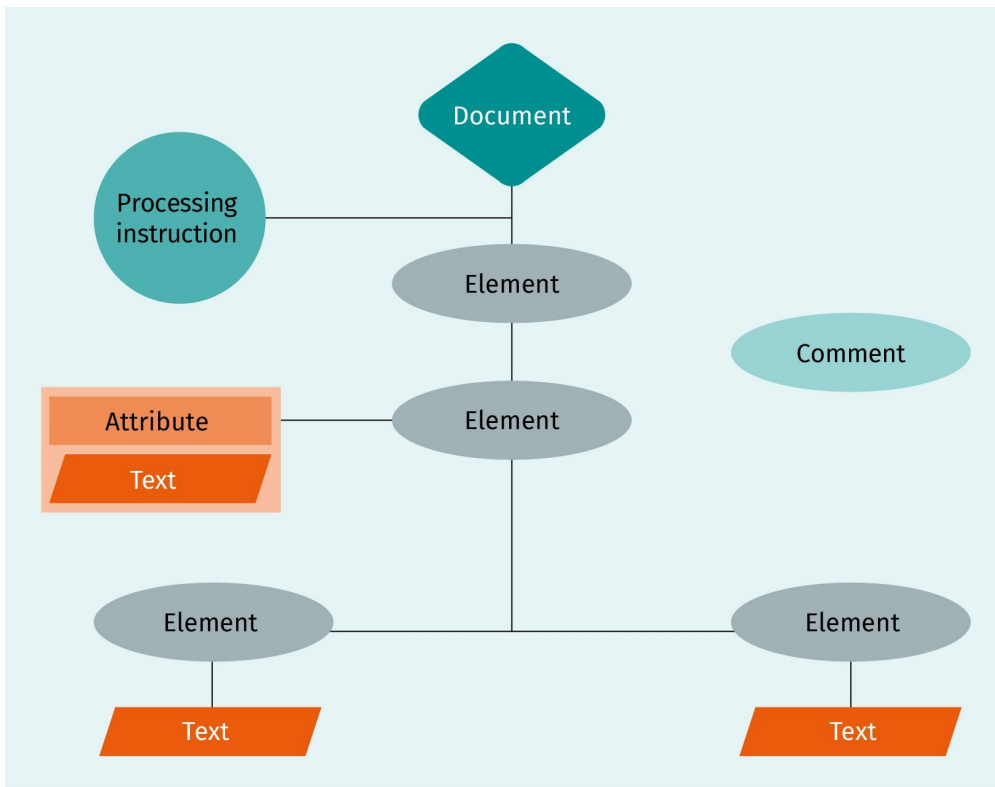


Source: Sturm (2020).

Class
A class is understood here as a summary of various functions that can be used to manipulate an XML document.

The basic idea is that the **class** of elements in an XML document provides functions of the type `setAttribute()` or `getAttribute()`, with which the values of an element of an XML document can be manipulated or read. The fundamental building block is the node interface as part of DOM, which is used to systematically traverse the nodes of a tree. Based on this warehouse example, the corresponding DOM tree, which can be generated by an XML parser, can be represented as shown in the following figure.

Figure 75: Illustration of the Structure of a DOM Tree



Source: Sturm (2020), based on Vonhoegen (2018), p. 414.

Let us now try to read out the XML document using MSXML. MSXML is a tool developed by Microsoft for handling XML documents, which has implemented the DOM interface (see Microsoft, n.d.). We use an approach based on JavaScript for this purpose.

The fundamental basis here is to first locate the root node of the DOM tree. This is done by the command (Vonhoegen, 2018):

```
stockNode = xmlDoc.documentElement;
```

In the next step we want to get a list of all child elements, i.e., all items in the warehouse, which we can carry out as follows:

```
nodeList = lagerNode.childNodes;
```

The return value here is a list whose elements we can access by means of an index “i” as follows:

```
itemNode= nodeList(i)
```

We read the respective attributes of the products as follows.

Code

```
articleNode = articleNode.firstChild;
DesignationNode = ItemNode.nextSibling;
ColorNode = LabelNode.nextSibling;
PriceNode = ColorNode.nextSibling;
StockNode = PriceNode.nextSibling;
```

For example, if we want to check all items in the warehouse for their item number or if we want to search for a specific item number, we can achieve this as follows.

Figure 76: XML Readout

```
for (var = 0; i < nodeList.length; i++)
{
  if (stockNode.childNodes(i).firstChild.firstChild.nodeValue == Itemnumber)
  }
  found = i;
```

Source: Sturm (2020).

If we want to search our warehouse for an item and, if it is located there, output all other parameters characterizing it, we achieve this as follows (Vonhoegen, 2018).

Figure 77: XML Search

```
function SearchArticles(Itemnumber)
{
  var xmlDoc, StockNode, ItemnumberNode, DesignationNode;
  var ColorNode, PriceNode, StockNodes;
  var found, NodeList;
  xmlDoc = new ActiveXObject("Msxml2.DOMDocument.6.0");
  xmlDoc.load("stockdata.xml");
  StockNode = xmlDoc.documentElement;
  NodeList = StockNode.childNodes;
  for (var = 0; i < NodeList.length; i++)
  {
    if (StockNode.childNodes(i).firstChild.firstChild.nodeValue ==
    Itemnumber.)
  }
  found = i;
  ItemNode = StockNode.childNodes(found);
  ItemnumberNode = ItemNode.firstChild;
  DesignationNode = ItemnumberNode.nextSibling;
  ColorNode = DesignationNode.nextSibling;
  PriceNode = ColorNode.nextSibling;
  InventoryNode = PriceNode.nextSibling;
}
```

Source: Sturm (2020).

5.3 Use of JSON

An alternative for the platform-independent handling of data (storage and forwarding), which has also become very popular alongside XML, is JavaScript object notation (JSON) (Vonhoegen, 2018). A corresponding JSON connection exists for almost all known programming languages, so it is possible to exchange data even across different languages (concerning server and client). It is important that JSON (like XML) concerns pure text exchange. JSON shows a dictionary-like syntax, where the name of the respective attribute is given in quotation marks first, followed by a colon, before the actual stored value is given, which is also noted in quotation marks. The basic syntax is based on the notation form of JavaScript (Vonhoegen, 2018). An important advantage over XML is the more compact representation option, which enables a significant reduction of the data size when exchanging data. Furthermore, XML is based on a more predefined structure than is required for JSON. As already mentioned with XML, JSON can also be used to serialize objects in order to transfer or store and reload them.

As an example, let's consider the description of a person with name, age, and place of residence. In JSON, this information can be modeled as follows:

```
var person = {"name": "Anna", "age": "30", "city": "Stockholm"}
```

The name, age, and place of residence keys can then be used to extract the values of the person variable stored under this key.

With the Java command

```
person.name = "Anna";
```

the stored value of the variable person can be retrieved with the key name. Assignments can be made using the same syntax. Finally, as an example, let's look at how we can model the previously created XML document for warehouse management as a JSON object:

Code

```
var warehouse = {"warehouse": [
{"Itemnumber": "12345", "Designation": "Product 1", "Color": "Blue",
"Price": "25.90", "Stock": "22"}
{"Itemnumber": "65432", "Designation": "Product 2", "Color": "Black",
"Price": "9.99", "Stock": "34"}
]}
```

JSON is capable of handling a wide variety of data types, such as arrays (each formulated in square brackets [...]), objects ({...}), strings (in "..."), numbers, null values, and Boolean values.

5.4 Common JavaScript Frameworks

As we have seen, a range of different JavaScript frameworks has been developed in recent years. In this section, we highlight one framework in more detail: Vue.js.

Vue.js is one of the last JavaScript frameworks to emerge. After the long reign of jQuery, new JavaScript libraries have appeared in an attempt to remedy the shortcomings of their predecessor. Among those currently in vogue are React.js, Angular, and Vue.js.

By introducing the concept of responsiveness and component-based application development, these libraries have revolutionized the way to develop a web application with JavaScript. Among these new libraries, Vue.js is the simplest approach.

Vue was created by Evan You when he was working at Google on AngularJS. Vue is a powerful, simple, and easy-to-use JavaScript framework whose first prototype appeared in 2013 (Firdaus et al., 2016). The first public release in early 2014 met with immediate success from the very first day (Chau, 2017). Vue is the successful result of combining some of the excellent ideas of Angular and React.js and putting them in an easy-to-use package (Firdaus et al., 2016). Designers deliberately erased imperfections encountered in React.js and Angular in order to create something simpler and better. Compared with other popular frontend frameworks, Vue comes out on top for simplicity and ease of use (Firdaus et al., 2016). For developers working in JavaScript, Vue has become a popular and well-established framework. Vue.js is currently available in version 3.x, but the older version 2.x is still widely used. This leads to some differences in the syntax and functionality. In this course we will take version 2.x into account. As a result, the following examples will use the syntax of version 2.x of Vue.js.

Getting Started with Vue.js

While Vue can be used as a JavaScript module in more sophisticated setups, it can also simply be included as an external script in the body of your HTML document.

A very basic example is a simple HTML file with a “Greeting from Vue!” message, created using the VS code editor.

Figure 78: Simple HTML File Showing a Greeting Message

```
1 <!DOCTYPE html>
2 <html lang= "en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8 </head>
9 <body>
10    <h1>Greetings from Vue!</h1>
11 </body>
12 </html>
```

Source: Hadid (2021).

Now we will carry on and do the same job using Vue.js. First, we will include Vue and create a new instance, as follows:

Figure 79: Simple Vue.js Code file Showing a Greeting Message

```
8 </head>
9 <body>
10    <div id="app">
11        <h1>Greetings from Vue!</h1>
12    </div>
13 </body>
14 <script src="https://unpkg.com/vue"></script>
15 <script>
16     new Vue({
17         el: '#app',
18     })
19 </script>
20 </html>
```

Source: Hadid (2021).

The main HTML tags and structure should be familiar to you. Let us go over some of the other aspects.

Vue framework

We use the simplest and most portable way of embedding Vue into our HTML quickly and easily without having to install the framework. We load a hosted version of Vue library from a content delivery network (CDN), unpkg. Unpkg has a fast and independent site that provides the most popular libraries on NPM. The following link will always point to the latest version of Vue on npm in our application:

Code

```
<script src="https://unpkg.com/vue"></script>
```

The embedded link will be updated with the release of a new version of Vue.js. As this course book may not be updated as frequently as new versions of Vue.js become available, please take into account the most current library of Vue.js from the CDN.

Application space (or view)

The Vue code will be placed within the HTML tags `main`, `section`, and so on (except `<body>` and `<html>` tags). Throughout this section, we use a `div` tag with an `id` of `#app`, the element to which we are referring. The `#app` element with the `id` will be referred to as the app space or view (Street, 2017). Try to see this as a container that Vue works at and all HTML and tags and code for your application should be placed within this container.

Vue initialization and first message display

Each Vue application is initialized by creating a new Vue instance with the Vue function

Code

```
new Vue ({
  el : '#app'
  // options
})
```

The property `el` is used to link the Vue application to the HTML app space. The `id` property used in the HTML tag informs Vue which tag element should be linked with the Vue application (i.e., where the Vue application will be contained).

Code

```
<div id="app">
</div>
```

Note that the `el` property connects the Vue model to any element in our view using the element's `id`. In this case, we bind our Vue model to an element whose `id` attribute value is `app`.

DOM

DOM is an API for HTML and XML that allows us to build documents, navigate their structure, and add, modify, or delete elements and their contents.

The property `el` can be written by indicating the corresponding **DOM** (Document Object Model) element, as follows:

Figure 80: Using a DOM Element to Specify the “el” Property

```
9 <body>
10   <div id="app">
11     <h1>Greetings from Vue!</h1>
12   </div>
13 </body>
14 <script src="https://unpkg.com/vue"></script>
15 <script>
16   new Vue({
17     el: document.getElementById('#app'),
18   });
19 </script>
```

Source: Hadid (2021).

In the sequel, we create a small application to display a message (Kyriakidis & Maniatis, 2016). One way to accomplish this is to assign the message we want to display to a variable object named `data` (see line 16). Then, the `data` object is passed as parameter to the `Vue` object (line 19). To show our message on the page, we twist the message in the mustache-style curly braces `{{}}` (line 11). Anything that is inside our message will be displayed automatically in the `<h1>` tag.

Figure 81: Using Vue Instance to Show a Message

```
9 <body>
10   <div id="app">
11     <h1>{{ message }}</h1>
12   </div>
13 </body>
14 <script src="https://unpkg.com/vue"></script>
15 <script>
16   var data = { message: 'Greetings from Vue!' };
17   new Vue({
18     el: '#app',
19     data: data
20   })
21 </script>
```

Source: Hadid (2021).

Another way to define the message variable is to do it directly inside the `Vue` constructor in the `data` object:

Figure 82: Defining the Message Variable in Vue Instance

```
15 <script>
16     new Vue({
17         el: '#app',
18         data: { message: 'Greetings from Vue!' }
19     })
20 </script>
```

Source: Hadid (2021).

Opening in the browser

To view the file, locate it in your web browser. In Chrome, it is as simple as File|Open File. When it loads, the page should show a “Greeting from Vue!” message.

Directives

We can add functionality to our program using directives. The concept of directives is interesting, as they make code easier to understand and work with. These are special properties that we add to HTML tags using the `v-` prefix, e.g., `v-bind`, `v-cloak`, `v-for`, `v-else`, `v-else-if`, `v-model`, `v-on`, `v-once`, `v-text`, and `v-html` (Imsirovic, 2018).

Two-way binding (v-model)

Suppose we want to change the message on user input, so we use `v-model`, a directive of Vue. We use two-way data binding to dynamically change the message value when the user changes the message text inside an input, as shown here.

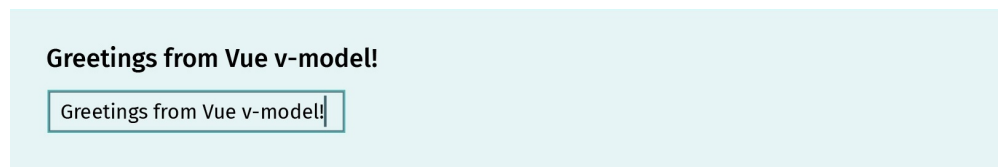
Figure 83: v-model Example

```
9 <body>
10     <div id="app">
11         <h1>{{ message }}</h1>
12         <input v-model="message">
13     </div>
14 </body>
15 <script src="https://unpkg.com/vue"></script>
16 <script>
17     new Vue({
18         el: "#app",
19         data: { message: "<h1>Greetings from Vue v-model!</h1>" },
20     });
21 </script>
```

Source: Hadid (2021).

The moment you change the input field, the value of the message will change alongside, as shown.

Figure 84: Data Binding Using v-model



Source: Hadid (2021).

HTML declarations

The `v-html` directive allows you to output content without using the mustache-style curly bracket syntax.

In your view app space, add the `v-html` attribute to an element.

Figure 85: Adding v-html to a div Element

```
9 <body>
10   <div id="app">
11     <div v-html="message"></div>
12   </div>
13 </body>
```

Source: Hadid (2021).

In the JavaScript, set the `message` variable to a string containing some HTML tags, as shown.

Figure 86: Setting the Message Variable to a HTML String

```
15 <script>
16   new Vue({
17     el: "#app",
18     data: { message: "<h1> Greetings from Vue v-html!</h1>" },
19   });
20 </script>
```

Source: Hadid (2021).

V-show

To explain the `v-show` directive, we will present a simple example to facilitate understanding. Suppose you need to toggle the display of an item, according to a set of criteria. For example, a submit button should not appear unless you have first typed in a message (Kyriakidis & Maniatis, 2016).

Here we have the Vue code:

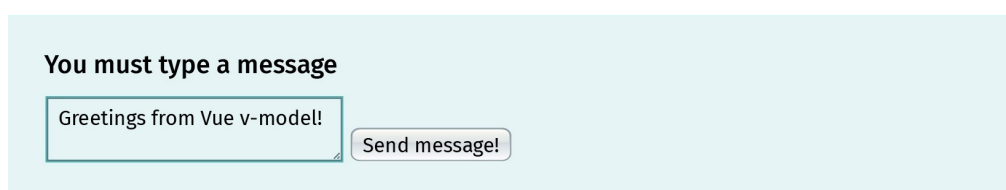
Figure 87: Using v-show

```
9 <body>
10   <div id="app">
11     <h1>You must type a message</h1>
12     <textarea v-model="message"></textarea>
13     <button v-show="message">Send message!</button>
14     <pre>
15       {{{data | json}}}
16     </pre>
17   </div>
18 </body>
19 <script src="https://unpkg.com/vue"></script>
20 <scrip>
21   new Vue({
22     el: "#app",
23     data: { message: "Greetings from Vue v-model!" },
24   });
25 </script>
```

Source: Hadid (2021).

Inside the `textarea` tag, we are going to display our message. We bind the value of `textarea` with our `message` variable using `v-model` so it displays our message. Next to that, a button will be displayed conditionally. Here, the button appears since there is a message present in the `textarea`, as shown.

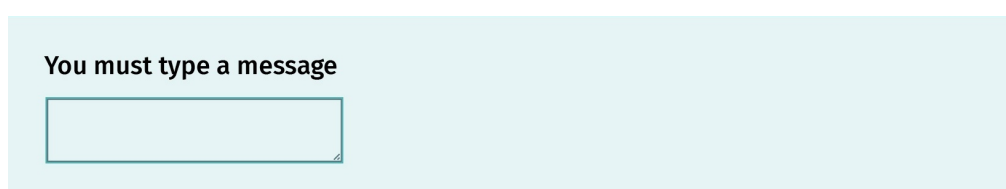
Figure 88: Using v-show: "Send message!" Button



Source: Hadid (2021).

The button disappears after removing the message in `textarea` (i.e., `textarea` is empty), as shown.

Figure 89: Using v-show: "Send message!" Button No Longer Appears



Source: Hadid (2021).

Conditional rendering

These include `v-if`, to show a container whether a statement equals true, and `v-else`, to show an alternative.

The `v-if` directive is used to toggle the display CSS property of an element with a condition. If the condition is true, it will make it visible; otherwise, it will be invisible.

Figure 90: Using `v-if`

```
 9 <body>
10   <div id="app">
11     <h2 v-if="isVisible"> Greetings from Vue! Now you see me </h2>
12   </div>
13 </body>
14 <script src="https://unpkg.com/vue"></script>
15 <script>
16   new Vue({
17     el: "#app",
18     data: {isVisible: true}
19   })
20 </script>
```

Source: Hadid (2021).

Right now, your Vue app would be displaying the contents of your element.

`v-if` does not only work with Boolean true/false values. You can also check whether a data property is equal to a specific string:

```
<div v-if="isVisible == 'yes'"> Greeting from Vue </div>
```

`v-else` allows you to make an alternative element based on the opposite of the `v-if` statement. If the result is true, the first element is displayed; otherwise, the element containing `v-else` is shown.

Figure 91: Using v-else

```
9 <body>
10   <div id="app">
11     <h2 v-if="isVisible"> Greetings from Vue! Now you see me </h2>
12     <h2 v-else> Greetings from Vue! Now you don't</h2>
13   </div>
14 </body>
15 <script src="https://unpkg.com/vue"></script>
16 <script>
17   new Vue({
18     el: "#app",
19     data: {isVisible: false}
20   })
21 </script>
```

Source: Hadid (2021).

Now, your Vue app would be displaying the contents of your second element.

You can also use `v-else-if` should you wish to chain your conditions.

Figure 92: Using v-else-if

```
9 <body>
10   <div id="app">
11     <h2 v-if="isVisible"> Greetings from Vue! Now you see me </h2>
12     <h2 v-else-if="otherVisible"> Greetings from Vue! Now you
13       might not see me</h2>
14     <h2 v-else> Greetings from Vue! Now you don't</h2>
15   </div>
16 </body>
17 <script src="https://unpkg.com/vue"></script>
18 <script>
19   new Vue({
20     el: "#app",
21     data: {isVisible: false,
22           otherVisible: false}
23   })
24 </script>
```

Source: Hadid (2021).

The second message is displayed if the `isVisible` variable is false, but the `otherVisible` is true. In this example, the third message will be displayed, since both `isVisible` and `otherVisible` are false.

Looping using v-for directive

The directive used here will be `v-for`. As the name suggests, it performs a for loop on the contents of an array or on the properties of an object. It repeats the HTML element on which it is positioned. On an array, the for loop will be executed as many times as the array elements. On an object, the for loop will be executed as many times as the object has properties.

Figure 93: Example: Using v-for on the Elements List

```
9 <body>
10   <div id="app">
11     <ul>
12       <li v-for="e in elements">{{e}}</li>
13     </ul>
14   </div>
15 </body>
16 <script src="https://unpkg.com/vue"></script>
17 <script>
18   new Vue({
19     el: "#app",
20     data: { elements : ["Element1", "Element2",
21                       "Element3", "Element4",
22                       "Element5"] },
23   });
24 </script>
```

Source: Hadid (2021).

The v-for directive is written into the tag. It is the item that will be repeated each time, including a single element from the list elements in each tag block, as shown

Figure 94: Example Output

- Element1
- Element2
- Element3
- Element4
- Element5

Source: Hadid (2021).

Handle events using v-on directive

The v-on directive is used to handle events with Vue. It is used by indicating the name of the event to be processed, followed by the function to be executed.

Let us use this directive to manage the clicks on the two buttons + and - increasing/decreasing an integer value initialized to 0.

Figure 95: Using v-on to Increment and Decrement a Value

```
9 <body>
10   <div id="app">
11     <button v-on:click="incr(2)">+</button>
12     <button v-on:click="decr(2)">-</button>
13     <p>The value is {{ counter }}.</p>
14   </div>
15 </body>
16 <script src="https://unpkg.com/vue"></script>
17 <script>
18   new Vue({
19     el: "#app",
20     data: { counter: 0},
21     methods:
22     {
23       incr(step) { step = step || 1, this.counter += step; },
24       decr(step) { step = step || 1, this.counter -= step; },
25     },
26   });
27 </script>
```

Source: Hadid (2021).

The `incr()` method is invoked when the + button is clicked, and the `decr()` method is invoked when clicking on the – button.

The processing methods `incr()` and `decr()` must be defined in the method section of the view object; otherwise, they are inaccessible when clicking on the buttons. The increment step (step parameter), if not specified during the call, is set to one by default. Just indicate `incr(2)` or `decr(2)` in the `v-on` directive to increment or decrement in steps of two instead of one.

Computed Values

The data object in Vue is used to store and retrieve data. However, sometimes you want to manipulate the data before outputting it in your applications. We can do that using the computed object in Vue. A computed property calculates and returns a value, rather than just storing it.

The computed object can have as many properties as required; however, each property corresponds to a function that returns the value of the corresponding property.

These functions can utilize data already on the Vue instance and return a value (e.g., string, number, array) that can then be used in the view.

Here is an example code that uses a computed property count to calculate the output.

Figure 96: Example of a Computed Property

```
9 <body>
10   <div id="app">
11     <h2>{{count}}</h2>
12   </div>
13 </body>
14 <script src="https://unpkg.com/vue"></script>
15 <script>
16   new Vue({
17     el: "#app",
18     data: {
19       items: [1, 2, 3, 4, 5],
20       sum: 0,
21     },
22     computed: {
23       count: function () {
24         for (let i = 0; i < this.items.length; i++) {
25           this.sum = this.sum + this.items[i];
26         }
27         return "The count is " + this.sum;
28       },
29     },
30   });
31 </script>
```

Source: Hadid (2021).

We have set two variables: the first variable, `items`, is an array set to `[1, 2, 3, 4, 5]`, and the second variable, `sum`, is set to `0`. We create the `count` property, which returns the sum of the element of the array.

Methods

Beyond classic HTML codes, including reactive variables such as `message` or `isVisible`, it is possible to include method calls. For this, we use the `methods` property to define them in the Vue object. A Vue method is a function associated with the Vue instance.

Let us write a `display_header()` method that concatenates the `welcome` and `name` reactive variables defined in the `data` property. Access to these variables will be through `this.welcome` and `this.name` in the methods defined via the `methods` property.

Figure 97: Defining and Using a Method

```
9 <body>
10   <div id="app">
11     <h2>{{display_header()}}!</h2>
12   </div>
13 </body>
14 <script src="https://unpkg.com/vue"></script>
15 <script>
16   new Vue({
17     el: "#app",
18     data: {
19       Welcome: "Greeting from",
20       name: "vue",
21     },
22     methods: {
23       display_header: function () {
24         return this.Welcome + " " + this.name;
25       },
26     },
27   });
28 </script>
```

Source: Hadid (2021).

The `display_header()` method is defined in the `methods` property of the Vue object, and uses the variables defined in the `data` property with a prefix of `this`. This method is then used in the template surrounded by double braces, by calling it directly (here `display_header()`).

If other methods need to be defined, just separate them with commas (as if defining them as a JSON object).

Watchers

A Vue watcher is called when a reactive variable is modified. This allows treatment to be applied if necessary. We provide an example to illustrate the use of a watcher. We have a Vue instance that shows a message and allows you to change it by clicking a button:

Figure 98: Example: Using a Watcher

```
9 <body>
10   <div id="app">
11     <h1>{{ message }}</h1>
12     <button @click="change_Message()">Change My Message!</button>
13   </div>
14 </body>
15 <script src="https://unpkg.com/vue"></script>
16 <script>
17   new Vue({
18     el: "#app",
19     data: {
20       message: "Greetings from Vue!",
21     },
22     methods: {
23       change_Message: function () {
24         this.message = "Greetings from Vue again!";
25       },
26     },
27     watch: {
28       message: function () {
29         console.log(this.message)
30       },
31     },
32   });
33 </script>
```

Source: Hadid (2021).

We define an observation function (watcher property) in the watch section of the Vue object. This function has the same name as the observed reactive variable, i.e., message. The function defined in the watch section will be called only if the corresponding message variable changes, i.e., when the button is clicked. When the message changes we want to do something, like printing the new message in console log.

Components

Vue allows you to break down the code of your application into a set of components, which are then assembled to form the application. The advantage is splitting the application into several independently managed sub-applications. The created components can be reused in other parts of the application, or even in other applications.

How to use components

A Vue component is created using the `Vue.Component(name, options)` method.

- The name parameter is a string corresponding to the name of the component.
- The options parameter is an object describing the options for creating the component. We will see that many are used when creating the Vue object (by `new Vue`).

Now, we will re-write the initial display message example using a simple component, as follows.

Figure 99: Example: Using a Component to Display a Message

```
9 <body>
10   <div id="app">
11     <h2>{{count}}</h2>
12   </div>
13 </body>
14 <script src="https://unpkg.com/vue"></script>
15 <script>
16   new Vue({
17     el: "#app",
18     data: {
19       items: [1, 2, 3, 4, 5],
20       sum: 0,
21     },
22     computed: {
23       count: function () {
24         for (let i = 0; i < this.items.length; i++) {
25           this.sum = this.sum + this.items[i];
26         }
27         return "The count is" + this.sum;
28       },
29     },
30   });
31 </script>
```

Source: Hadid (2021).

In the previous example, we are initializing a Vue root component on #app, and inside that, we use the Vue component `display-message`, which displays our “Greeting from Vue!” message.

The `Vue.component()` method uses almost the same options used when creating the Vue object.

- The `template` property represents the template that will be displayed when using this component.
- The attributes of the component are called the properties of the component. `Props` (short for properties) lists all the properties we can pass down to a component. We put the attributes in its `props` section, as an array of strings. Here, we have only a “message” attribute.
- The `el` property is only used in the root components when creating the Vue object.

The Vue object is created as before by indicating in the `el` property, the location in which we will insert the indicated template. The latter integrates the component by using it in `<display-message> </display-message>`.

A component accepts other properties (Copes, 2018):

- Data, the component local state
- Methods, the component methods
- Computed, the computed properties associated with the component
- Watch, the component watchers



SUMMARY

Client-side programming is an alternative approach to server-side programming. This allows calculations and smaller functionalities of a website or web application to be executed locally on the user's computer or browser without the need to send data from the user to a server (and, if necessary, back to the user). This prevents unnecessary data throughput and increases the response time of a website (usability).

A special task in client-side programming is performed by JavaScript in conjunction with DOM. JavaScript, which can be executed in a user's browser, can be used to embed further functionalities in addition to the pure description of a web page, as is classically done by HTML.

JSON makes it possible to exchange data efficiently across different programming languages and platforms. This data format, which is based on Java, is recognized by a large number of programming languages (both in the web area and the software development area) and is now used in many different ways.

To finish this unit, we presented the Vue framework of JavaScript. We discussed the quickest way to start learning Vue. We presented the mustache templates, Vue's declarative code, and its easy-to-understand syntax. Then, we introduced directives, methods, computed properties, and watchers. Finally, we presented an approach to building component, templates, and props in Vue.

UNIT 6

WEB APPLICATION TESTING AND SECURITY

STUDY GOALS

On completion of this unit, you will be able to ...

- test your web application by identifying performance bottlenecks.
- use constructive and analytical quality management methods to ensure the software quality of your web application.
- protect your web application by using security concepts and principles such as OWASP.

6. WEB APPLICATION TESTING AND SECURITY

Introduction

Web applications are integrated into various fields (e-commerce, education, banking, leisure, medical, etc.). They have become complex and sophisticated programs that increasingly integrate recent technologies from the Web world. These technologies bring new challenges for developers and testers. In addition, the growing importance of web technologies in our lives; the importance of data transiting through it; the demands of the market, competition, and the quality expected in every application and product; and monetary transactions via the internet raise the thorny problem of the reliability and security of these technologies. Applications evolve quickly, and many of them are poorly designed with shortcomings in terms of navigability, functionality, and traceability.

Testing is the execution or evaluation of a system or components of a system by manual or automated means to verify that it meets specifications or to find the differences between the obtained and expected results (Huffman Hayes, 1994).

Testing is very important in web application development and takes a significant part in development. Web testing, or web application testing, is a software practice that ensures quality by testing that the functionality of a given web application is working as intended or as per the requirements. Web application testing allows organizations to simulate traffic spikes, find bugs, explore vulnerabilities, and verify application functionality (Cavero-Baptista, 2021). There are several types: functionality, usability, interface, compatibility, database, performance, and security. According to Pressman and Maxim (2014), the testing process generally includes the following:

- Unit testing is a process aimed at testing each component of the software taken in isolation (subsystem, module, etc.). This test concentrates on each software component, which is a rather simple task (unless the components are heavily connected (coupling)). In the case of abnormal behavior of the component, it is easier to discover, locate, and correct the error. Finally, unit tests introduce the concept of parallelism during the testing process; several components can be tested independently of the others and simultaneously, which saves resources (time, cost, efficiency, etc.).
- Integration testing makes it possible to verify the correct use of the interfaces between components. This type of test verifies the correct use of the interfaces between component and that the entire system performs the specified functions.
- Validation testing occurs at the end of the integration process. It tests the entire system using real data.

The range of web application testing tools is extensive. Each test type (functional, unit, etc.) requires its own tool, and in a rapidly evolving field, it is difficult to find an effective, sustainable, and interoperable.

To maintain the high reliability required of web applications, we must develop effective testing tools to identify problems. There are various tools on the market for testing web applications. Some examples include both commercial and open-source test automation tools, such as SonarQube, SoapUI, LoadRunner, WinRunner, Silk Performer, JMeter, Selenium, and Apache JMeter.

Web application security is a central part of any web-based business. The global nature of the internet exposes the properties of the web to attacks from distinct locations and at varying levels of scale and complexity. The security of web applications specifically addresses the security surrounding websites, web applications, and web services such as API.

When developing applications, developers are not necessarily aware of the security measures required, and security often faces budgetary and operational constraints. In fact, considering security too late could mean additional cost, as well as the risk incurred in case of attack. It is for this reason that it is important to consider security early in development.

Security testing validates web applications' requirements related to security properties of assets that include confidentiality, integrity, availability, authentication, authorization, and nonrepudiation (Committee on National Security Systems, 2010).

There are a multitude of attacks that can compromise a web application. Web attacks take advantage of security holes (or vulnerabilities) in your web applications. Common methods of attack include malware, phishing, man-in-the-middle, denial-of-service, SQL injection, zero-day exploit, DNS tunneling, XSS (cross-site scripting), and remote command execution path traversal (Cisco, 2017).

6.1 Testing of Web Applications

Since a test plan is initially a general concept, it must be standardized so that both customers and developers can meet on a standardized basis. Various consortia exist for this purpose, such as the Institute of Electrical and Electronic Engineers (IEEE), who are concerned with the standardization of a large number of technical systems and procedures. In conjunction with the testing of developed applications, they have also developed a working definition of "software test documentation," for example, the IEEE 829-2008, which provides concrete specifications for the creation of a test plan (Institute of Electrical and Electronics Engineers, n.d.). However, this exact specification would go far beyond the scope of this course.

A test can be performed on different levels. It is possible to test an application at a component or unit level. This means that the individual (sub-)components (or units) can be tested, or the entire application. If component tests are carried out first, this offers the advantage of error sources being recognized and eliminated earlier.

The focus here is on checking the syntactic correctness of an application. In this context, syntactic correctness is the correct application of the syntax of a programming language, such as the correct use of brackets or the confirmation of statements by means of a semicolon (e.g., in the programming language C). Modern editors, such as Adobe Dreamweaver, offer extensive functions for this, which recognize themselves whether an application is syntactically correctly formulated with regard to syntax (Williamson & Epstein, 2002). It is also necessary to check whether an application delivers the expected output data when input data is entered several times. This is usually done by developing a comprehensive test plan for an application, which is then systematically tested. A test plan is usually structured in such a way that it clearly defines the tasks and objectives of the test, i.e., it details what should be tested and in what order. The use of resources (e.g., hardware and employees) is also planned.

Another goal of a test plan is to test against the requirements of the project. Here, we want to test what the software will do if an undesired input is made. This can mean, for example, testing what happens when the value -1, 11, or a letter is entered in an input field that only allows numerical input from 0 to 10. In this example, we would expect an error message and no subsequent calculation. It is therefore common practice in software development to allot a larger time and monetary budget when testing the correctness of an application than was given for the actual development. The end of a software development process is followed by a validation period. This entails an examination of the developed application, which aims at evaluating the value of a software for its intended purpose. For this purpose, a requirements specification is typically drawn up and tested.

In addition to checking the correctness of a software's operation and functionality, it may also be necessary to characterize the performance of the web application. Performance characterization concerns the response time of a web page. The response time of a web page should be in the millisecond range so that a customer can enjoy using the web page and does not lose interest due to wait times, as this could result in them visiting another website or force them to reload the page in question. The latter can also artificially increase the load on the server in question, possibly slowing down the performance of the applications running on it. In addition to purely measuring the performance of a website, it is also of crucial importance to be able to detect and eliminate **bottlenecks** on a website (Walter, 2008).

Bottleneck

This is a link in a functional chain that limits the maximum efficiency.

Automatic Performance Characterization

To carry out a performance test, it is necessary to first define a test plan. This includes information about which requests are to be sent to the website (i.e., what information is sent, at what intervals requests are sent, or how many requests are sent). Types of requests include, for example, **HTTP requests**, **SOAP requests**, and **REST requests**. Since testing an application is typically a monotonous task, it is advantageous to automate it to a certain extent. For this purpose, the JMeter tool, developed by the Apache Group, can be used. This tool can independently check the correctness of a web page and its performance in a wide parameter space in a relatively automated manner (Apache, n.d.). It is possible to simulate a server load by calling the test protocol several times. This is known as a load and performance test or, more simply, a load test. This can be used to examine the reaction of the web application to these general conditions.

HTTP requests

Inquiries regarding the request for a website from a server are called HTTP requests.

SOAP requests

This stands for simple object access protocol requests and denotes how systems exchange data.

For the sake of completeness, however, it should be mentioned that the concrete tests (i.e., which parameters are to be tested and in which order) must still be defined and written down in code. The time required for this regularly exceeds preliminary estimates. JMeter contains extensive reporting tools that allow it to present the result of the examination of a web application to the developer in various ways. Alternatively, the performance tool Perl-LWP (Alders, n.d.) can be used to create HTTP requests that are sent to a server. The resulting responses can also be evaluated and processed using Perl-LWP.

REST requests

This stands for representational state transfer requests. These are requests regarding distributed systems.

If tests should be run automatically, “capture and replay” tools are used to execute load and performance tests. These are frameworks for application tests that make it possible to record user input or interactions with a web application (capture) and then run this recording repeatedly (replay) in order to test the desired scenario and simulate load on the server by calling it as many times as desired. Prominent examples of these capture and replay tools are the open-source framework Selenium (n.d.) and Hewlett-Packard’s Unified Functional Testing (UFT) (Micro Focus, n.d.).

Common Result of a Performance Characterization

When characterizing a website or a web application, the developer often comes to the same conclusion. Bottlenecks and performance losses are not usually caused by the network itself or by inefficient programming, but rather by database calls, which slow down loading times. It is always advantageous to communicate as little as possible with databases and keep data temporarily outside a database, instead influencing it accordingly wherever this makes sense and is possible in terms of performance (Walter, 2008).

Whether a software system as a whole fulfills requirements can only be tested after completion; however, the risk of waiting until the end of implementation to perform quality assurance is too high. Therefore, quality assurance in software engineering is carried out in parallel to all other activities.

Software Quality

In very general terms, the term **software quality** is defined in DIN ISO standard 9126 as the set of characteristics and characteristic values of a software product that relate to its ability to meet specified requirements (Cleff, 2010). According to this definition, software should be checked to see whether what has been delivered meets the previously defined requirements. Here, the importance of a specification becomes clear: according to DIN, the quality of software can only be decided on the basis of this specification (“defined requirements”) (Cleff, 2010). In practice, however, it becomes apparent that the perceived quality of a software is primarily determined by whether the customer thinks the software meets their actual requirements. Since many requirements are often only recognized during software development, it must be continuously ensured with regard to customer satisfaction that the set of specified requirements also contains the set of actual requirements.

Software quality

This is the capability of a software, i.e., web application, to conform to certain requirements or quality attributes.

In addition to determining the quality of a software system as a whole, the quality of artifacts generated during software development (software models, requirements, architectures, documents, and software components) must be determined.

Quality Management

Quality management

This includes all organized measures that seek to improve the quality of products, processes, or services.

The term **quality management** (QM) covers all organized measures that serve to improve the quality of products, processes or services of any kind. Since it is too much of a risk to wait until the completion of the software implementation to determine software quality, software fragments that have already been developed are tested during the development process. However, it is not only the generated program code that is considered in software quality management. Because the implementation is directly dependent on the specifications and decisions made during requirements engineering (which consist of specification and architecture activities), errors will carry over directly into the implementation. Therefore, these artifacts must also be considered in quality management for software engineering.

Typical quality management activities include

- quality planning. Quality requirements are prepared and documented in cooperation with the client.
- quality control. Quality testing activities are monitored, managed, and controlled during the software development process.
- quality inspection. Activities are proposed that will ensure that specified quality requirements for products, processes, and services are met.
- quality improvement. Product and process data is evaluated for improvement purposes.

Constructive and Analytical Quality Management

Quality management can be divided into constructive and analytical.

In constructive quality management, all quality characteristics of the products or processes are defined *a priori* (i.e., before creation) in order to avoid errors during software development and to ensure or increase the quality of the created artifacts. The measures used for constructive quality management include

- technical measures (e.g., use of modeling languages, tools, and development environments).
- organizational measures (e.g., guidelines, standards, templates, and checklists).
- socio-psychological measures (e.g., training, working atmosphere, and common hobbies).

Analytical quality management involves ex-post (i.e., after creation) measures to examine and evaluate the current quality level of the test objects in order to systematically detect defects and determine their extent. A distinction can be made between static and dynamic test procedures.

Static tests

During a review or audit, the test object is analyzed, assessed, and examined. The information obtained is compiled, condensed into metrics or key figures if necessary, and, finally, evaluated.

Dynamic tests

The system under test (SUT) is executed with fixed input values and the result is evaluated. Through constructive quality management, the effort required in analytical quality management can be reduced.

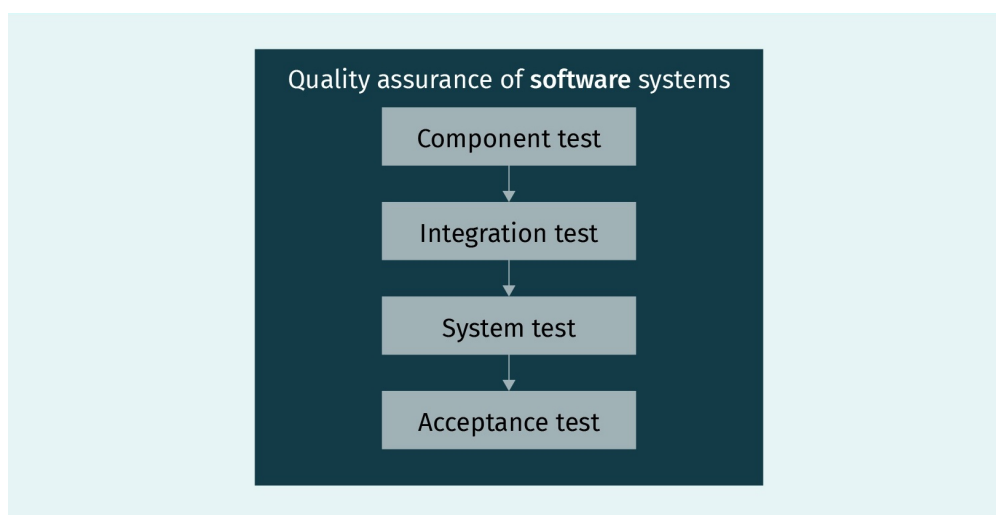
Test Object

All of the artifacts created in software engineering can be tested for compliance with quality requirements. This means that, in addition to the generated program code, professional requirements, technical specifications, architectural descriptions, and even the test cases can be tested. However, dynamic tests are only possible for executable artifacts. All other artifacts are tested using static procedures.

Test Levels

As mentioned above, it is too much of a risk to wait until implementation is complete to start software quality assurance. However, in industrial software development, testing activities are started as soon as the first software artifacts have been generated and before the system as a whole can be tested in its deployment environment. The figure below shows a schematic overview of the various test stages in software engineering.

Figure 100: Test Levels for Quality Assurance of Software Systems



Source: Brückmann (2013).

Component Test

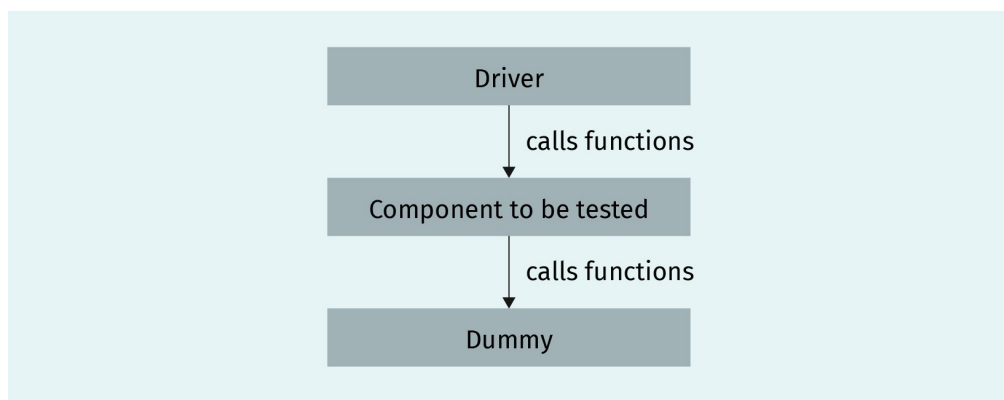
To enable the implementation of a software system by several software developers, the system is broken down into its logical parts (i.e., components, modules, classes, building blocks). This is also known as a module test or unit test. Each component is implemented separately and integrated into the system after completion. During or after completion of each software component, it can be checked for compliance with specified requirements. The isolated testing of individual software components is called component testing. Examples of this include

- premium calculation, where the calculation of insurance premiums are carried out depending on the selected benefits.
- a shopping cart, where the total cost of all items is calculated.
- an address component, where the correct address for correspondence is generated from customer data.

Integration Test

As soon as at least two software components have been completed, they can be combined to form a system (i.e., integrated). During or after integration, the interaction of groups of components is tested in integration tests. These integration tests check whether the components work together as described in the specification. With complex software systems that have many components, often not all components are finished at the same time. In these cases, missing components are simulated by “drivers” and “dummies.” The figure below illustrates the interaction of components, drivers, and dummies.

Figure 101: Drivers and Dummies



Source: Brückmann (2013).

Drivers are designated software fragments that simulate calling other components. Dummies simulate components that are called by other components. Technical interfaces for external systems are simulated in the integration test using dummies and drivers. This is because only the finished software system can be connected to external systems, but the interaction of the interfaces must be tested beforehand. Examples of this include testing

- the integration of an address component with an external service for address validation.
- the integration of a PDF printer with a digital shopping cart for generating invoices.
- the integration of an inventory system with a collection system for automatically sending invoices.

System Test

After development work has been completed and all software components have been integrated into a finished system, the system as a whole is tested. The aim of the system test is to check whether the system meets the specified requirements. In addition to the functional requirements, specified quality characteristics are also tested. Among other things, how the system behaves in particularly demanding situations (e.g., a high number of users and large amounts of data over both short and long periods of time) is checked by means of targeted stress and load tests. The system test is usually performed without the involvement of the client, and is the last test the organization creating the system performs before the software is handed over.

A major challenge in system testing is to replicate the customer's environment as faithfully as possible. This includes the hardware and software environment in which the system is operated, as well as the software systems to which the system under test has technical interfaces, a realistic utilization and realistic user behavior, and real data sets, if possible. The provision of data sets that are both original and comply with data protection regulations represents a major and often insoluble problem.

Acceptance Test

The final stage is the acceptance test, in which the finished system is installed at the client's location and tested under actual operating conditions. The acceptance test checks whether the system meets the contractually agreed performance characteristics from the client's point of view. The client either carries out the acceptance test themselves or is involved in its execution. The acceptance test is a special type of system test, and it is generally difficult to draw a clear distinction between the two. However, the most important difference compared to system testing is that it is performed by an organization other than the one that developed the system. A successful acceptance test is often the contractually agreed prerequisite for invoicing.

6.2 Basic Security Concepts and Principles

Since websites are typically held on the internet, which is accessible to the public, it is not surprising that a large number fall victim to cyberattacks. The importance of web application security is a point that is often underestimated in practice when developing web applications. One key aspect in particular is access control, the settings concerning who is allowed to access various parts of a server. It is fair to assume that a user requires limited access rights to a server, since they usually only need to execute simple read commands in

a directory. This contrasts to the requirements of an administrator, who requires full access and the ability to change the configuration of a server. If a user obtains administrator rights by mistake, they would have full access to a server. This should be avoided, especially with respect to cyberattacks.

A standardized procedure for improving the security of applications and services on the internet is offered by the Open Web Application Security Project (OWASP). This is a community of companies, individuals, and educational institutions from all over the world which, as a non-profit organization, is committed to making transparent decisions for the end user regarding the security risks of software (Open Web Application Security Project, n.d.). The organization has compiled (and regularly updates) a list, the “OWASP Cloud Top 10,” which summarizes the most dangerous risks for web applications. In the following section, we will take a brief look at this list (Wetter, 2012; Open Web Application Security Project, n.d.):

Table 5: Overview of OWASP

Risk	Meaning
Insecure data storage	The incorrect and improper or even insensitive storage of credentials and confidential data
Weak server side controls	Weak security measures on a server
Insufficient transport layer protection	Transmission of sensitive data in plain text, i.e., without encryption
Poor authorization and authentication	Lack of security measures during authorization and authentication of a login process
Client side injection	Insertion of malicious content into a database, e.g., through SQL injection
Improper session handling	Lack of protection of an authorized or authenticating login
Security decisions via untrusted inputs	When developing applications, it is possible to use hidden fields and values or hidden functionality to separate users with high and lower access rights. This can be tapped by attackers.
Side channel data leakage	Access to cache content by untrusted third parties
Broken cryptography	Non-functioning encryption of data transmission
Sensitive information disclosure	The hard coding of sensitive data

SQL injection
This is a frequently used attack procedure used for targeted injection of malicious code into an SQL database.

Hard coding
This is the deposit of data into the source code of an application.

Source: Sturm (2020), based on Wetter (2012).

Structure of a Web Application

A typical web application involves the interaction of several computers or server systems (including the client computer, which sends HTTP requests to a web server), the web server (which receives and processes the client's requests), the application server (which can execute the corresponding web programming), and, sometimes, a database server. The first thing to consider when assessing the security of web applications is making only the necessary parts of the system visible to the outside world. In the present example, this applies exclusively to the web server.

Larger network structures usually consist of several subnetworks that are interconnected by means of switches. The advantage of this is that access control can take place on each of the switches to prevent attackers from penetrating further into the network. Each of the network components that are to be directly accessible from outside are upstream of the hidden network. The upstream network is often referred to as the demilitarized zone (DMZ), in which the respective web server can be found. The sensitive database server, on the other hand, would be located in a part of the network that is not accessible from the outside and can only be reached via the internal IP addresses of the network.

Configuration

A lack of security on a server can be due to both developer errors in the server system and configuration errors in the system, the former of which can often be eliminated by updates and patches from the manufacturer. The configuration of a server system is usually a complex undertaking and improper configuration can allow attackers to gain access to the server. This is usually due to incorrectly opened ports, but can also be due to incorrectly assigned access rights for the server's folders. Risks arising from port releases can be minimized, for example, by using and correctly configuring a firewall. The underlying concept here is that individual services on a system, e.g., a server or a PC, can be made accessible to the outside world via specific ports. A server or PC can be visualized as a house with a large number of windows. Each window is assigned to a specific service through which the system or PC can interact with the outside world (Walter, 2008). The basic idea is to close all ports and windows and block access from the outside in order to exclude unwanted visitors. The only exceptions are the ports that must be used. This is achieved by using an access list. Here, it is possible to set up port forwarding so that requests from the outside are forwarded to the appropriate part of the server if the firewall is properly configured.

In principle, it is also advisable to run as few services on a server as possible. Services are processes that fulfill a specific task, such as resource management or user administration (Mandl, 2020). This stems from the fact that every service executed can also represent a security risk for the server. Thus, by reducing the number of services running, the opportunity for intrusion attempts can also be minimized.

Denial of Service (DoS) and Distributed Denial of Service (DDoS)

In addition to targeted attempts to influence a server by penetrating the system, there is also the possibility of crippling a server through a denial of service (DoS) or a distributed denial of service (DDoS) attack. Here, the idea is to exhaust a server with a flood of simul-

taneous requests so that it eventually becomes unavailable to customers. The performance of the server is denied (denial). The ubiquity of web services, which is desirable (and sometimes vital) to corporations, can be deliberately influenced by attackers. If this attack originates from an attacking computer, we are dealing with a DoS attack. Furthermore, it is also possible to initiate this type of attack simultaneously, e.g., by means of viruses that are present on a large number of affected computers, and thus carry out a global attack. This is referred to as a DDoS (Eckert, 2018).

**SUMMARY**

When developing web pages and web applications, and before the actual programming begins, it is necessary to record which requirements the website should fulfill in a requirements specification or requirements analysis. Based on this document, it is possible to create initial activity diagrams, which can be translated into code in a final step.

After the development of a web page or a web application has been completed, the next step is to test the web application. The aim here is to determine whether the web application performs a function as expected.

In quality assurance activities, the artifacts generated in the course of software development (program code, documents, models, test cases) are checked to determine whether they meet specified requirements. Both measures for constructive and analytical quality assurance are used. The quality assurance of program code takes place in various test stages, depending on the current status of the project.

Security in web applications plays a role that should not be underestimated in maintaining the functionality of a website or a web application and protecting sensitive data from unauthorized access. In the operation of software systems, care must be taken to ensure that the system is available to users and secured against failure and threat scenarios. Changes to running systems must be planned with extreme care and implemented with the help of release plans. In addition to release planning, IT service management activities are part of operations.

After the first release of a software system, continuous adjustments and changes should be made to the system. During this further development, business functions are added to the system or existing business functions are changed.