# **Supporting CSS Multi Direction Languages in 2023**

Supporting a multi-direction languages website, which supports both common language directions (left-to-right and right-to-left), has historically required a lot of work.

However, in 2023, after we obtained most of the main features of CSS Logical Properties, a native way to support multi-direction language websites, it is now a lot easier.

Despite this, however, CSS Logical Properties still need to be improved and require additional solutions.

In this post, I want to examine how we solved the missing parts of supporting a multi-direction language website.



# **What are CSS Logical Properties?**

Let me say a few words on CSS Logical Properties in case you are not familiar with them.

In the past, websites worked with fixed axes according to physical directions, such as top/right/bottom/left.

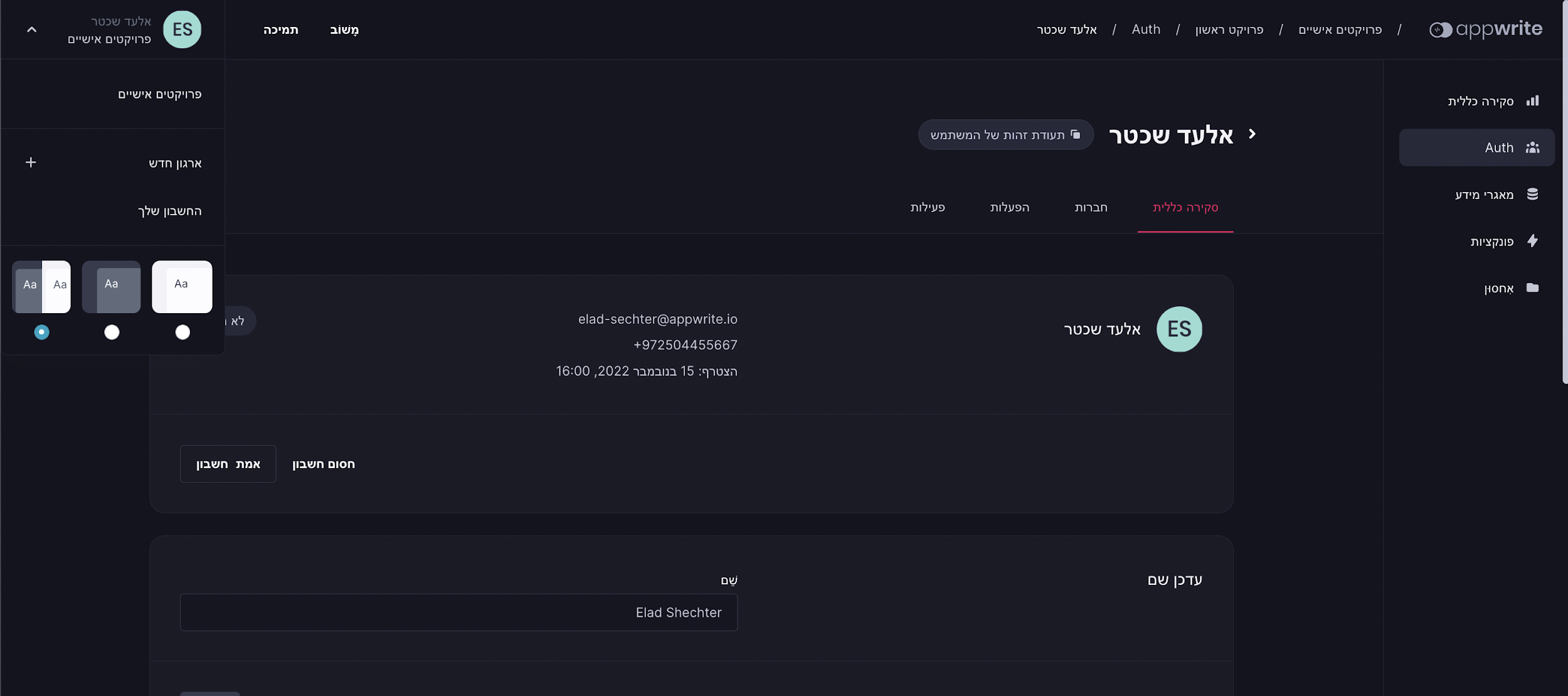
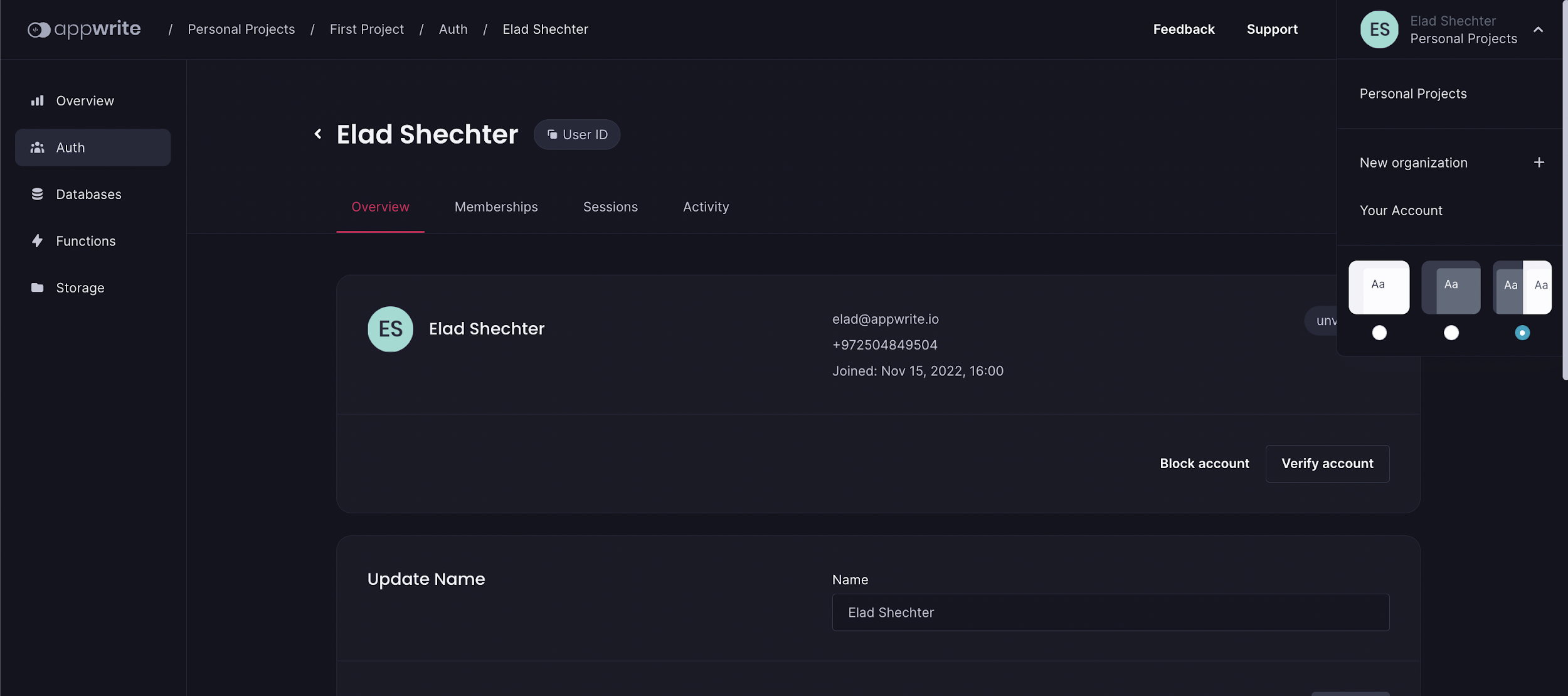
From those physical directions, we got our old familiar CSS Physical Properties, such as margin-right, padding-bottom, border-left, and so on.

## **The Problems with Physical Properties**

When trying to support a multi-direction languages website using languages that are going from left-to-right (direction: ltr) and right-to-left (direction: rtl), in general you need to replace the second type of language, all left-to-right or all right-to-left, resulting in a kind of horizontal mirror of the website.

This means that we needed to load a different CSS file for every type of such languages.

**Example of How a Multi-Direction Website Can Look:**

****

## **CSS Logical Directions**

The solution to this issue came with CSS Logical Properties. Instead of the physical directions, we now have two axes:

* **Inline axis** – this is the axis of the text.
* **Block axis** – this is the axis of the flow of the website.

Each of those axes has a start and an end direction.

**Inline axis:**

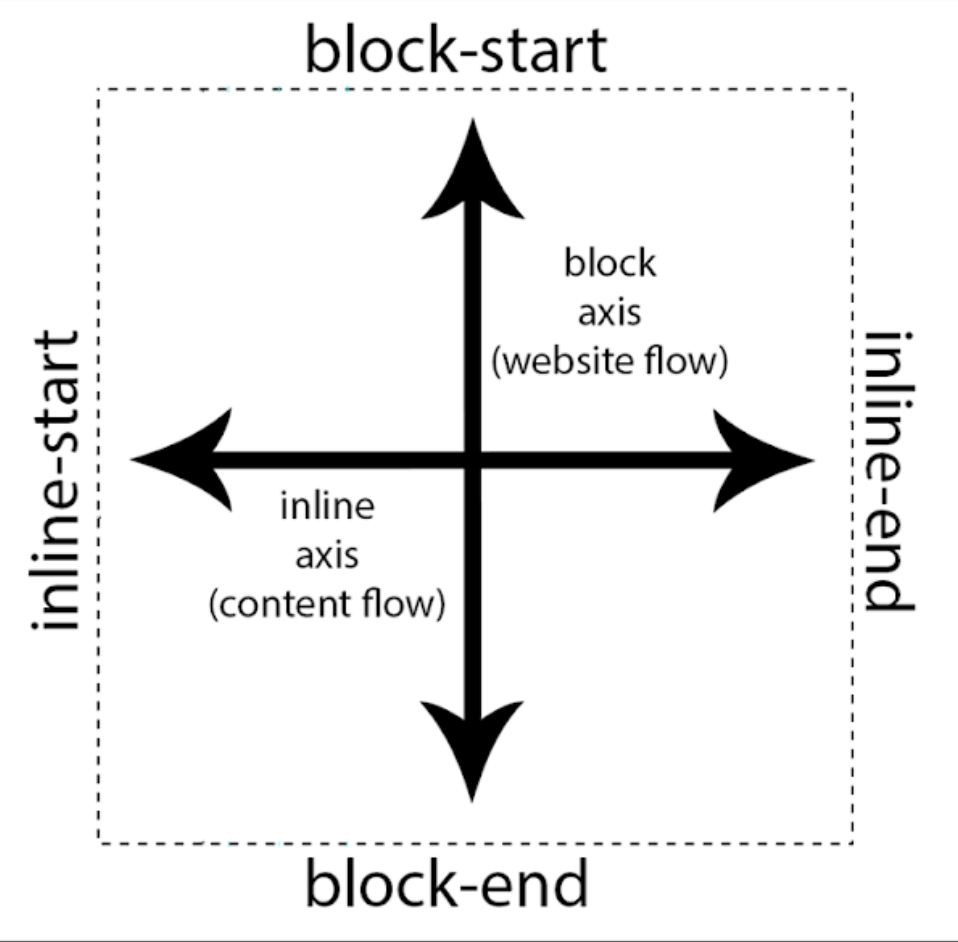
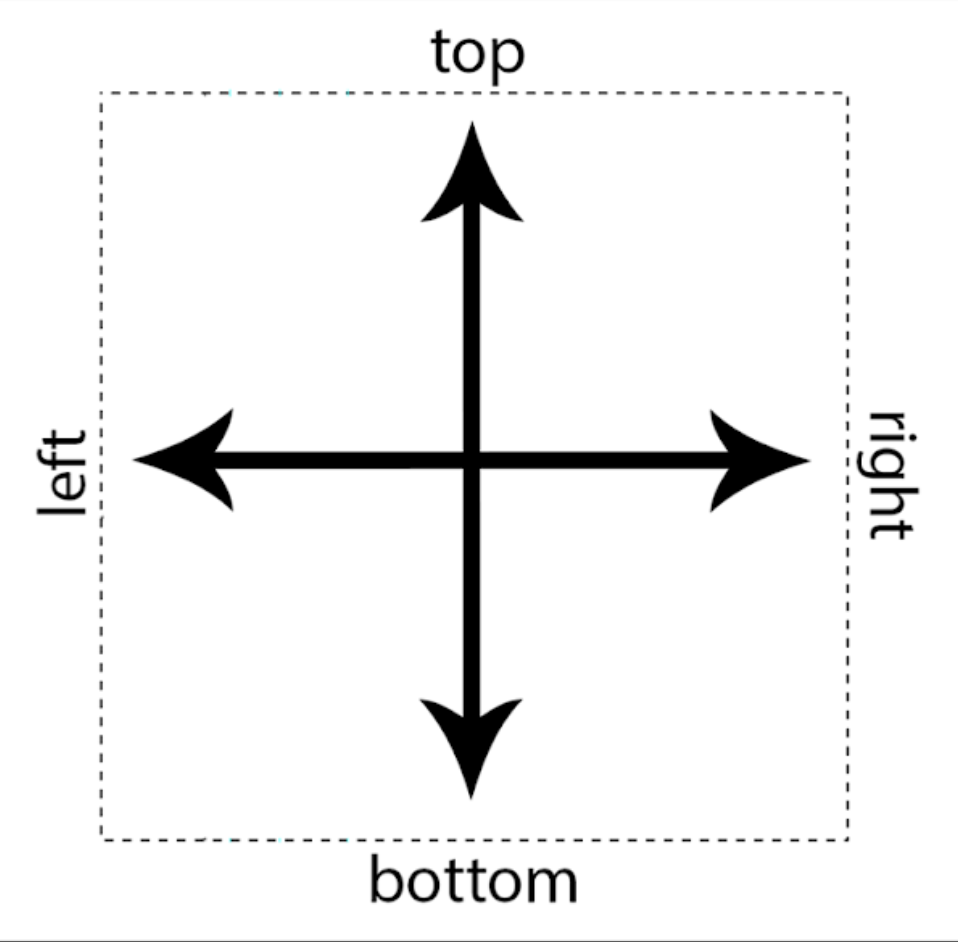
To describe the inline axis, we have its two directions:

* **Inline-start** – describes the start of the line of text; in English, this is the **left**.
* **Inline-end** – describes the end of the line of text; in English, this is the **right**.

**Block axis:**

To describe the block axis, we have its two directions:

* **Block-start** – describes the start of the flow of the website; in English, and most languages today, this is the **top**.
* **Block-end** – describes the end of the flow of the website; in English, and most languages today, this is the **bottom**.



From CSS Physical direction to Logical Directions

## **CSS Logical Properties**

From these logical directions, we obtain updates for most of our physical properties, for example:

margin-left => margin-inline-start

padding-top => padding-block-start

/\* position properties - for example for: position: fixed; \*/

top => inset-block-start

bottom => inset-block-end

left => inset-inline-start

right => inset-inline-end

When using the new CSS Logical Properties, however, the values flip according to the direction property value: ltr (left-to-right [default]) or rtl (right-to-left).

**Example of left-to-right:**

html {

direction: ltr; /\* default value \*/

}

div {

margin-inline-start: 20px; /\* = margin-left: 20px \*/

}

**Example of right-to-left:**

html {

direction: rtl;

}

div {

margin-inline-start: 20px; /\* = margin-right: 20px \*/

}

## **What is Top and Bottom?**

This affects the **main axis**, the flow of the website, although it is not commonly used on the web. It is made for old Far East languages such as traditional Chinese (not the familiar “simplified” Chinese). For this, we have the writing-mode property.

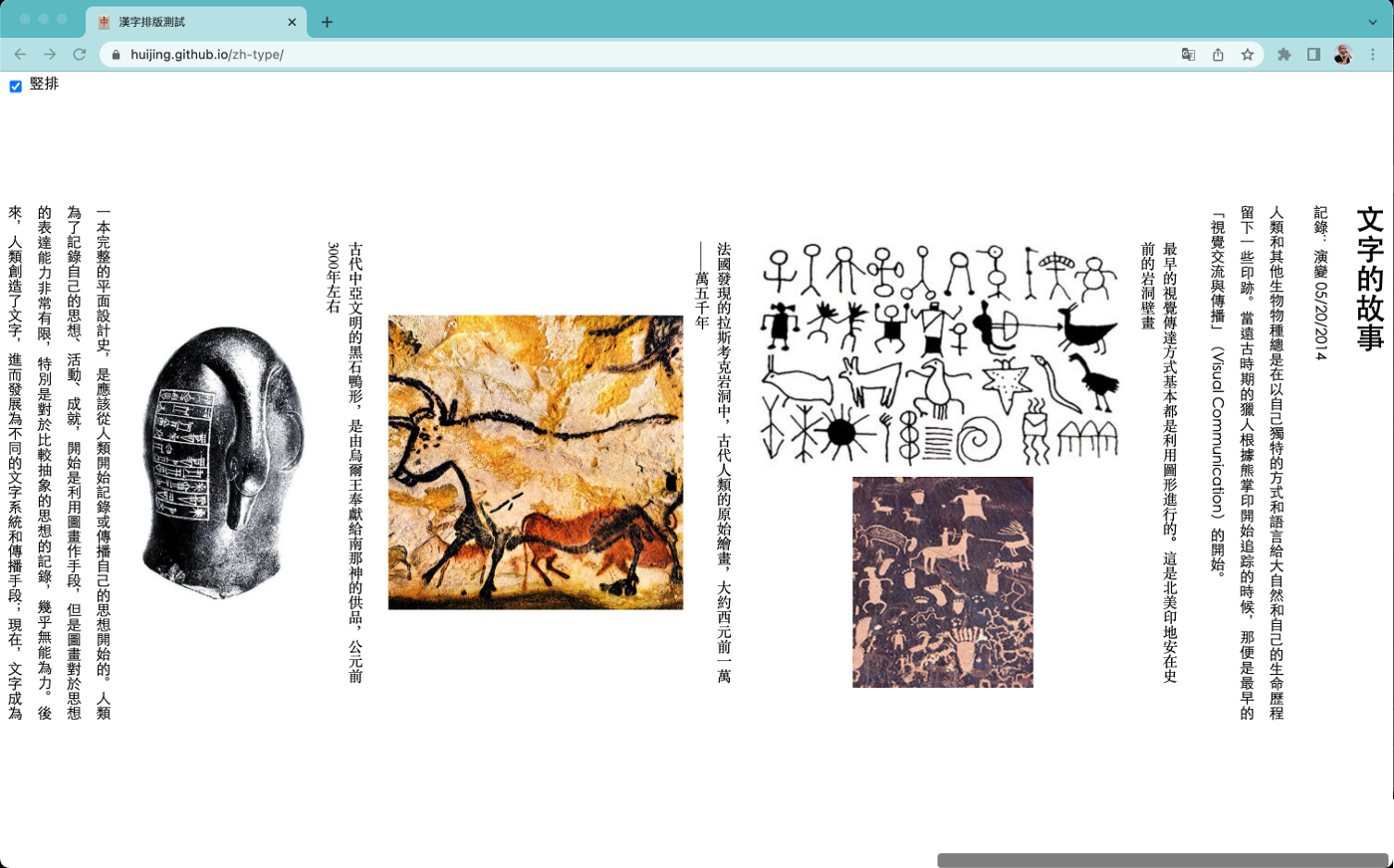
writing-mode: horizontal-tb; /\* top to bottom (default value) \*/

writing-mode: vertical-rl; /\* right to left \*/

writing-mode: vertical-lt; /\* left to right \*/

Changing the writing-mode value into vertical-lr or vertical-rl values will change to flow of the website instead of top-to-bottom into right-to-left or left-to-right. **This means that the top is not the top anymore!** The **scroll is now horizontal** instead of vertical, and the **text goes from top to bottom**.

[**Demo**](https://huijing.github.io/zh-type/) (by [HJ Chen](https://twitter.com/hj_chen)):



The writing-mode property is almost unused; 99.99% of websites work only according to text direction property, ltr or rtl.

This also means that using the main axis/block axis properties, such as margin-block-start, does not affect us day-to-day. However, I still prefer using them to keep all the websites using CSS Logical Properties.

# **Are CSS Logical Properties Good Enough?**

CSS Logical Properties will do most of the job, but I encountered several problems after using them. Let’s talk about those problems and how you can solve them.

# **Transform Properties**

These days, we do most of the layout alignment with modules of CSS Flexbox and CSS Grid. However, in some cases, we want to locate things via the transform property using the translateX() value, for example.

The problem with the transform property is that its axes do not work in logical directions. For example, transform: translateX(100px) will always move to the right, no matter the direction value of the website.

transform: translateX(100px); /\* will move the element always right \*/

## **Solution**

If we need to flip the element to the second side according to the type of language, we need the value to change to a negative value instead of a positive value.

To do so, I create a CSS variable that by default has a value of 1.

:root {

--transform-direction: 1;

}

I multiply this value with the translateX value using the CSS calc() function.

transform: translateX( calc(-100% \* var(--transform-direction)) );

With the value of 1, it does not have any effect, which is good because this is the default, and we do not want to see any effect.

**Support for RTL languages:**

Now, the only thing left for me to do is to change the value of the --transform-direction variable to -1.

To do so, I simply created another :root selector, which overrides only when the main <html> element has the native dir=“rtl” attribute on it.

:root[dir="rtl"] {

--transform-direction: -1;

}

This will take affect only when the <html> element has the dir=“rtl” attribute:

<html dir="rtl">

# **Related Direction Arrows Icons**

When changing the website direction, some things can give unexpected results, such as an arrow pointing in the wrong direction.



To solve this issue, we can again use the — transform-direction variable, but now in a slightly different way.

To flip the icon horizontally, I’m using the scaleX function of the transform property. Now, I’m just giving it the — transform-direction value.

In this way, when the dir=“rtl” for right-to-left language styles we declared before sets the value equal to -1, this value will flip the arrows in the horizontal direction and keep them in their own places.

.icon-arrow-left,

.icon-arrow-right {

transform: scaleX(var(--transform-direction));

}

This opposite direction can look strange to us but remember that this is to support right-to-left languages such as Arabic and Hebrew.



# **More Things**

There are several CSS Logical Properties, or more precisely logical values, that are not widely supported at present.

One of these is the logical value of the float property. The float supports the new logical values inline-start and inline-end, but for now they are only supported in Firefox.

To solve this, I am adding two new variables that represent the start and the end of the line.

The default values, of course, are according to English, with the ability to override those values in case we have the dir=“rtl” attribute of the <html> element.

/\* default - left-to-right languages \*/

:root {

--start-direction: left;

--end-direction: right;

}

/\* Support for right-to-left languages \*/

:root[dir="rtl"] {

--start-direction: right;

--end-direction: left;

}

Now the only thing left for me to do is to use the variables according to my need. Example:

float: var(--start-direction);

Those variables can be useful in more cases, such as setting the background-position property, for example:

background-position: var(--end-direction) top;

background-repeat: no-repeat;

# **CSS Variables for Help**

As we saw here, I defined three CSS variables that help me to complete the support of CSS Logical Properties.

The CSS code looks like this:

/\* default - left-to-right languages \*/

:root {

--transform-direction: 1;

--start-direction: left;

--end-direction: right;

}

/\* Support for right-to-left languages \*/

:root[dir="rtl"] {

--transform-direction: -1;

--start-direction: right;

--end-direction: left;

}

If you are using **Sass preprocessor,** you can write it in this way:

:root {

/\* default - left-to-right languages \*/

--transform-direction: 1;

--start-direction: left;

--end-direction: right;

/\* Support for right-to-left languages \*/

&[dir="rtl"] {

--transform-direction: -1;

--start-direction: right;

--end-direction: left;

}

}

# **To Summarize**

In this post, I examined how you can support a multi-direction website.

The first way is by using wherever you can native CSS Logical Properties. The second way is by using three CSS variables.

# **Final Words**

That’s all.

I hope you have enjoyed this article and learned from my experience.

If you like this post, I would appreciate likes and sharing :-)

You can follow me via [**Twitter**](https://twitter.com/eladsc).

**Who Am I?**

I am Elad Shechter, a CSS architect at Appwrite.