# **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, took a lot of work several years ago.

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

But with all the mentioned above, 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’s say a few words on CSS Logical Properties if you aren’t familiar with them.

In the past, the website worked with fixed axis according to physical directions, such as: top/right/bottom/left.

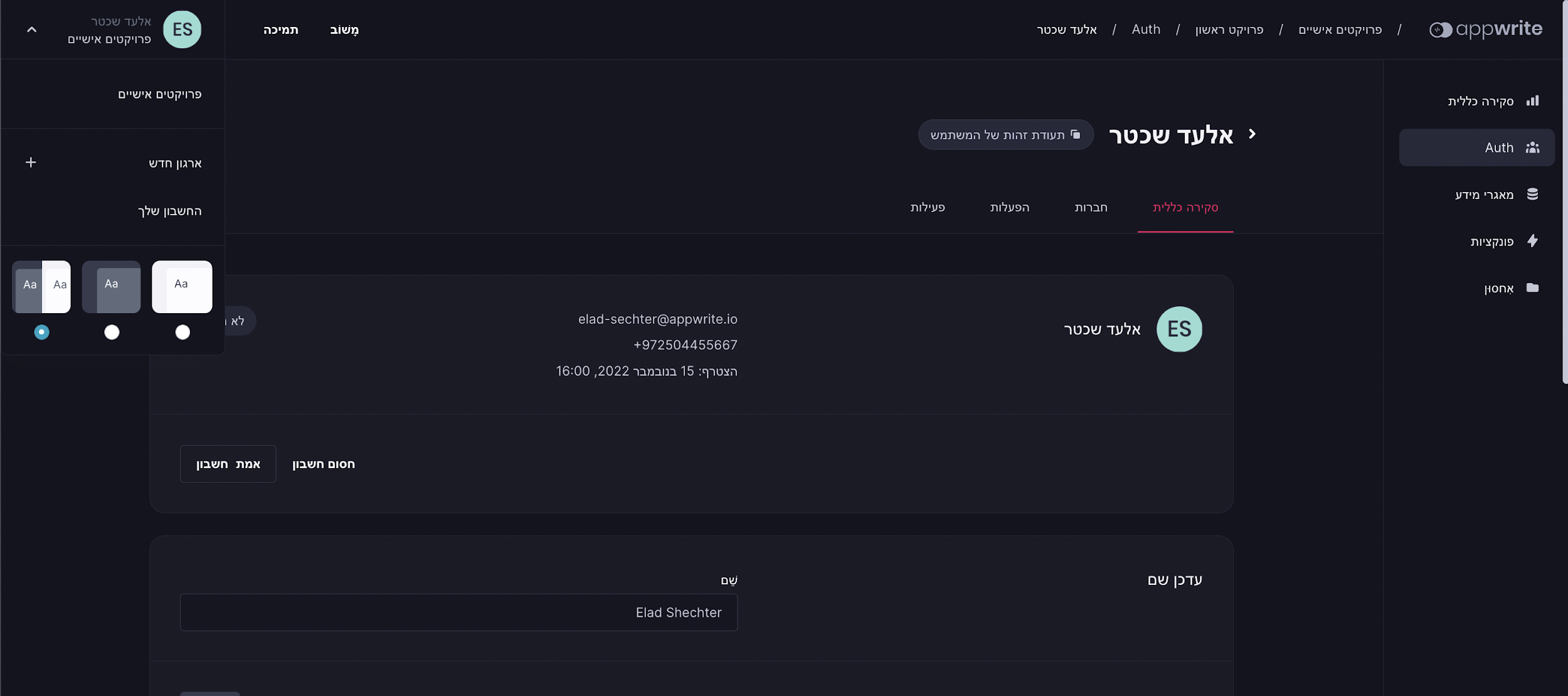
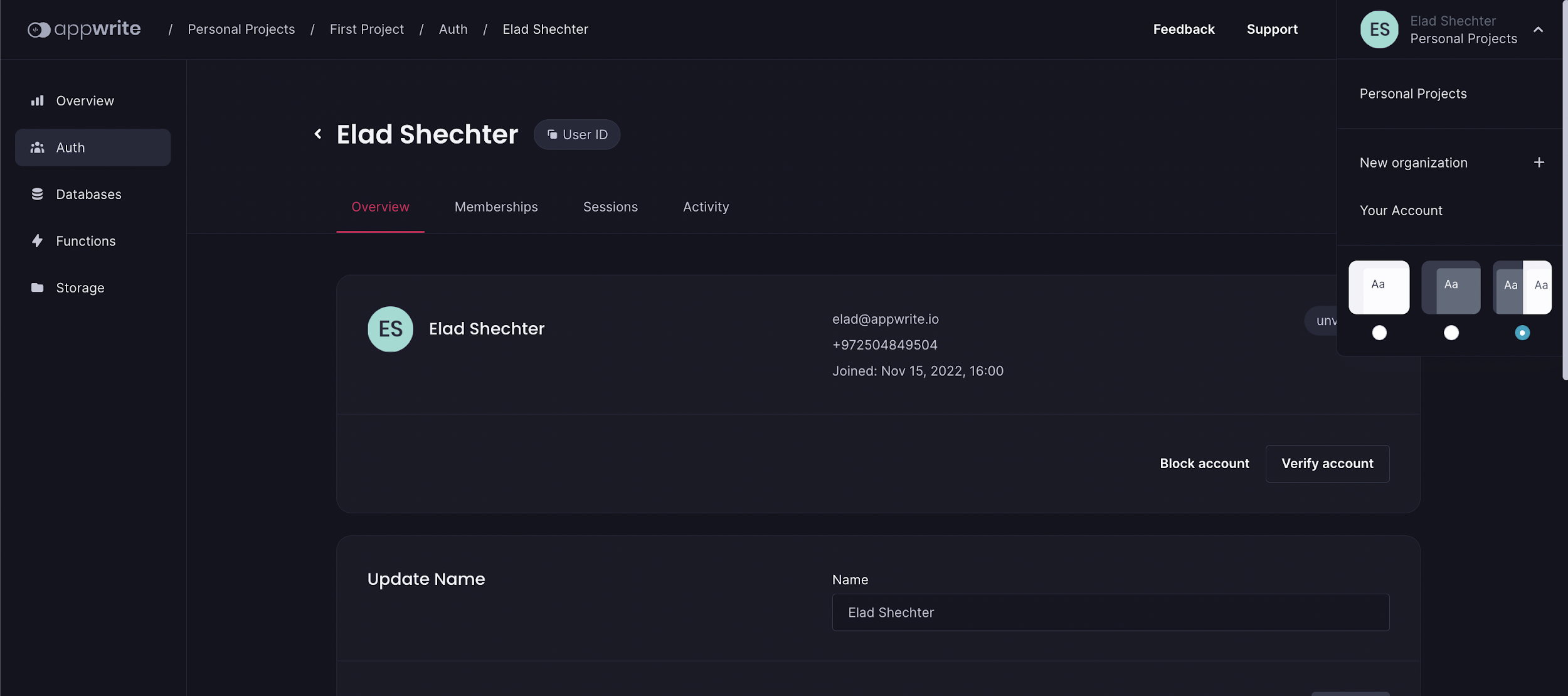
From those physical directions, we got our old familiar “CSS Physical Properties” like: margin-right, padding-bottom, border-left, and so on.

## **The Problems With Physical Properties**

When trying to support multi-direction languages website, like 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 and all right to left. It is a kind of horizontal mirror of our website.

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

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

****

## **CSS Logical Directions**

To solve this issue came 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.

Every one of those axes has a start and end direction.

**Inline-axis:**

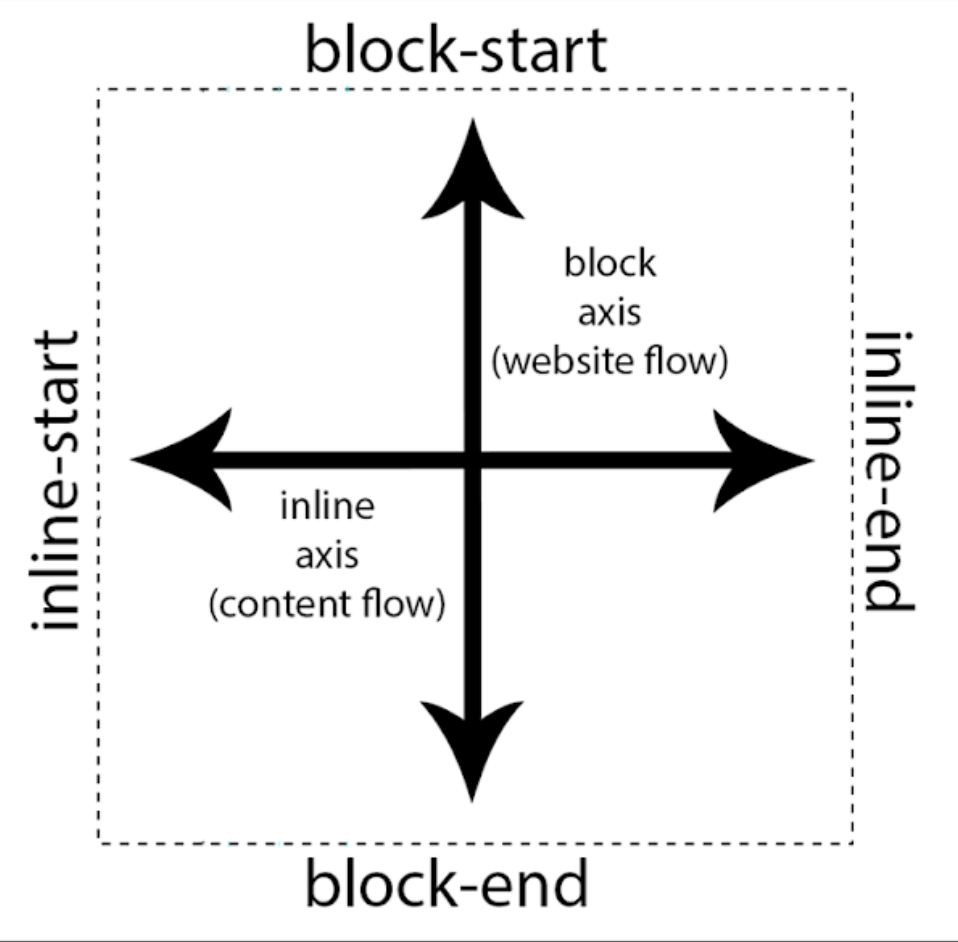
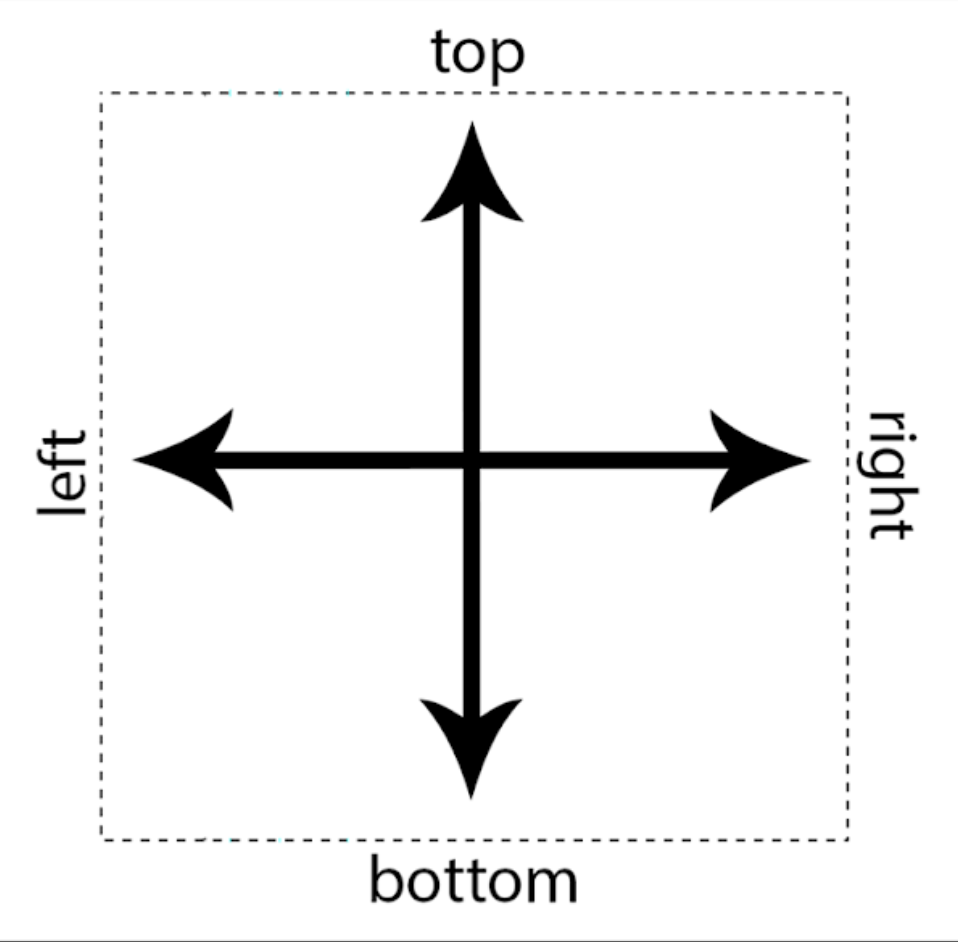
To describe the inline-axis, we have its two directions:

* **Inline-start** — describe the start of the line of text; in English, it's the **left** direction.
* **Inline-end** — describe the end of the line text; in English, it's the **right** direction.

**Block-axis:**

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

* **Block-start** — describe the start of the flow of the website; in English, and most of the languages today, is the **top**.
* **Block-end** — describe the end of the flow of the website; in English, and most languages today, it is the **bottom**.



From CSS Physical direction to Logical Directions

## **CSS Logical Properties**

From these Logical directions, we got 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

Now, when using those new CSS Logical Properties, the value flip according to the direction property values: ltr (left-to-right/default) or rtl (right-to-left).

**Example in left-to-right:**

html {

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

}

div {

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

}

**Example right-to-left:**

html {

direction: rtl;

}

div {

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

}

## **What is Top & Bottom?**

To affect the **main-axis**, the flow of the website, it's less commonly used on the web. Its made for old far east languages like traditional Chinese (not the correct familiar [simlify] Chinese); for that, we have the writing-mode property.

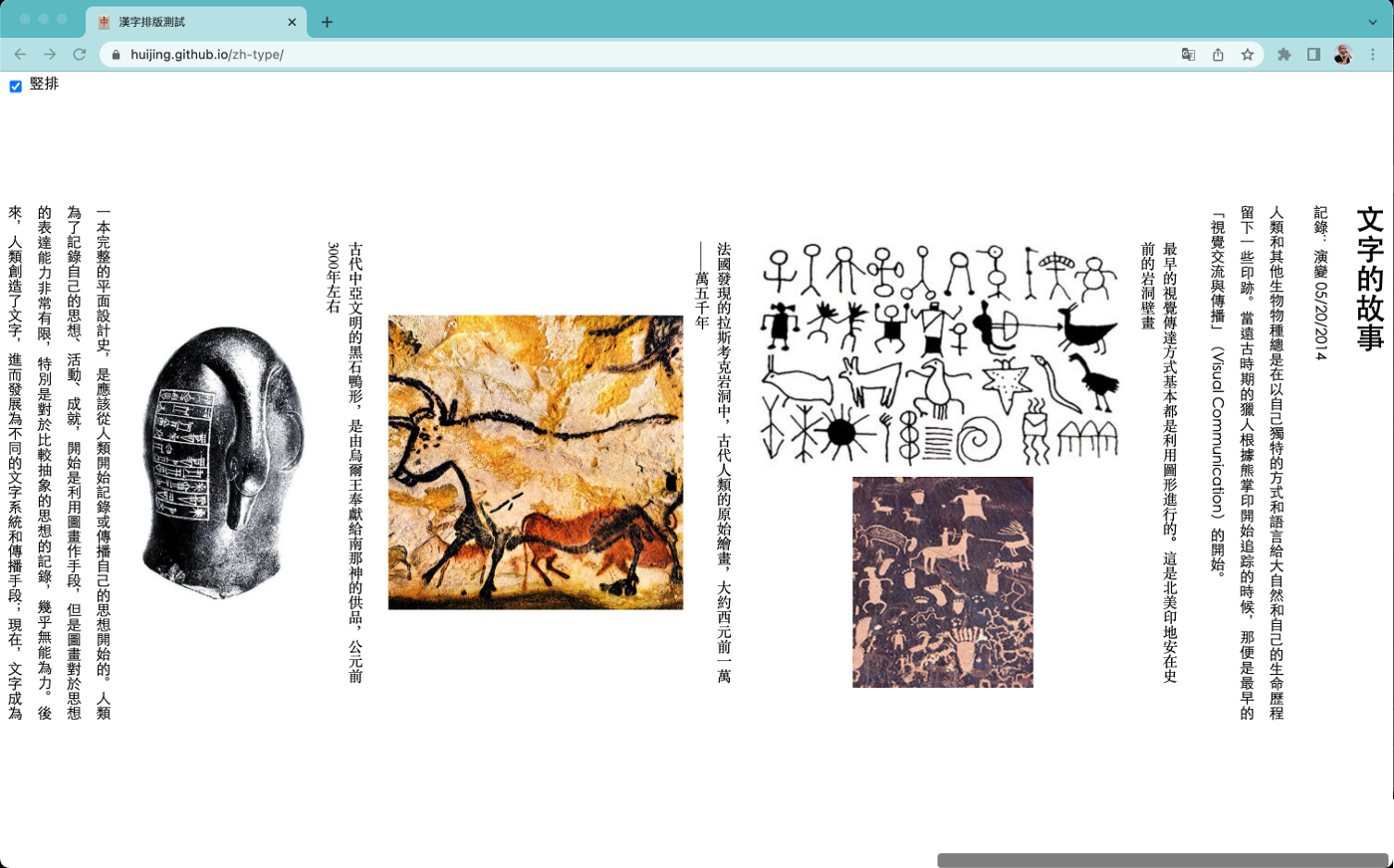
writing-mode: horizontal-tb; /\* top to bottom (default value) \*/

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 isn’t 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 percent of websites are working only according to text direction property, ltr or rtl.

This also means that using the main-axis/block-axis properties, like margin-block-start, aren’t affecting us in the day to day. Mentioning the above, I still prefer using them to keep all the websites with 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. Now let’s talk about those problems and how you can solve them.

# **Transform Properties**

Most of the layout alignment these days we do with modules of CSS Flexbox and CSS Grid. But in some cases, we will want to locate things via the transform property via translateX() value, for example.

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

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

## **Solution**

If yo 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 which by default is with the value of 1.

:root {

--transform-direction: 1;

}

This value I multiple with the transalteX value using the CSS calc() function.

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

With the value of 1, it doesn’t make any effect, which is good because this is the default, and we don’t want to make any effect.

**Support RTL languages:**

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

To do so, I just 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 affect only when the <html> element is has the dir=”rtl” attribute:

<html dir="rtl">

# **Related Direction Arrows Icons**

When changing the website direction, some things can get unexpected results, like an arrow pointing in the wrong direction.



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

To flip the icon in horizontal way 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” of right-to-left languages style styles we declared before will affect the value and it will be eqall to -1, this value will flip the arrows in horizontal direction and keep them in there own places.

.icon-arrow-left,

.icon-arrow-right {

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

}

This opsite direction can look to you weird, but remember that this is to support right-to-left languages like Arabic and Hebrew.



# **More Things**

There are several of CSS Logical Prperties, or more procise logical values that aren’t supported good enough.

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

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

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

/\* defalut - 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 like the background-position property, for example:

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

backgroun-repeat: no-repeat;

# **CSS Variables for Help**

As we so here, I defined three CSS variables that are helping me to complete the support of CSS Logical Properties.

The CSS code for it will look like this:

/\* defalut - 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 {

/\* defalut - 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. And the second wayThis is while using three CSS variables.

# **Final Words**

That’s all.

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

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

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

**Who Am I?**

I am Elad Shechter, a CSS Architect at Appwrite.