Software Development 101

CSS

#### Specificity

Specificity is the algorithm used by browsers to determine the CSS declaration that is most relevant to an element and determines the property value to apply to the element. The specificity algorithm calculates the weight of a CSS selector to determine which rule from competing CSS declarations gets applied to an element.

**ID column**

Includes only ID selectors, such as #example. For each ID in a matching selector, add 1-0-0 to the weight value.

**CLASS column**

Includes class selectors, such as .myClass, attribute selectors like [type="radio"] and [lang|="fr"], and pseudo-classes, such as :hover, :nth-of-type(3n), and :required. For each class, attribute selector, or pseudo-class in a matching selector, add 0-1-0 to the weight value.

**TYPE column**

Includes type selectors, such as p, h1, and td, and pseudo-elements like ::before, ::placeholder, and all other selectors with double-colon notation. For each type or pseudo-element in a matching selector, add 0-0-1 to the weight value.

**No value**

The universal selector (\*) and the pseudo-class :where() and its parameters aren't counted when calculating the weight so their value is 0-0-0, but they do match elements. These selectors do not impact the specificity weight value.

**Inline styles**

Inline styles added to an element (e.g., style="font-weight: bold;") always overwrite any normal styles in author stylesheets, and therefore, can be thought of as having the highest specificity. Think of inline styles as having a specificity weight of 1-0-0-0.

The only way to override inline styles is by using !important.

**The !important exception**

CSS declarations marked as important override any other declarations within the same cascade layer and origin. Although technically, !important has nothing to do with specificity, it interacts directly with specificity and the cascade. It reverses the cascade order of stylesheets.

Flexible Box Module

Flexbox is a one-dimensional layout method for distributing **flex items** in rows or columns.

Flex items are the 'children' of a flex container. They are positioned along a **main-axis** and a **cross-axis**. The main-axis is horizontal by default, so the items flow into a row. You can flip the main-axis by setting flex-direction to column .

Flex direction

Adding the flex-direction property to the flex container allows us to change the direction in which our flex items display.

* flex-direction: row;
* flex-direction: row-reverse;
* flex-direction: column;
* flex-direction: column-reverse;

**Alignment, justification and distribution of free space between items**

A key feature of flexbox is the ability to align and justify items on the main- and cross-axes and to distribute space between flex items. Note that these properties are to be set on the *flex container*, not on the items themselves.

**Align content**

The align-items property will align the items on the cross-axis. The options are:

* stretch
* flex-start
* flex-end
* center

**Justify content**

The justify-content property is used to align the items on the main axis, the direction in which flex-direction has set the flow. The initial value is flex-start which will line the items up at the start edge of the container, but you could also set the value to:

* flex-start
* flex-end
* center
* space-around
* space-between
* space-evenly

#### Media query syntax

A media query is composed of an optional **media type** and any number of **media feature** expressions, which may optionally be combined in various ways using logical operators. Media queries are case-insensitive.

Media types define the broad category of device for which the media query applies: all, print, screen. The type is optional (assumed to be all) except when using the not or only logical operators.

Media features describe a specific characteristic of the user agent, output device, or environment:

* any-hover
* any-pointer
* aspect-ratio
* color
* color-gamut
* color-index
* device-aspect-ratio Deprecated
* device-height Deprecated
* device-width Deprecated
* display-mode
* dynamic-range
* forced-colors
* grid
* height
* hover
* inverted-colors
* monochrome
* orientation
* overflow-block
* overflow-inline
* pointer
* prefers-color-scheme
* prefers-contrast
* prefers-reduced-motion
* resolution
* scripting
* update
* video-dynamic-range
* width