CSS for PMG developers

CSS Specificity

CSS Specificity

Is the algorithm that the browser uses to determine which CSS rule to apply when multiple rules could apply to the same element.

Code sandbox (1)

Code sandbox (2)

CSS Specificity

Specificity is calculated based on the types of selectors used in a rule. Selector with highest score wins.

Consist of values in 3 columns (from the most important one)

- ID Column - How many times ID selector ('#selector') is used
- Class Column - How many times class selector ('.selector'), attribute selector ('attr') and pseudo-class selector (':pseudo-class') is used.
- Element Column - How many times element selector ('element') and pseudo-element selector ('::before') is used

Demo

https://specificity.keegan.st/

CSS Specificity in PMG

Use class, attribute, and pseudo class selectors as much as possible for general styling. Lower the specificity the better.

```css
.button {
    font-size: 16px;
}
```

Use ID selectors for specific styling - for example overriding styles for certain component

```css
#widget-it-container .button {
    font-size: 20px;
}
```

CSS Specificity hack

```css
.button {
    color: red;
}
/* Higher specificity */
.button.button {
    color: blue;
}
```

Avoid using `!important` as it is usually not necesssary.

Responsive Web Design (RWD)

Mobile first

Simple: Always start with styles for smallest possible screen that needs to be supported (mobile, table, desktop).

```css
/* full width column on mobile */
.layout {
    width: 100%;
    margin-left: 16px;
    margin-right: 16px;
}

/* 2 columns on tablet */
@media (min-width: 768px) {
    display: grid;
    grid-template-columns: 1fr 1fr;
}

/* 4 columns on desktop */
@media (min-width: 1024px) {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
}
```

Why mobile first?

- To keep it consistent and intuitive
- Mobile styles looks OK on desktop but desktop styles on mobile aren’t
- Overriding desktop styles to mobile is usually harder
  - try to revert css styles from previous slides for mobile only

CSS Grid

What is CSS Grid

Allows to display elements in a grid layout consisting of rows and columns. Similar to tables but more flexible and powerful.

Only few of possible options:

- gap between elements
- making all elements the same size
- auto-fitting size based on available space
- areas to move elements around

Examples and patterns

It is very complex subject and requires practice to master.

https://gridbyexample.com/

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout

Grid systems

Bootstrap’s Flexbox grid system != Bootstrap CSS grid

Bootstrap Flexbox Grid system

- Uses flexbox to simulate grid in one dimension (columns - multiple lines still allowed).
- Supports gutters
  - gaps between column content and edges of the column
- Supports responsive breakpoints
- Shortcut to expose grid-like capabilities as utility classes with RWD

When to use Bootstrap’s flexbox grid systems

Probably never - Use CSS grid system instead

When to use Bootstrap’s CSS Grid system

- For big layouts (sidebars, main content)
- Using many classes becomes unreadable therefore fallback to css grid
- Don’t bother with components requiring pixel perfect requirements or if can be solved simpler with pure css

How about Flex?

Still very useful for one-dimensional layouts (rows or columns) but not for two-dimensional layouts (rows and columns).

Good practices

Button vs link

- Link `<a>`
  - navigating to other pages
  - opening in a new tab
- Button `<button>`
  - performing action at a page that does not require navigation
  - submitting a form

Button vs link

Don’t do it

```html
<a href="javascript:void(0)">Action</a>
```

Do this instead

```html
<button>Action</button>
```

Devtools is your friend

- Modifying styles in browser
- Use computed tab
- Enforce element state

Questions?