Session 10

Mohamed Emary

April 7, 2024

1 Psuedo Classes

Pseudo classes are used to define a special state of an element. They have the same specificity as regular classes.

- `:first-child` - Selects the first child of an element
- `:last-child` - Selects the last child of an element
- `:nth-child(n)` - Selects the nth child of an element
- `:not(selector)` - Selects all elements that do not match the given selector

The order of the `nth-child` is relative to its siblings inside the parent element.

```
<div class="parent">
  <div class="child">1</div>
  <div class="child">2</div>
  <div class="child">3</div>
  <div class="child">4
    <div class="child">1</div>
    <div class="child">2</div>
  </div>
</div>
```

Using `.child:first-child` here will select the first child of the parent element.

Using `.child:nth-child(2)` here will select the second child of the parent element, and the second child of the 4th child of the parent element. `.child` here specifies the class of the element we want to select.

Using `.child:nth-child(2n)` will select all even children of the parent element, and using `.child:nth-child(2n+1)` will select all odd children of the parent element.

The `:nth-child` pseudo class has some compatibility issues with older browsers.

`:not(.item)` will select all elements that do not have the class `item`.

2 CSS Combinators

`.item > p` - Selects all `<p>` elements that are direct children of a `<div>` element

```
<div class="item">
  <p>First paragraph</p>      // Will be selected
  <div>
    <span>
      <p>Second paragraph</p> // Will not be selected
    </span>
  </div>
</div>
```

`.item + p` - Selects the first `<p>` element that is placed immediately after an `.item` element

```
<div>
  <div class="item"></div>
  <p>Second paragraph</p> // Will be selected
  <p>First paragraph</p>  // Will not be selected
</div>
```

`.item ~ p` - Selects all `<p>` elements that are siblings of an `.item` element

```
<div>
  <div class="item"></div>
  <p>Second paragraph</p> // Will be selected
  <p>Second paragraph</p> // Will be selected
</div>
```

using `*` will select all elements.

So in summary:

- `>` Works wil direct children
- `+` Works with the first sibling
- `~` Works with all siblings

  Note that the space is a combinator as well, and it selects all descendants. for example `.item p` will select all `<p>` elements that are descendants of an `.item` element.

3 Some Extra Selectors

3.1 Attribute Selectors

We can select elements based on their attributes.

- `[attribute]` - Selects all elements with a specific attribute
- `[attribute=value]` - Selects all elements with the specified attribute and value
- `div[class]` - Selects all `<div>` elements with a class attribute
- `div[class^="it"]` - Selects all `<div>` elements whose class attribute value begins with `it`. So for example, it will select `<div class="item">` or `<div class="item-1">` and so on.

- `div[class$="em"]` - Selects all `<div>` elements whose class attribute value ends with `em`. So for example, it will select `<div class="item">` or `<div class="problem">` and so on.
- `div[class*="it"]` - Selects all `<div>` elements whose class attribute value contains the substring `it`. So for example, it will select `<div class="item">` or `<div class="visited">` and so on.

4 Extra CSS Properties

4.1 Scroll Behavior

`scroll-behavior: smooth;` - Adds a smooth scrolling effect to the page

It can be used with an anchor tag to scroll to a specific part of the page smoothly.

4.2 Object Fit

Since images are replaced elements they can be styled using the `object-fit` property.

The default value is `fill`, which will stretch the image to fit the container.

Figure 1: fill

We can change it to `contain` to make the image fit the container without stretching it. But it will look something like this:

Figure 2: contain

Using `object-fit: cover;` will make the image cover the container without stretching it. It will look something like this:

Figure 3: cover

Using `object-fit: none;` will add the image with its original size without stretching it. It will look something like this:

Figure 4: none

Using `object-fit: scale-down;` will compare between `contain` and `none` and will use the smaller one. Our image is larger than the container so it will look the same as `contain`.

Figure 5: scale-down

4.3 Nesting Selectors

We can nest selectors in CSS. For example:

```
.parent {
  color: red;
  .child {
    color: blue;
    p{
      font-family: Arial;
    }
    &:hover {
      color: green;
    }
  }
}
```

This will make the text color of the `.parent` element red, and the text color of the `.child` element inside the parent element blue.

We can have nested selectors inside nested selectors. For example, the `p` element inside the `.child` element will have the font family of Arial.

The `&` symbol is used to refer to the parent selector. So `&:hover` will select the parent element when it is hovered which is the `.child` element in this case and change its text color to green.

4.4 Important Property

We can use the `!important` property to override any other style. It has the highest specificity even higher than inline styles.

```
p {
  color: red !important;
}
```

This will make the text color of `p` element red even if it has another color specified in the inline style.

If two same selectors have the `!important` property, the one that comes later will override the previous one. (The last one will be applied)

```
p {
  color: red !important;
}
p {
  color: blue !important;
}
```

This will make the text color of `p` element blue.

4.5 Inherit and Initial

Some properties are inherited by default. For example, font family, font size, and text color are inherited. So if we specify the font family for the `body` element, all the text inside the `body` element will have the same font family.

However, other properties like background color are not inherited, so we have to specify them for each element.

We can use the `inherit` keyword to make a property inherit the value of its parent element.

```
.parent{
  padding-top: 20px;
}

.child{
  padding-top: inherit;
}
```

The child element will inherit the `padding-top` value of the parent element, so it will have a padding top of 20px.

The `initial` keyword resets the property to its default value if it's inherited. For example, if we have a `color` property in the `body` element, and we want to reset it to its default value in the `p` element, we can use the `initial` keyword.

```
body{
  color: red;
}

p{
  color: initial;
}
```

The text color of the `p` element will be the default color which is black.

5 Solving Compatibility Issues

5.1 CSS

With CSS3, alot of new features were added, and some of them are not supported by older browsers. So we have to make sure that our website is compatible with all browsers.

To solve compatibility issues, we need to used prefixes for some properties.

- `-webkit-` for Chrome, Safari, and modern Opera
- `-moz-` for Firefox
- `-ms-` for Internet Explorer
- `-o-` for Opera

For example, `linear-gradient` is a CSS3 property that is not supported by all browsers. So we have to add prefixes for it.

```
background: -webkit-linear-gradient(red, blue);
background: -moz-linear-gradient(red, blue);
background: -ms-linear-gradient(red, blue);
background: -o-linear-gradient(red, blue);
background: linear-gradient(red, blue);
```

And it's important to add the **unprefixed version at the end**, so that the browser will use it if it supports it.

Compatibility issues are browser-specific, not web-engine-specific. For example, Chrome and Opera use the same engine, but they have different compatibility issues.

We can use a tool like Autoprefixer to add prefixes automatically to our CSS code.

6 Semantic HTML

HTML5 Introduced semantic tags like:

- `<header>` - For page header or section header
- `<footer>` - For page footer or section footer
- `<section>` - For Sections of a page
- `<main>` - For the main content of a page (Can't be repeated)
- `<nav>` - For navigation bar
- `<mark>` - For highlighted text
- `<figcaption>` - For figure caption
- `<article>` - For articles or blog posts
- `<aside>` - For content aside from the content it is placed in

Semantic tags like `<header>` have meaning, while non-semantic tags like `<div>` don't have meaning.

These elements are not supported by older browsers.

They are important for SEO and accessibility, so we have to make sure that they are supported by all browsers.

7 Font Awesome Icons

Sometime we need to add icons to our website, icons like phone, map, email, logo, etc. We can use images for that, but images are not scalable and they are larger in size which is not good for the website performance, So to solve this issue we can use an icons library like font awesome.

Font Awesome is a library of vector graphics icons that we can use in our website.

Vector graphics are images that can be scaled to any size without losing quality. They are also smaller in size than raster images so it will not affect the website performance.

It allows you to use icons like these:

To use font awesome we can use any of the following two methods:

Method 1: Using CDN

Visit font awesome page on cdnjs and copy the link of the `all.min.css` file with the version you want of font awesome library.

Then link it to your page as a stylesheet like below:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax
↪  /libs/font-awesome/6.5.2/css/all.min.css">
```

This will add **all font awesome icons**, but if you want to add only some icons, for example, brands icons only you can copy its link from cdn too like this:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax
↪  /libs/font-awesome/6.5.2/css/brands.min.css">
```

Method 2: Downloading the files

1. Download the font awesome compressed file from here then extract it.
2. Copy the `all.min.css` file from the `css` folder and paste it in your project CSS folder.
3. Copy the `webfonts` folder from the extracted folder and paste it in your project folder.
4. Add the following link to your HTML file:

```
<link rel="stylesheet" href="./css/all.min.css">
```

After completing any of the above methods, we can use font awesome icons in our HTML file.

To use a font-awesome icon like , we have to use the `<i>` tag with the `fa` class and the icon name class.

```
<i class="fa-brands fa-apple"></i>
```

The `fa-brands` class is used to specify the category of the icon, and the `fa-apple` class is used to specify the name of the icon.

The font-awesome icons are treated as text, so we can style them using text properties like `color`, `font-size`.

We can even add `hover` effects to them.

```
.fa-apple {
  color: blue;
  font-size: 30px;
}
.fa-facebook:hover {
  color: darkblue;
}
```

You can resize them using classes too like `fa-lg`, `fa-2x`, `fa-3x`, `fa-4x`, `fa-5x`, `fa-6x`, `fa-7x`, `fa-8x`, `fa-9x`, `fa-10x`. Some of these classes make icons appear inline, while others make them appear as block.

To give our icons the same width, we can use the `fa-fw` (**F**ont-**A**wesome **F**ixed **W**idth) class.

You can also animate icons using some classes like `fa-beat`, `fa-spin`, `fa-pulse`.

There a lot more you can do like stacking icons, making icons appear inline or block, and more. You can see all that in the font-awesome documentation from here

8 Summary

1. **Pseudo-Classes:**
   - They are used to target specific element states, not just element types.
     - Examples: `:first-child` (selects first child element), `:last-child` (selects last child element), `:nth-child(n)` (selects nth child element), `:not(selector)` (selects elements not matching a selector).
2. **CSS Combinators:**
   - They are used to combine selectors and target elements based on their structure in the HTML document.
     - `>` (greater than): selects direct children of an element.
     - `+` (plus): selects the first sibling element immediately after a specific element.
     - `~` (tilde): selects all sibling elements of a specific element.
     - Space: selects all descendant elements.
3. **Attribute Selectors:**
   - They are used to target elements based on their attribute presence or specific attribute values.
     - `[attribute]`: selects elements with a specific attribute.
     - `[attribute=value]`: selects elements with a specific attribute and value.
     - Selectors exist for elements starting with `^=`, ending with `$=`, or containing `*=` a specific value.
4. **Extra CSS Properties:**
   - `scroll-behavior: smooth;`: creates a smooth scrolling effect on the page.
   - `object-fit`: controls how images are displayed within their container, it can be:
     - `contain`
     - `cover`
     - `none`
     - `scale-down`
5. **Nesting Selectors:**
   - Allows for targeting styles based on the element hierarchy in HTML.
   - The `&` symbol within nested selectors refers to the parent selector.
6. **Important Property:**
   - `!important`: overrides any other style rule, even inline styles, due to its high specificity.
7. **Inherit and Initial:**
   - Properties can be inherited from parent elements or reset to their default values.
     - `inherit`: makes a property inherit the value from its parent element.
     - `initial`: resets a property to its default value.
8. **Solving Compatibility Issues:**

- To solve compatibility issues of some CSS3 properties we can use prefixes:
  - Prefixes are introduced for properties not supported by older browsers (`-webkit-`, `-moz-`, `-ms-`, `-o-`).
  - Tools like Autoprefixer can automate adding prefixes.

9. Semantic HTML:

- HTML tags with meaning used for better SEO and accessibility.
  - Examples: `<header>`, `<footer>`, `<section>`, `<main>`, `<nav>`, etc.

10. Font Awesome Icons:

- Explains how to integrate Font Awesome, a library of vector graphics icons, into your website.
  - Methods for including Font Awesome using a CDN or downloading files.
  - How to use Font Awesome icon classes within HTML tags.
  - Styling and customizing Font Awesome icons with CSS.