Session 3

Mohamed Emary

March 10, 2024

Review & Questions

In the first part of the session, eng. Shimaa reviewed the previous sessions and asked some questions to the students to make sure that they understand the previous sessions.

CSS

CSS stands for *Cascading Style Sheets*. It is a style language used for describing the look and formatting of a document written in HTML.

General look of a CSS *Rule*

```
selector {
  property: value; /* declaration */
}
```

Where should CSS code be? & How to link it?

CSS code can be placed in three different locations:

- **Inline** in the HTML element inside the `style` attribute.
- **Internal** in the head section of the HTML document in a separate `<style>` tag.
- **External** in a separate file linked to the HTML document using the `<link>` tag.

Inline CSS

You can apply CSS to an HTML element using the `style` attribute.

```
<p style="color: red;">This is a paragraph.</p>
```

Internal CSS

You can write CSS inside the head section of the HTML document using the `<style>` tag inside the `<head>` element of the HTML document.

```
<head>
  <style>
    p {
      color: red;
    }
  </style>
</head>
```

External CSS file

You can link an external CSS file to an HTML document using the `<link>` tag inside the `<head>` element of the HTML document.

```
<head>
  <link rel="stylesheet" href="css/style.css">
</head>
```

In the example above `css/style.css` is the path of the CSS file in your project. You should create a folder called `css` and put the CSS file inside it with a name of your choice for example `style.css`.

Why to separate CSS from HTML?

There are multiple reasons to separate the CSS from the HTML:

- **Maintainability**: It is easier to maintain and update the code when the HTML and CSS are separated.
- **Reusability**: You can use the same CSS file for multiple HTML files.
- **Performance**: The browser can cache the CSS file and use it for multiple pages.

If we have different styles for the same element, what would happen?

If you have different styles for the same element, the style that is defined last will be applied.

```
<head>
  <style>
    p {
      color: red;
    }
  </style>

  <!-- In this linked file, the color of p is blue -->
  <link rel="stylesheet" href="css/style.css">
</head>
```

In the example above the color of the paragraph will be blue not red, because the linked file is defined after the internal style.

Selectors

Selectors are used to select the HTML elements that you want to style only.

Tag

To select an HTML element, you can use the tag name of that element, for example to select the paragraph tag you should use `p` in the CSS file.

Example:

HTML:

```html
<p>This is a paragraph.</p>
```

CSS:

```css
p {
  color: red;
}
```

Class

To select an HTML element, you can use the class name of that element, and to use the class in the CSS file you should use a dot `.` before the class name for example if you have a class called `intro` you should use `.intro` in the CSS file.

Example:

HTML:

```html
<p class="intro">This is a paragraph.</p>

<p>This is not red.</p>
```

CSS:

```css
.intro {
  color: red;
}
```

In the example above, the color of the first paragraph will be red but the second paragraph will not be affected.

Some guidelines to follow when using class:

1. You can't use spaces in class names, but you can use a hyphen `-` or an underscore `_` to separate the words.
2. You should also use a descriptive name for the class to make it easier to understand the code.

To give an element multiple classes you can separate them with a space inside the same `class=""` attribute, for example `class="intro text-center"`, but you can't use the class attribute more than once in the same element.

Classes are also used to reduce code repetition so if you want to apply the same style to multiple elements you can give them the same class.

ID

To select an HTML element, you can also use the id of that element, and to use the id in the CSS file you should use a hash `#` before the id name for example if you have an id called `intro` you should use `#intro` in the CSS file.

The difference between the class and the id is that the class can be used for multiple elements but the id should be **unique** in the HTML document.

Grouping

You can group multiple elements to apply the same style to them using a `div` element.

Example:

HTML:

```
<div class="intro">
  <h1>This is a heading.</h1>
  <p>This is a paragraph.</p>
</div>
```

CSS:

```
.intro {
  color: red;
}
```

What if you want to apply a style to an element only if it is *inside* a specific element? (Nested Selectors)

You can use the space to select an element only if it is inside another element. For example if you want to apply a style to the paragraph only if it is inside a `div` with a class `intro` you can use `.intro p` in the CSS file.

What if you want to apply a style to an element with a specific class only? Or to an element with multiple classes?

To apply a style to a paragraph only if it has a class `intro` you can use `p.intro`, and to apply a style to an element only if it has both the classes `intro` and `center` you can use `.intro.center` without a space between the classes names.

What if you apply a style to multiple classes?

You can use a comma `,` to apply the same style to multiple classes, for example `.intro, .center` will apply the same style to the elements with the class `intro` or the class `center`.

Specificity

Specificity is the means by which browsers decide which CSS property values are the most relevant to an element and, therefore, will be applied.

The following list of selector types is by increasing specificity:

1. Universal selectors (e.g., `*`)
2. Type selectors (e.g., `h1`)
3. Class selectors (e.g., `.example`)
4. ID selectors (e.g., `#example`)

The rules above also applies when combining multiple selectors in the same rule, for example if you have a rule with a tag and a class, the class will be more specific than the class only.

Some Styling Properties

The default value for `height` is `auto` and for `width` is `100%`.

If you want your styling to be dynamic and responsive, you should use a relative unit like `%`, for example `width: 100%` will make the width of the element 100% of the width of its parent element.

Block & Inline Elements

1. **Block-level elements** start on a new line and take up the full width available.

Example block elements are `<div>`, `<h1>` to `<h6>`, `<p>`, `<form>`, `<header>`, `<footer>`, `<section>`, and `<ul>`.

2. **Inline elements** do not start on a new line and only take up as much width as necessary.

Example inline elements are `<a>`, `<span>`, `<img>`, `<label>`, `<input>`, `<strong>`, `<em>`, and `<b>`.

You can change whether an element is block or inline using the `display` property. For example, you can change a `<div>` to an inline element using `display: inline;` or an `<a>` tag to a block element using `display: block;`.

With inline elements `width` and `height` properties have no effect.

3. **Inline-block** elements are similar to inline elements, but they can still have width and height.

Replaced Elements

How can `<img>` element be inline and still have width and height?

This is because the `<img>` element is **replaced inline element**.

Replaced elements can be given explicit width and height values using the `width` and `height` properties. This allows you to control the size of the element, regardless of its content.

A replaced element in HTML is an element that is replaced with another element, such as an image, a video, or an audio file. Replaced elements are not rendered in the same way as other HTML elements, and they do not have the same properties or behaviors. Replaced elements are used to embed content that cannot be created with HTML.

The most common replaced elements are:

- `<img>` : Inserts an image into the document.
- `<video>` : Inserts a video into the document.
- `<audio>` : Inserts an audio file into the document.
- `<iframe>` : Inserts a frame into the document.
- `<input>` : Inserts an input field into the document.

You can set the width and height of an `<img>` element using the `width` and `height` attributes in the HTML, or you can use CSS. Here's an example:

```
<img src="image.jpg" alt="An image" style="width: 200px; height:
↪  200px;">
```

In this example, the image will be displayed as a `200px` by `200px` square, regardless of the actual dimensions of image.jpg.

Summary