UPC Alumni

My guide to Website building

Oriol Farrés / oriol.farres@estudiantat.upc.edu

Maig del 2024

Oriol Farrés / oriol.farres@estudiantat.upc.edu

May 2024

Contenido

Oriol Farrés / oriol.farres@estudiantat.upc.edu

May 2024

1.Introductory Overview

1.1. Necessary Files for Crafting a Website

To start building a website, you'll need three essential files:

- index.html (HTML file)
- style.css (CSS file)
- main.js (JavaScript file)

1.2.Index (HTML)

HTML (Hypertext Markup Language) is the standard markup language for creating web pages. It provides the structure and content of your web pages, defining elements such as headings, paragraphs, lists, links, and images.

Template code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Name of the Web</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
      //code here
</body>
</html>
```

1.3.Style (CSS)

CSS (Cascading Style Sheets) is a stylesheet language used to define the presentation and layout of HTML documents. It allows you to style the appearance of elements, apply colors, fonts, spacing, and create responsive designs.

Template code:

```css
* {
    margin: 0;
    padding: 0;
    font-family: 'Poppins', sans-serif;
    box-sizing:content-box;
}

body {
    margin: 0; /* Reset default margin */
    padding: 0; /* Reset default padding */
    width: 100vw; /* Set the width to 100% of the viewport width */
    height: 100vh; /* Set the height to 100% of the viewport height */
    background-image: url(""); /* Set the background image */
    background-size: 100% 100%; /*Background image cover whole element */
    background-position: center;
    color: #ffffffda; /* Set the text color */
}
```

1.3.Main (JavaScript)

JavaScript is a programming language that enables interactive and dynamic functionality on web pages. It allows you to add behavior to your HTML elements, handle user interactions, manipulate the DOM (Document Object Model), and fetch data from external sources.

2.HTML

2.1. How to structure the code

Contrary to languages such as C++, HTML does not typically involve the use of methods or multiple files. Consequently, the presentation aspect of HTML becomes paramount. In my initial project, I structured the body of the document into 3 distinct sections:

- Header
- Main
- Footer

In HTML, the <header>, <main>, and <footer> elements primarily serve to organize and structure the content of a webpage. While they do not inherently perform any actions or functions, they play a crucial role in providing clarity and semantic meaning to the markup.

These are tecnhically named "***Semantic Data***" and there are also the <nav>, <section>, <article>, <aside>.

Template code:

```
<body>

    <header>
      //code here
    </header>


    <main>
      //code here
    </main>


    <footer>
      //code here
    </footer>
</body>
```

2.2. Data types

In HTML, data types refer to the various types of content that can be used to represent information on a webpage. HTML provides several elements for structuring and presenting different types of data. Here's a list of some common data types in HTML along with corresponding HTML elements.

2.2.1. Text Data

- `<p>`: Represents a paragraph of text.
- `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`: Heading elements for different levels of headings.
- `<span>`: Defines a section of text that can be styled separately from the rest of the content.

2.2.2. List Data

- `<ul>`: Represents an unordered list.
- `<ol>`: Represents an ordered list.
- `<li>`: Represents a list item within `<ul> (unordered list)` or `<ol> (ordered list)`.

2.2.3. Link Data

- `<a>`: Defines a hyperlink, linking to another webpage or resource.

2.2.4. Image Data

- `<img>`: Represents an image.
- `<figure>` and `<figcaption>`: Used to group images and their captions.

2.2.5. Table Data

- `<table>`: Represents tabular data.
- `<tr>`: Represents a row in a table.
- `<th>`: Represents a header cell in a table.
- `<td>`: Represents a data cell in a table.

2.2.6. Form Data

- `<form>`: Represents a form for collecting user input.
- `<input>`: Represents an input control (text input, checkbox, radio button, etc.).
- `<textarea>`: Represents a multiline text input control.
- `<select>`: Represents a dropdown list.
- `<button>`: Represents a clickable button.

2.2.7. Multimedia Data

- `<audio>`: Represents sound content.
- `<video>`: Represents video content.
- `<iframe>`: Represents an embedded HTML document within the current document.

2.2.8. Metadata Data

- `<meta>`: Provides metadata about the HTML document (e.g., character set, viewport settings).

2.3. How to comment in HTML

To comment in HTML you use this notation: "<! your text here >!

2.4. How to link in HTML

To link in HTML you use this notation `<a href="index.html">TEXT HERE</a>`.

Linking to other pages (HTML) is allowed, as well as linking to public URLs.

Oriol Farrés / oriol.farres@estudiantat.upc.edu

3.CSS

3.1. Positioning Text

In CSS, positioning text elements on a webpage involves the use of various properties and techniques to control their placement, spacing, and alignment. Two key concepts in text positioning are padding and margin.

3.1.1 Pixels (px) vs percentage (%)

3.1.1.1 In Padding

Pixels: When you specify padding using pixels, you define an absolute measurement. The padding will be a fixed size regardless of the size of the containing element.

Percentage: When you specify padding using percentage values, the padding is calculated as a percentage of the width of the containing block element. The padding adjusts dynamically based on the size of the containing element.

3.1.1.2 In Margin

Pixels: Margin defined in pixels creates a fixed amount of space around the element, independent of the containing block's size.

Percentage: Margin specified in percentage values calculates the margin as a percentage of the width of the containing block element. The margin adjusts dynamically based on the size of the containing element.

3.1.2 Padding

Padding refers to the space between the content of an element and its border. It provides internal spacing within an element, pushing the content away from its edges.

1 Parameter: It applies the same padding to all sides of the element

e.g: `padding: 10px;`

2 Parameters: 1st value-> top&bottom, 2nd value->left&right

e.g: `padding: 10px 20px;`

3 Parameters: 1st value-> top, 2nd value->left&right, 3rd value-> bottom

e.g: `padding: 10px 20%, 15px;`

4 Parameters: 1st value-> top, 2nd value-> right, 3rd value-> bottom, 4th value-> left

e.g: `padding: 10%, 20px, 10px, 22%;`

3.1.3 Margin

Padding refers to the space between the content of an element and its border. It provides internal spacing within an element, pushing the content away from its edges.

1 Parameter: It applies the same padding to all sides of the element

e.g: `padding: 10px;`

2 Parameters: 1st value-> top&bottom, 2nd value->left&right

e.g: `padding: 10px 20px;`

3 Parameters: 1st value-> top, 2nd value->left&right, 3rd value-> bottom

e.g: `padding: 10px 20px, 15px;`

4 Parameters: 1st value-> top, 2nd value-> right, 3rd value-> bottom, 4th value-> left

e.g: `padding: 10px 20px, 10px, 22px;`

3.1.4 MY RECOMENDATION

Using REM and and padding-left/right/bottom/top.

3.1.4.1 What's REM?

The `rem` unit stands for "root em". It is a relative unit of measurement that is based on the font size of the root element (`<html>`).

Here's how `rem` units work:

- **Base font size:**
  The `rem` unit is calculated based on the font size of the root element (`<html>`).
  By default, the root font size is usually set to `16px` in most browsers.
- **Relative to Root Font Size:**
  When you specify a length or size using `rem`, it is relative to the font size of the root element.
  For example, if the root font size is `16px`, `1rem` is equal to `16px`.
- **Scalability:**
  One advantage of using `rem` units is that they are scalable and adjust automatically when the root font size changes.
  If you change the font size of the root element, all `rem` units on the page will scale accordingly.

3.1.5 Position value

The `position` property in CSS is used to control the positioning of elements within a web page layout. It allows developers to specify how an element should be positioned relative to its containing element or the viewport. Here's an overview of how the position property is commonly used:

Static Positioning:

Default value. Elements are positioned according to the normal flow of the document.

Relative Positioning:

Positioned relative to its normal position in the document flow.

Absolute Positioning:

Removed from the normal document flow.

Fixed Positioning:

Positioned relative to the viewport. Remains fixed in its position even when the page is scrolled. Does not affect the layout of other elements. Often used for creating headers, footers, or navigation bars that remain visible at all times.

Sticky Positioning:

Positioned based on the user's scroll position. Acts like relative positioning until the element reaches a specified point, then it "sticks" in place like fixed positioning. Useful for creating elements such as sticky headers or sidebars that stick to the top or side of the viewport as the user scrolls.

3.2. Types of CSS selectors

CSS selectors are patterns used to select and style elements in HTML documents. They enable you to target specific elements or groups of elements based on their attributes, relationships with other elements, or their position in the document structure.

corresponding HTML elements.

3.2.1. Classes

- ‘.’

Class selectors target elements with a specific class attribute. Classes allow you to apply the same styles to multiple elements, promoting code reusability and maintainability.

3.2.2. IDs

- ‘#’

ID selectors target a single element with a specific ID attribute. IDs should be unique within the HTML document and are often used to apply styles to unique elements or sections.

3.2.3. Type Selectors

Type selectors target elements based on their HTML tag names. They allow you to apply styles to all elements of a specific type. Type selectors are not prefixed with any symbol and are useful for applying global styles to elements like <p>, <h1>, , etc.

3.2.4. Universal Selectors

The universal selector selects all elements in the document. It is useful for resetting default styles or applying global styles.

3.2.5. Attribute Selectors

Attribute selectors allow you to target elements based on the presence or value of their attributes. They are enclosed in square brackets ([]) and can have various matching options, such as exact match, partial match, etc.

3.2.6. Pseudo-classes and Pseudo-elements

Pseudo-classes target elements based on their state or position, such as:

- :hover
- :focus
- :first-child

Pseudo-elements target specific parts of an element's content, like:

- ::before
- ::after
- ::first-line

3.3. How to comment in css

To comment in css you use this notation: “/* text here */”

4.ANNEX

4.1 Full HTML template

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Oriol Farrés Portfolio</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <header>
        //code here
    </header>

    <main>
        //code here
    </main>

    <footer>
        //code here
    </footer>
</body>
</html>
```