Guillaume T. 04-2024

CSS

WEB
2 - HTML & CSS

Document summary

The text covers fundamental educational objectives in web development, elucidating the roles of HTML and CSS, HTML document structure, and the integration of CSS styling. It emphasizes the application of HTML elements, CSS properties, and the importance of understanding the Document Object Model (DOM). Additionally, it delves into advanced layout techniques like Grid and Flexbox, alongside responsive design using Media Queries. The narrative underscores the significance of accessibility and SEO, elucidating various methods to enhance SEO practices. Furthermore, it provides comprehensive insights into CSS anatomy, selectors, properties, layout models like Flexbox and Grid, and the implementation of Media Queries for responsive design.

Table of content

1. Educational objectives 3
2. Anatomy of CSS document 4
  2.1. Ruleset 4
  2.2. Declaration 4
3. Adding CSS to HTML 5
  3.1. Inline 5
  3.2. Internal 5
  3.3. External 5
4. CSS Selectors 6
  4.1. Type 6
  4.2. Id 6
  4.3. Class 6
  4.4. Universal 6
  4.5. Grouping 6
  4.6. Descendant 6
  4.7. Child 6
  4.8. Other selectors 7
  4.9. Precise selectors 7
5. Pseudo-classes and pseudo-elements 8
  5.1. Pseudo-class 8
  5.2. Pseudo-element 8
6. CSS Properties 9
  6.1. Text properties 9
  6.2. Background properties 9
  6.3. Color properties 9
  6.4. Font properties 9
    6.4.1. External font 9
  6.5. Properties and Inheritance 9

7. Layout 11
   7.1. Positioning 11
   7.2. Flexbox 12
   7.3. Grid 12
8. Media Queries 14
   8.1. Define variables 14
   8.2. Accessibing variables 14

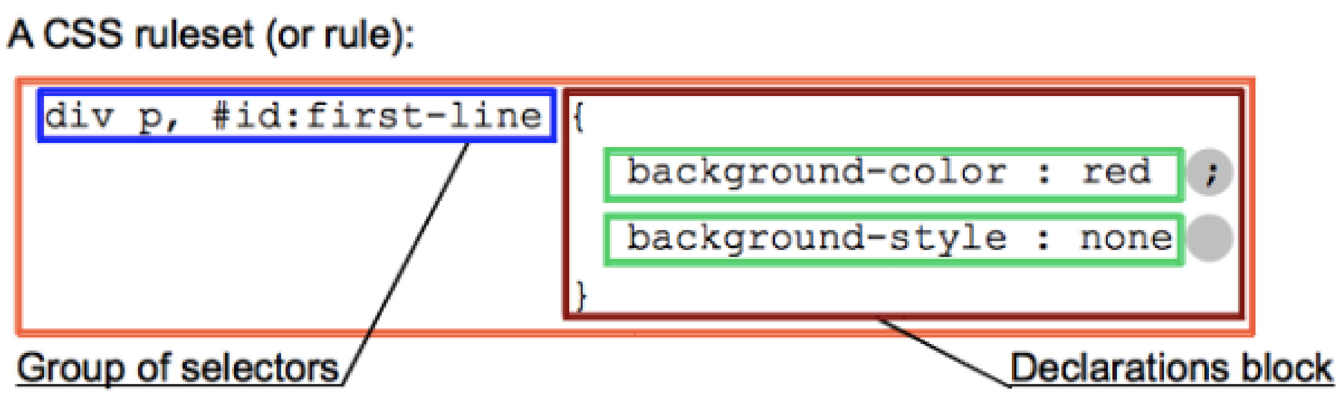
1. Educational objectives

- Describe the role of HTML and CSS in web development
- Describe the structure of an HTML document
- Create a HTML document with a CSS stylesheet
- Use HTML elements and CSS properties
- Describe the role of the Document Object Model (DOM) and it’s structure
- Use Grid and Flexbox for layout
- Use Media Queries for responsive design
- Cite the importance of accessibility and SEO (Search Engine Optimization) in web development
- Cite different ways to enhance SEO

2. Anatomy of CSS document

2.1. Ruleset

A CSS document is a sequence of rulesets, used for styling specified parts of the document.



A ruleset is composed of

- a comma-separated list of selectors, and
- a declarations block describing the styling of the selected elements.

Rulesets apply declarations to specific parts of the document. Declaration blocks are preceded by one or more comma-separated selectors. Selectors are conditions selecting some elements of the page.

2.2. Declaration

A declaration is a property-value pair that describes the styling of the selected elements. A declaration block is a sequence of declarations enclosed in curly braces.

```
selector {
  property: value;
}
```

3. Adding CSS to HTML

CSS can be added to an HTML document in three ways:

3.1. Inline

Inline CSS is added directly to the HTML element using the `style` attribute.

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

3.2. Internal

Internal CSS is added to the HTML document using the `<style>` tag in the `<head>` section.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: red;
    }
  </style>
</head>
<body>
  <p>This is a red paragraph.</p>
</body>
</html>
```

3.3. External

External CSS is added to the HTML document using the `<link>` tag in the `<head>` section.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <p>This is a red paragraph.</p>
</body>
</html>
```

4. CSS Selectors

CSS selectors are used to select the elements to which the ruleset will apply. There are several types of selectors:

4.1. Type

Type selectors select elements based on their tag name.

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

This rule will apply to all **`<p>`** elements. in the html page.

4.2. Id

Id selectors select elements based on their `id` attribute.

```css
#myId {
  color: red;
}
```

This rule will apply to the element with the **`id="myId"`** attribute.

4.3. Class

Class selectors select elements based on their `class` attribute.

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

This rule will apply to all elements with the **`class="myClass"`** attribute.

4.4. Universal

The universal selector `*` selects all elements.

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

This rule will apply to all elements in the html page.

4.5. Grouping

Grouping selectors select multiple elements.

```css
h1, h2, h3 {
  color: red;
}
```

4.6. Descendant

Descendant selectors select elements that are descendants of another element.

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

This rule will apply to all **`<p>`** elements that are descendants of a **`<div>`** element.

4.7. Child

Child selectors select elements that are direct children of another element.

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

This rule will apply to all **`<p>`** elements that are direct children of a **`<div>`** element.

Exemple

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div > p {
      color: red;
    }
  </style>
</head>
<body>
  <div>
    <p>This is a red paragraph.</p>
    <span>
      <p>This isn't a red paragraph.</p>
    </span>
  </div>
</body>
</html>
```

4.8. Other selectors

- a b selects all elements that match b and are (potentially non-direct) descendants of elements that match a.
- a > b selects all elements that match b and are direct children of elements that match a.
- a   b selects all elements that match b and are siblings of elements that match a (i.e. both a and b have the same parent) and b appears after a.
- a + b selects all elements that match b and are adjacent siblings of elements that match a (i.e. both a and b have the same parent, and b immediately follows a).

4.9. Precise selectors

The id, class and attribute selectors can be combined by adding them as suffixes to other selectors.

```
p#myid                    /* a p element with id "myid" */
*.myclass                 /* any element with class "myclass" */
p.myclass#myid[attr=value] /* a p with class "myclass", id "myid" and "attr=value" */
```

5. Pseudo-classes and pseudo-elements

5.1. Pseudo-class

Pseudo-classes are used to define the special state of an element.

```css
a:link {
  color: blue;
}
```

This rule will apply to all **<a>** elements that are unvisited links.

5.2. Pseudo-element

Pseudo-elements are used to style parts of an element.

```css
p::first-line {
  color: red;
}
```

This rule will apply to the first line of all **<p>** elements.

6. CSS Properties

CSS properties are used to describe the styling of the selected elements. There are several types of properties:

6.1. Text properties

```css
p {
    text-align: center;
    line-height: 2em;
    letter-spacing: 2em;
    text-decoration: underline;
    text-transform: uppercase;
}
```

6.2. Background properties

```css
body {
    background-color: blue;
    background-image: url('../image.jpg');
    background-repeat: no-repeat;
    background-size: auto;
}
```

6.3. Color properties

```css
p {
    /* named-color values */
    color: red;
    color: orange;

    /* hex-color values */
    color: #090;
    color: #009900;

    /* rgb() values, rgba() also has alpha (opacity) */
    color: rgb(34, 12, 64);
    color: rgba(34, 12, 64, 0.6);
}
```

6.4. Font properties

```css
p {
    font-family: Arial, sans-serif;
    font-size: 16px;
    font-weight: bold;
    font-style: italic;
}
```

The font property sets an element’s font to a system font.

- The first font specified that is available is used to display the element.
- The recommended font size units are px, em and %. Pixels are fixed, wheareas em and % are relative to the font size specified for the document.

6.4.1. External font

```css
@import url('https://fonts.googleapis.com/css?family=Roboto&display=swap');

body{
    font-family: 'Roboto', sans-serif;
}
```

6.5. Properties and Inheritance

Some properties are inherited by child elements.

- Do Inherit: color, font, line-height, letter-spacing, text-align, etc.
- Do not Inherit: background-color, border, padding, margin, etc.

7. Layout

The display property specifies if/how an element is displayed.

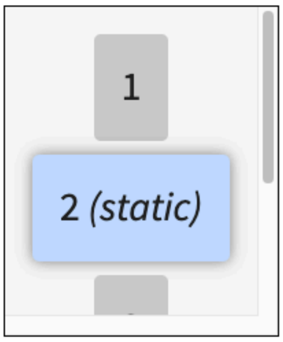
The default value of display depends on the element. The most common values are:

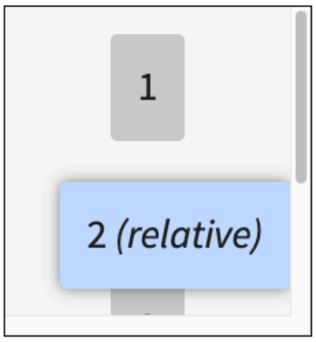
- block: element generates line break before and after itself. Can have specified height and width. If unspecified, takes up the whole width of its parent element (e.g. div, p, h1, h2, etc.).
- inline: element does not generate line breaks, only takes up as much width as necessary (e.g. span, a, img, etc.). It will be on the same line as other inline elements, if there is space.
- inline-block: element is displayed as an inline element, but can have specified width and height (e.g. buttons, input fields, etc.).
- none: element is not displayed at all.
- flex: element is displayed as a block-level flex container (e.g. div, section, etc.).

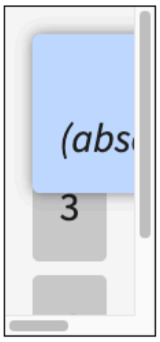
7.1. Positioning

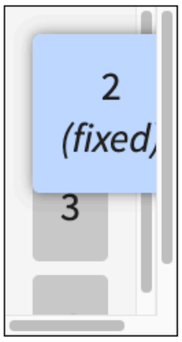
The position property specifies the type of positioning method used for an element.

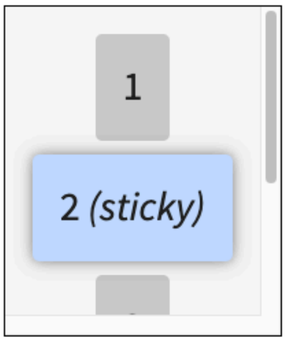
- static: The element is positioned according to the normal flow of the document.
- relative: The element is positioned according to the normal flow of the document, and then offset relative to itself.
- absolute: The element is positioned relative to its closest non-static positioned ancestor.
- fixed: The element is positioned relative to the initial containing block established by the viewport and does not move with scroll.
- sticky: The element is positonned according to the normal flow of the document, but behaves like fixed when scrolled out of view. If another element is positioned sticky nearby, they will "push" each other out of the way.











7.2. Flexbox

Flexbox is a layout model that allows elements to align and distribute space within a container.

```
<div class="container">
    <div class="item">Item A</div>
    <div class="item">Item B</div>
    <div class="item">Item C</div>
</div>
```

```
.container {
    display: flex;
    flex-direction: row;
}
.item {
    order: 1;
    flex-grow: 1
}
```

- Container (parent) properties : flex-direction, flex-wrap, flex-flow, justify-content, align-items, align-content
- Item (child) properties : order, flex-grow, flex-shrink, flex-basis, flex, align-self

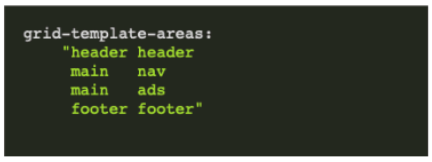
7.3. Grid

Grid is a layout model that allows elements to align and distribute space in two dimensions.

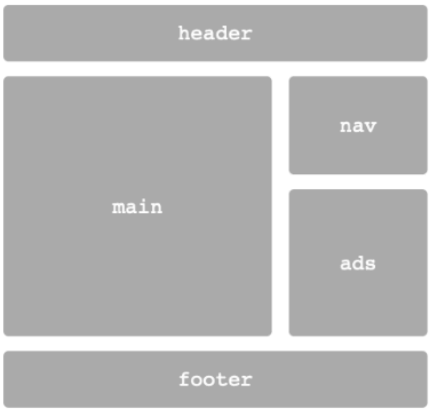
```
<div class="container">
    <div class="item">Item A</div>
    <div class="item">Item
    B</div>
    <div class="item">Item C</div>
</div>
```

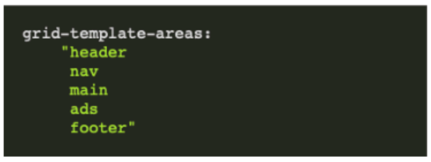
```
.container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
}
.item {
    grid-column: 1 / 3;
}
```

- Container (parent) properties : grid-template-column, grid-template-rows, grid-template-areas, grid-template, ...
- Item (child) properties : grid-column-start, grid-column-end, grid-column, grid-row, grid-area, ...

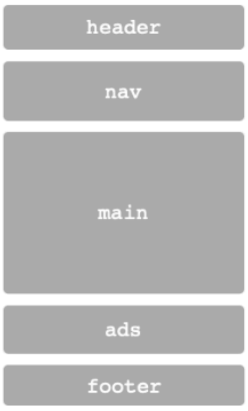


```
grid-template-areas:
    "header header
     main   nav
     main   ads
     footer footer"
```





```
grid-template-areas:
    "header
     nav
     main
     ads
     footer"
```



8. Media Queries

Media queries are used to apply different styles based on the device’s characteristics.

```
h1 { font-size: 50px; }     /* General rule */

@media (min-width: 576px) { /* Tablet dimensions */
    h1 { font-size: 60px; }
}

@media (min-width: 768px) { /* Desktop dimensions */
    h1 { font-size: 70px; }
}
```

8.1. Define variables

```
:root { /* Global variables */
    --main-color: #06c;
    --main-bg-color: #fff;
}

.subcontent { /* Local variables ; override global variables */
    --main-color: #c06;
    --main-bg-color: #fff;
    --content-specific-color: #f00;
}
```

8.2. Accessibing variables

```
.my-element {
    color: var(--main-color);
    background-color: var(--main-bg-color);
}
```

The var() function can take fallback values in case one of the variables is not defined.

```
.my-element {
    color: var(--main-color, #06c);
    background-color: var(--main-bg-color, #fff);
}
```