DAKAR AMERICAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COURSE: Introduction to Web Development

Lecture 2: CSS Fundamentals

Mor GUEYE
Software Engineering

Fall 2025

GOAL OF THE LECTURE

- Understand CSS Fundamentals.
- Understand the CSS Box Model
- Control element display and positioning
- Practice

What is CSS ?

- CSS (Cascading Style Sheets) is a language used to describe how HTML elements should be displayed on the screen.
- CSS Rules control layout, colors, fonts, and other visual aspects of a webpage.
- It separates **content** (HTML) from **presentation** (CSS), enabling better design flexibility.

CSS Syntax

- A CSS rule consists of:
  - **Selector**: Specifies which HTML element(s) the rule applies to.
  - **Declaration Block**: Contains one or more declarations, each specifying a CSS property and value.

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

```
h1 {
    color: blue;
    font-size: 24px;
}
```

Applying CSS

Three Ways to Apply CSS:

- **Inline CSS**: Written directly within an HTML tag.
  - <h1 **style="color: red;"**>Welcome</h1>
- **Internal CSS**: Placed within a <style> block in the <**head**> section of the HTML.
  - <**style**> h1 { color: red; } </**style**>
- **External CSS**: Written in a separate file and linked to the HTML.
  - <**link** rel="stylesheet" href="styles.css">

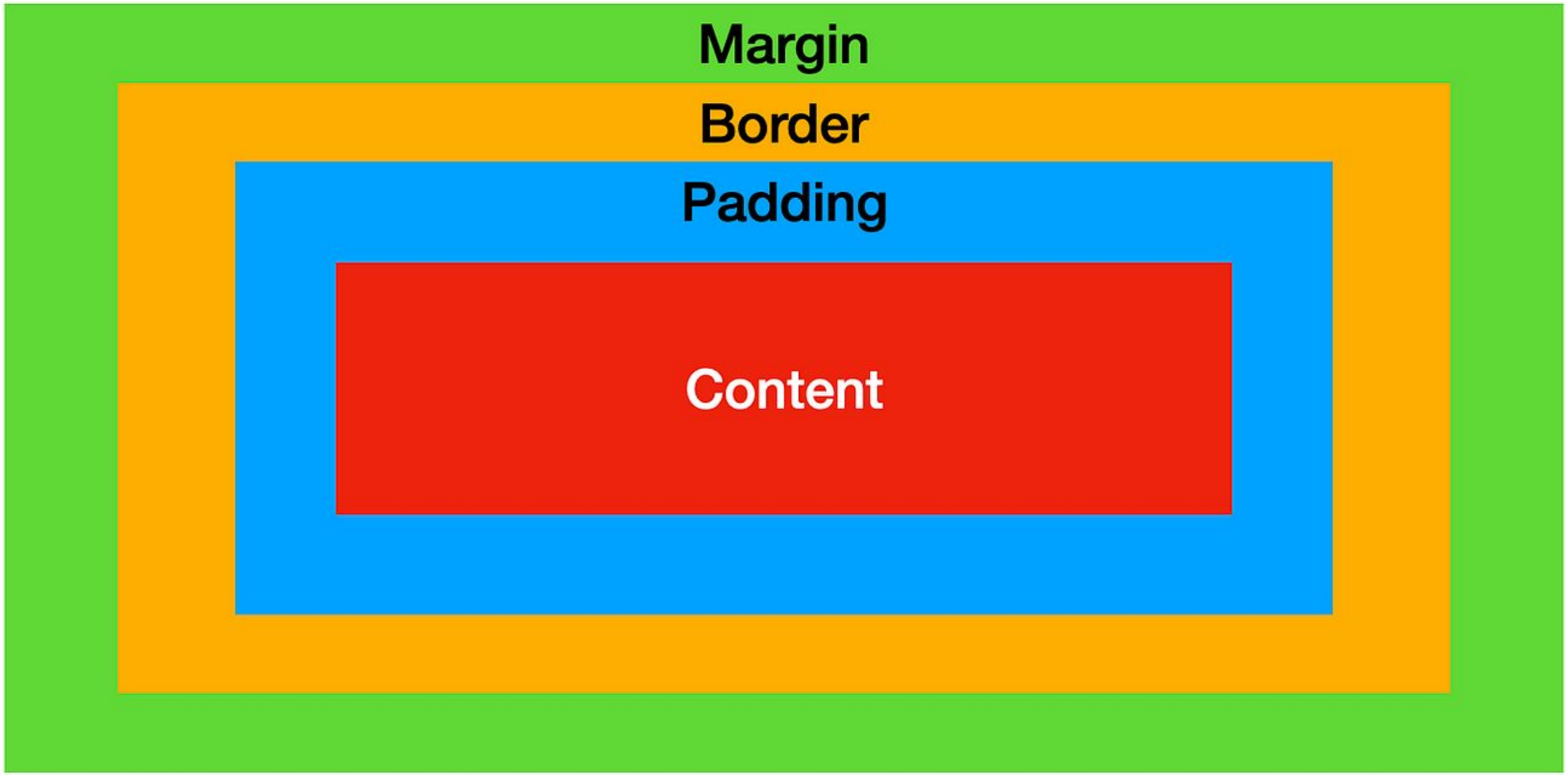
The Box Model

CSS Box Model is the fundamental concept for understanding layout in CSS.

- **Every element is a box with four parts:**
  - **Content**: The actual content inside the box (text, image, etc.).
  - **Padding**: Space between the content and the border.
  - **Border**: Surrounds the padding (can be visible or hidden).
  - **Margin**: Space between the border and other elements.

```
div {
    padding: 20px;
    border: 2px solid black;
    margin: 10px;
}
```

The Box Model



Display & Visibility

Every element has a display behavior that controls how it flows in the layout and how it interacts with other elements.
completely.

- **block** → takes full width (div, p, h1)
- **inline** → fits content (span, a)
- **inline-block** → inline but allows width/height
- **none** → hides element

Display & Visibility : block

- block Elements take full width of their parent container.
- Always start on a new line.
- Height/width/margin/padding can be applied.
- Examples: , <p>, <h1>

```
<div>Block element 1</div>

<div>Block element 2</div>
```

Display & Visibility : inline

- Element takes only as much width as its content.
- Does not start on a new line.
- You cannot set width/height directly.
- Examples: , <a>,

```
<p>This is <span>inline text</span> inside a paragraph.</p>
```

Display & Visibility : inline-block

- Behaves like inline (stays in same line),
- but you can also set width, height, margin, padding like a block.
- Useful for creating button-like elements.

```
<span style="display:inline-block; width:100px; height:30px; background:lightgray;">
  Inline-block box
</span>
```

Display & Visibility : none

- Completely removes the element from the layout.
- Other elements behave as if it doesn’t exist..

```
<p style="display:none;">This text is hidden and takes no space.</p>
```

Display & Visibility : visibility: hidden

- The element is invisible but still takes up space in the layout.
- Different from display: none.

```
<p style="visibility:hidden;">You can’t see me, but I take
space.</p>
```

Display & Visibility : display: none vs visibility: hidden

👉 Key Difference:

- ❑ display: none = element is removed from layout.
- ❑ visibility: hidden = element is invisible but still occupies space.

Positioning

Positioning is a fundamental concept that controls how elements are placed within a web page

- static → default
- relative → relative to itself
- absolute → relative to parent
- fixed → relative to viewport
- sticky → sticks while scrolling

```
div { position: absolute; top: 50px; left: 100px; }
```

Pseudo-classes & Pseudo-elements

These selectors target elements based on their state or a specific part of them.

- **Pseudo-classes (e.g., :hover, :focus):** Selects an element when it is in a certain state.
  - **Example: a:hover { color: red; }** changes the color of a link to red when the mouse hovers over it.
- **Pseudo-elements (e.g., ::first-letter, ::before):** Selects and styles a specific part of an element.
  - **Example: p::first-letter { font-size: 2em; }** makes the first letter of every paragraph significantly larger.

CSS Selectors: Attribute Selectors

- Target elements based on their attributes and their corresponding values.
- Give you more specific control than just using element names or classes, allowing you to style elements that share a certain characteristic.
- Types :
  - Simple Attribute Selector
  - Exact Value Selector
  - Substring Value Selectors

CSS Selectors: Attribute Selectors

Simple Attribute Selector [attribute]

This selector targets any element that has a specific attribute, regardless of its value.

- **Example:** **a[title]** selects all **<a>** (link) elements that have a **title** attribute. This is useful for styling elements that have a tooltip or some form of descriptive text.

CSS Selectors: Attribute Selectors

Exact Value Selector [attribute="value"]

This is the most common form. It targets elements where a specific attribute has an exact value.

- **Example:** input[type="text"] selects only the input fields that are for text entry. This is a common way to style different types of form inputs, like text boxes, checkboxes, and radio buttons, differently.

CSS Selectors: Attribute Selectors

Substring Value Selectors

These are more flexible selectors that match partial attribute values.

- **Starts With [attribute^="value"]**: Selects elements where the attribute's value begins with a specific string.
  - **Example:** a[href^="https"] will style all links that use a secure protocol, ensuring they stand out.
- **Ends With [attribute$="value"]**: Selects elements where the attribute's value ends with a specific string.
  - **Example:** a[href$=".pdf"] can be used to add a special icon next to links that lead to PDF files.
- **Contains [attribute*="value"]**: Selects elements where the attribute's value contains a specific string anywhere within it.
  - **Example:** a[href*="google"] will style all links that point to a URL containing "google," no matter where that string appears in the address. This is great for highlighting external links to specific domains.

CSS Selectors: Combinator Selectors

Descendant Selector : Selects all elements that are descendants of a specified element. The descendant element doesn't have to be a direct child; it can be nested many levels deep.

- **Example: div p { color: green; }** will make any paragraph *inside* a ** ** green, regardless of how many other elements are in between.

Child Selector (>): Selects all elements that are *direct children* of a specified element.

- **Example: ul > li { list-style-type: none; }** will remove the bullet points from the list items that are immediately nested within a **<ul>** element.

CSS Selectors: Combinator Selectors

General Sibling Selector (~)

The **general sibling selector** **(A ~ B)** selects all sibling elements **B** that follow a specified element **A**, as long as they have the same parent. Unlike the adjacent selector, this one isn't limited to just the very next element. It will apply the style to every subsequent sibling element.

Example:

- **h2 ~ p** will select every **<p>** tag that comes after an **<h2>** tag.
- If there are multiple paragraphs after the heading, all of them will be affected by the style.

CSS Selectors: Combinator Selectors

Adjacent Sibling Selector (+)

The **adjacent sibling selector (A + B)** selects an element **B** that immediately follows element **A**, as long as they share the same parent. This selector is very specific, as it only targets the one element that comes directly after the first one specified.

Example:

- **h2 + p** will only select the first **<p>** tag that comes right after an **<h2>** tag in the HTML.
- If you have two paragraphs after the heading, only the first one will be affected by the style.

Units in CSS

Absolute Units:

px - Pixels (most common)

pt - Points (1/72 of inch)

cm, mm, in - Physical measurements

Relative Units:

% - Percentage of parent element

em - Relative to font-size of element

rem - Relative to root element font-size

vw - 1% of viewport width

vh - 1% of viewport height

vmin - 1% of viewport's smaller dimension

vmax - 1% of viewport's larger dimension

Colors in CSS

CSS supports multiple ways to define colors.

- **Named colors:** red, blue, green.
- **HEX values:** A six-digit hexadecimal code.
- **RGB values:** Defines color using red, green, and blue values
- **HSL values :** (Hue, Saturation, Lightness)
- **Modern colors spaces** : oklh, Lab, lch

Colors in CSS

```
color: red; /* Named color */

color: #ff0000; /* Hexadecimal */

color: #f00; /* Short hexadecimal */

color: rgb(255, 0, 0); /* RGB */

color: rgba(255, 0, 0, 0.5); /* RGB with alpha */

color: hsl(0, 100%, 50%); /* HSL */

color: hsla(0, 100%, 50%, 0.3); /* HSL with alpha */

color: oklch(62% 0.25 29); /* OKLCH */
```

THANKS