# CSS Fundamentals

Emmanuel BRUNO
Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

2026-11-01

## Table des matières

1 Overview . . . . . 2
2 Introduction to CSS . . . . . 2
2.1 External CSS (styles.css) . . . . . 3
2.2 Minimal CSS+HTML Example . . . . . 3
3 CSS Syntax . . . . . 3
4 Common Selectors . . . . . 4
4.1 Element Selector . . . . . 4
4.2 Class Selector . . . . . 4
4.3 ID Selector . . . . . 4
4.4 Attribute Selector . . . . . 5
4.5 Parent Selector . . . . . 5
4.6 Ancestor Selector . . . . . 5
5 Styling Text and Elements . . . . . 6
6 The Cascade Mechanism: Introduction . . . . . 7
6.1 The Cascade Mechanism: Specificity . . . . . 7
6.2 The Cascade Mechanism: Importance . . . . . 7
6.3 The Cascade Mechanism: Source Order . . . . . 8
7 Box Model . . . . . 8
8 Layout Techniques: Display Property . . . . . 9
8.1 Positioning . . . . . 9
8.2 Flexbox . . . . . 10
8.3 Flexbox Example . . . . . 10
8.4 Grid . . . . . 11
9 Responsive Design and Media Queries . . . . . 11
9.1 Media Queries Syntax . . . . . 12
9.2 Media Query Examples . . . . . 12
9.3 Example 2: Responsive Layout . . . . . 13
9.4 Example 3: Hiding Elements . . . . . 13

9.5 Fluid Layouts ..... 13
9.6 Responsive Units ..... 14
9.7 Example: Fluid Layout with Percentage Width and Responsive Units ..... 14
10 Hands-on Exercises ..... 15
11 Using CSS Frameworks ? Why ? ..... 15
12 Bootstrap ..... 15
12.1 Key Features: ..... 16
12.2 Quickstart with Bootstrap ..... 16
12.3 Using Bootstrap Components: ..... 17
12.4 Creating a Responsive Grid: ..... 17
12.5 Adding Buttons and Forms: ..... 18
12.6 Adding More Bootstrap Components : Modals ..... 18
12.7 Alerts ..... 19
12.8 Cards ..... 19
12.9 Tooltips ..... 20

# 1 Overview

- **Topics:**
  - Introduction to CSS
  - CSS Syntax and Selectors
  - Styling Text and Elements
  - Box Model
  - Layout Techniques
  - Responsive Design
  - Hands-on Exercises

# 2 Introduction to CSS

- **What is CSS?**
  - CSS stands for **Cascading Style Sheets**.
  - It is used to control the presentation, formatting, and layout of web pages.
  - CSS separates content (HTML) from design, making it easier to maintain and update websites.
  - It is a set of specifications see. https://www.w3.org/Style/CSS/specs.en.html
- **Why is CSS Important?**
  - CSS enhances the user experience by allowing developers to create visually appealing and responsive web pages.
  - It enables consistent styling across multiple pages, improving the overall look and feel of a website.
  - CSS is essential for modern web development, as it allows for the creation of complex layouts and designs.

- **Key Points:**
  - **Selectors:** Target HTML elements to apply styles.
  - **Properties and Values:** Define the styles to be applied.
  - **Cascading:** Styles can be applied from multiple sources, with rules for resolving conflicts.

### 2.1 External CSS (styles.css)

```
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}

p {
  color: green;
  font-size: 16px;
}
```

### 2.2 Minimal CSS+HTML Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Minimal CSS Example</title>
  <!-- Link to External CSS -->
  <link rel="stylesheet" href="styles.css">
  <!-- Inline CSS -->
  <style>
    .inline-style {
      color: blue;
      font-size: 20px;
    }
  </style>
</head>
<body>
  <h1 class="inline-style">Hello, World!</h1>
  <p>This is a paragraph styled with external CSS.</p>
</body>
</html>
```

## 3 CSS Syntax

- **CSS Syntax:**

- A CSS rule consists of a selector and a declaration block.
- CSS:

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

# 4 Common Selectors

## 4.1 Element Selector

- **Element Selector:**
  - Targets all instances of an element.
  - CSS:

```
p { color: blue; }
```

  - HTML:

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

## 4.2 Class Selector

- **Class Selector:**
  - Targets elements with a specific class attribute.
  - CSS:

```
.classname { color: green; }
```

  - HTML:

```
<p class="classname">This paragraph has a class.</p>
<p>This paragraph does not have a class.</p>
```

## 4.3 ID Selector

- **ID Selector:**
  - Targets a single element with a specific ID attribute.
  - CSS:

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

- HTML:

```html
<p id="idname">This paragraph has an ID.</p>
<p>This paragraph does not have an ID.</p>
```

## 4.4 Attribute Selector

- **Attribute Selector:**
  - Targets elements with a specific attribute.
  - CSS:

```css
[type="text"] { color: blue; }
```

  - HTML:

```html
<input type="text" placeholder="Text input">
<input type="password" placeholder="Password input">
```

## 4.5 Parent Selector

- **Parent Selector:**
  - Targets the direct parent of an element.
  - CSS:

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

  - HTML:

```html
<div>
  <p>This paragraph is inside a div.</p>
</div>
<p>This paragraph is not inside a div.</p>
```

## 4.6 Ancestor Selector

- **Ancestor Selector:**
  - Targets an element that is a descendant of another element.
  - CSS:

```
div p { color: blue; }
```

- HTML:

```
<div>
  <p>This paragraph is inside a div.</p>
</div>
<p>This paragraph is not inside a div.</p>
```

## 5 Styling Text and Elements

- **Text Properties:**
  - `color`: Sets the text color.
  - `font-family`: Sets the font of the text.
  - `font-size`: Sets the size of the text.
  - `font-weight`: Sets the weight (boldness) of the text.
  - `text-align`: Aligns the text (left, right, center, justify).
  - CSS:

```
h1 {
  color: blue;
  font-family: Arial, sans-serif;
  font-size: 24px;
  font-weight: bold;
  text-align: center;
}
```

- **Background Properties:**
  - `background-color`: Sets the background color.
  - `background-image`: Sets a background image.
  - `background-repeat`: Controls the repetition of the background image.
  - `background-position`: Sets the initial position of the background image.
  - CSS:

```
body {
  background-color: lightgray;
  background-image: url('background.jpg');
  background-repeat: no-repeat;
```

```
  background-position: center;
}
```

# 6 The Cascade Mechanism: Introduction

- **What is the Cascade?**
  - The cascade is a mechanism that determines which CSS rules are applied when multiple rules target the same element.
  - It resolves conflicts between different CSS rules based on three main factors:
    - **Specificity:** How specific the selector is.
    - **Importance:** Whether the rule is marked as `!important`.
    - **Source Order:** The order in which the rules appear in the stylesheet.

## 6.1 The Cascade Mechanism: Specificity

- **Specificity:**
  - Specificity is a measure of how specific a CSS selector is.
  - More specific selectors have higher priority.
  - CSS:

```
/* Less specific */
p { color: blue; }
/* More specific */
.classname { color: green; }
/* Most specific */
#idname { color: red; }
```

  - HTML:

```
<p>This is a paragraph.</p>
<p class="classname">This paragraph has a class.</p>
<p id="idname">This paragraph has an ID.</p>
```

## 6.2 The Cascade Mechanism: Importance

- **Importance:**
  - The `!important` declaration can be used to override other rules.
  - CSS:

```
p { color: blue !important; }
.classname { color: green; }
```

- HTML:

```
<p class="classname">This paragraph will be blue due to the !important declaration.</p>
```

### 6.3 The Cascade Mechanism: Source Order

- **Source Order:**
  - When specificity and importance are equal, the last defined rule in the source order is applied.
  - CSS:

```
p { color: blue; }
p { color: red; }
/* The paragraph will be red because the last rule is applied */
```

  - HTML:

```
<p>This paragraph will be red because the last rule is applied.</p>
```

## 7 Box Model

- **Understanding the Box Model:**
  - Every HTML element is considered a box.
  - The box model consists of: content, padding, border, and margin.
  - CSS:

```
.box {
  width: 200px;
  height: 100px;
  padding: 10px;
  border: 1px solid black;
  margin: 20px;
}
```

- **Box Model Properties:**
  - **width and height:** Set the size of the content area.
  - **padding:** Space between the content and the border.
  - **border:** Surrounds the padding (if any) and content.
  - **margin:** Space outside the border.

# 8 Layout Techniques: Display Property

- **What is a Layout?**
  - A layout is the arrangement of elements on a web page.
  - It determines how content is organized and presented to the user.
- **What is a Layout Technique?**
  - A layout technique is a method used to control the positioning and alignment of elements on a web page.
  - Common layout techniques include using the display property, Flexbox, and CSS Grid.
- **Display Property:**
  - `display: block;`: The element takes up the full width available.
  - `display: inline;`: The element takes up only as much width as necessary.
  - `display: inline-block;`: The element is formatted as an inline element but can have a width and height.
  - CSS:

```css
.block { display: block; }
.inline { display: inline; }
.inline-block { display: inline-block; }
```

## 8.1 Positioning

- **What is Positioning?**
  - Positioning is a technique used to control the placement of elements on a web page.
  - It allows you to specify the exact location of an element within its containing element or the viewport.
- **Positioning:**
  - `position: static;`: Default positioning.
  - `position: relative;`: Positioned relative to its normal position.
  - `position: absolute;`: Positioned relative to the nearest positioned ancestor.
  - `position: fixed;`: Positioned relative to the viewport.
  - CSS:

```css
.relative { position: relative; top: 10px; left: 10px; }
.absolute { position: absolute; top: 20px; left: 20px; }
.fixed { position: fixed; top: 30px; left: 30px; }
```

## 8.2 Flexbox

- **What is Flexbox?**
  - Flexbox, or the Flexible Box Layout, is a layout model designed to provide a more efficient way to lay out, align, and distribute space among items in a container.
  - It is particularly useful for creating responsive layouts and handling dynamic content.
- **Key Concepts:**
  - **Flex Container:** The parent element that holds the flex items. Defined by `display: flex;`.
  - **Flex Items:** The child elements of the flex container.
- **Flex Container Properties:**
  - `flex-direction`: Direction of the flex items (row, column).
  - `justify-content`: Aligns items along the main axis (flex-start, center, space-between).
  - `align-items`: Aligns items along the cross axis (flex-start, center, stretch).
- **Flex Item Properties:**
  - `flex-grow`: How much a flex item will grow relative to the rest.
  - `flex-shrink`: How much a flex item will shrink relative to the rest.
  - `flex-basis`: Initial size of a flex item.

## 8.3 Flexbox Example

- **CSS:**

```css
.container {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}
.item {
  flex: 1;
  padding: 10px;
  background-color: lightblue;
  margin: 5px;
}
```

- **HTML:**

```html
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
```

```
  <div class="item">Item 3</div>
</div>
```

- **Explanation:**
  - The `.container` class defines a flex container with `display: flex;`.
  - `flex-direction: row;` arranges items in a row.
  - `justify-content: space-between;` distributes space between items.
  - `align-items: center;` aligns items along the cross axis.
  - Each `.item` grows to fill available space equally (`flex: 1;`).

## 8.4 Grid

- **Grid:**
  - A layout model for creating complex, responsive grid-based layouts.
  - CSS:

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
}
.grid-item {
  background-color: lightblue;
  padding: 20px;
  text-align: center;
}
```

  - HTML:

```
<div class="grid-container">
  <div class="grid-item">Item 1</div>
  <div class="grid-item">Item 2</div>
  <div class="grid-item">Item 3</div>
</div>
```

# 9 Responsive Design and Media Queries

- **What is Responsive Design?**
  - Designing websites that adapt to different screen sizes and devices.
  - Ensures a consistent and optimal user experience across various devices.
- **What are Media Queries?**

- Media queries are a CSS feature that allows you to apply different styles based on the characteristics of the device, such as screen width, height, orientation, and resolution.

- **Benefits of Media Queries:**
  - Allows for a responsive design that adapts to different devices.
  - Improves user experience by providing optimal layouts for various screen sizes.
  - Reduces the need for separate mobile and desktop versions of a website.

## 9.1 Media Queries Syntax

- Media queries use the `@media` rule followed by a condition and a block of CSS rules.
- Example:

```
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

- **Common Media Query Conditions:**
  - `max-width`: Applies styles if the viewport width is less than or equal to the specified value.
  - `min-width`: Applies styles if the viewport width is greater than or equal to the specified value.
  - `max-height`: Applies styles if the viewport height is less than or equal to the specified value.
  - `min-height`: Applies styles if the viewport height is greater than or equal to the specified value.
  - `orientation`: Applies styles based on the device orientation (portrait or landscape).

## 9.2 Media Query Examples

- **Example 1: Changing Background Color**
  - CSS:

```
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
@media (min-width: 601px) {
  body {
```

```
    background-color: white;
  }
}
```

- **Explanation:**
  - The first example changes the background color based on the viewport width.

## 9.3 Example 2: Responsive Layout

- CSS:

```
.container {
  display: flex;
  flex-direction: row;
}
@media (max-width: 600px) {
  .container {
    flex-direction: column;
  }
}
```

- **Explanation:**
  - The second example changes the flex direction of a container to a column layout on smaller screens.

## 9.4 Example 3: Hiding Elements

- CSS:

```
.sidebar {
  display: block;
}
@media (max-width: 600px) {
  .sidebar {
    display: none;
  }
}
```

- **Explanation:**
  - The third example hides the sidebar on screens with a width of 600px or less.

## 9.5 Fluid Layouts

- **What are Fluid Layouts?**
  - Fluid layouts use relative units like percentages for widths and heights to create flexible and responsive designs.

- They adapt to the size of the viewport, ensuring that content scales appropriately on different devices.

- **Percentage Width:**
  - Using percentage values for width allows elements to resize relative to their parent container.
  - Example:

```
.fluid {
  width: 100%; /* Element takes up 100% of its parent container's width */
  max-width: 1200px; /* Maximum width of the element */
  margin: 0 auto; /* Center the element horizontally */
}
```

## 9.6 Responsive Units

- **Viewport Width (vw):** 1vw is equal to 1% of the viewport's width.
- **Viewport Height (vh):** 1vh is equal to 1% of the viewport's height.
- **Example:**

```
.responsive {
  width: 50vw; /* Element takes up 50% of the viewport's width */
  height: 50vh; /* Element takes up 50% of the viewport's height */
}
```

## 9.7 Example: Fluid Layout with Percentage Width and Responsive Units

- CSS:

```
.container {
  width: 80%; /* Element takes up 80% of its parent container's width */
  max-width: 1200px; /* Maximum width of the element */
  margin: 0 auto; /* Center the element horizontally */
}
.responsive-box {
  width: 50vw; /* Element takes up 50% of the viewport's width */
  height: 50vh; /* Element takes up 50% of the viewport's height */
  background-color: lightblue;
}
```

- HTML:

```
<div class="container">
  <div class="responsive-box">Responsive Box</div>
</div>
```

## 10 Hands-on Exercises

Before studying a framework to be more efficient, let's do some exercises to practice CSS fundamentals.

Please go to CSS Hands-on

## 11 Using CSS Frameworks ? Why ?

- **Consistency:**
  - Provides a consistent look and feel across different pages and projects.
  - Ensures uniformity in design elements like buttons, forms, and navigation bars.
- **Efficiency:**
  - Saves time by providing pre-designed components and styles.
  - Reduces the need to write custom CSS from scratch.
- **Responsiveness:**
  - Built-in responsive design features that adapt to different screen sizes.
  - Ensures your website looks good on all devices.
- **Community and Support:**
  - Large community of developers contributing to the framework.
  - Extensive documentation and support resources available.

## 12 Bootstrap

- **Origin:**
  - Bootstrap was originally developed by Mark Otto and Jacob Thornton at Twitter.
  - It was released as an open-source project in 2011.
  - The framework quickly gained popularity due to its ease of use and comprehensive set of features.
- **Description:**
  - Bootstrap is a powerful front-end framework for faster and easier web development.
  - It includes HTML, CSS, and JavaScript components for creating responsive and mobile-first websites.
  - The framework provides a wide range of pre-designed components and utilities, such as navigation bars, buttons, forms, modals, and more.
  - Bootstrap is highly customizable and can be extended with custom styles and components.

- It follows a grid-based layout system, making it easy to create complex and responsive layouts.
- The framework is well-documented and supported by a large community of developers.

## 12.1 Key Features:

- **Responsive Design:**
  - Built-in responsive design features that adapt to different screen sizes.
  - Ensures your website looks good on all devices.
- **Pre-designed Components:**
  - A wide range of components like navigation bars, buttons, forms, modals, and more.
  - Saves time by providing ready-to-use elements.
- **Customizable:**
  - Easy to customize and extend with custom styles and components.
  - Allows for a high degree of flexibility in design.
- **Grid System:**
  - A powerful grid system for creating complex and responsive layouts.
  - Simplifies the process of designing multi-column layouts.
- **Cross-browser Compatibility:**
  - Ensures consistent appearance and behavior across different browsers.
  - Reduces the need for browser-specific adjustments.

## 12.2 Quickstart with Bootstrap

**Use update to date version of Bootstrap from Bootstrap Official Website**

- **Include Bootstrap via CDN:**
  - Add the following `<link>` and `<script>` tags to the `<head>` section of your HTML file.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bootstrap Quickstart</title>
  <!-- Bootstrap CSS -->
          <link          href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/
dist/css/bootstrap.min.css"        rel="stylesheet"        integrity="sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
</head>
```

```
<body>
  <!-- Your content here -->

  <!-- Bootstrap JS and dependencies -->
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4" crossorigin="anonymous"></script>

</body>
</html>
```

## 12.3 Using Bootstrap Components:

- Use Bootstrap **classes** to style your HTML elements.
- Example: Creating a responsive navigation bar.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
      <button   class="navbar-toggler"   type="button"   data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
          <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Features</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Pricing</a>
      </li>
    </ul>
  </div>
</nav>
```

## 12.4 Creating a Responsive Grid:

Use Bootstrap's grid system to create responsive layouts.

```
<div class="container">
  <div class="row">
    <div class="col-md-4">Column 1</div>
    <div class="col-md-4">Column 2</div>
    <div class="col-md-4">Column 3</div>
  </div>
</div>
```

## 12.5 Adding Buttons and Forms:

Use Bootstrap classes to style buttons and forms.

```
<button type="button" class="btn btn-primary">Primary Button</button>

<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
      <input type="email" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

## 12.6 Adding More Bootstrap Components : Modals

- Use Bootstrap classes to create modals for displaying content in a dialog box.
- Example:

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#exampleModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
```

```
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        This is a Bootstrap modal.
      </div>
      <div class="modal-footer">
                <button  type="button"  class="btn  btn-secondary"  data-bs-
dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

## 12.7 Alerts

- Use Bootstrap classes to create alert messages for feedback.
- Example:

```
<div class="alert alert-success" role="alert">
  This is a success alert—check it out!
</div>
<div class="alert alert-danger" role="alert">
  This is a danger alert—check it out!
</div>
```

## 12.8 Cards

- Use Bootstrap classes to create card components for displaying content.
- Example:

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title
and make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

## 12.9 Tooltips

- Use Bootstrap classes to add tooltips to elements for additional information.
- Example:

```
<button type="button" class="btn btn-secondary" data-bs-toggle="tooltip" data-bs-placement="top" title="Tooltip on top">
  Tooltip on top
</button>
<script>
  var tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-bs-toggle="tooltip"]'))
  var tooltipList = tooltipTriggerList.map(function (tooltipTriggerEl) {
    return new bootstrap.Tooltip(tooltipTriggerEl)
  })
</script>
```