



Web Dev course part one: Intro to HTML5 & CSS

-Intro HTML:

id/class/div/structure/META/semantic/link css to html.

-Intro to CSS:

- What is CSS?

- how to code in CSS(syntax)

- CSS selectors.

- Priorities in CSS

- CSS basic attributes(color,background-color,width,height)

- Box model(margin,border,padding,content)

What is HTML?

HTML stands for HyperText Markup Language. It is the language used to create web pages. HTML uses tags to tell the browser how to display the content of a web page. For example, the `<h1>` tag tells the browser to display the text that follows it as a heading.

HTML is a markup language, not a programming language. This means that it is used to describe the structure of a web page, not to control the behavior of a web page.

Why should our first HTML file be named index.html?

The index.html file is the default file that is loaded when a user visits a website. This is why it is important to name your first HTML file index.html. If you do not, the browser will try to load a file called index.html in the same directory as the file that the user requested.

In addition, search engines will typically index the index.html file for a website. This means that if you want your website to be found by search engines, you should name your first HTML file index.html.





Intro HTML: Structure & Importance des tags META

1-HTML Structure:

As shown in the image on the next page, every HTML file begins with `<!DOCTYPE html`, which has the sole purpose of defining the document as HTML5.

Following that, there's `<html lang="en">`, which merely indicates the language of the webpage, aiding web browsers and search engines.

Next, you'll come across the `<head>` tag, within which you'll find metadata (information not displayed on the site) used to define the document's title, character encoding, styles, scripts, and other meta-information. In the `<head>`, you often include links to icon libraries (such as Font Awesome) and a link to the CSS file.

2-Importance balises META:

Meta tags are important elements for web pages. For example, the viewport meta tag enables a website to be responsive and adapt to mobile devices. The charset meta tag specifies the character encoding used on the page, such as UTF-8 to support various international characters. These tags play a crucial role in improving a site's visibility in search results and ensuring a good user experience.

```
<!DOCTYPE html>//pour spécifier le type de document(html5)
<html lang="en">pour specifier la langue de la page
  <head>
    <meta charset="UTF-8" />//attribut qui determine la maniere dont le texte est transmis et donné
    <meta http-equiv="X-UA-Compatible" content="IE=edge" /> /*permet aux auteurs de sites web de
    choisir la version d'Internet Explorere dans laquelle la page doit etre rendue*/
    <meta name="viewport" content="width=device-width, initial-scale=1.0" /> //width de la page =screen width
    <title>Document</title> //titre du document
    <link rel="stylesheet" href="style.css" /> //lien vers css
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.3.0/css/all.min.css"
    /> //lien pour utiliser icones de font-awesome
  </head>
  <body> //le code qui est affiché à l'intérieur de body
</body>
</html>
```



HTML Tags Inside the `<body>` Element

In HTML, the content displayed on a web page is enclosed within the `<body>` element. Inside the `<body>`, you can use various tags to structure and format your content. Here are some of the essential HTML tags for this purpose:

`<p>` - Paragraph:

Use this tag to define and format text paragraphs. It automatically adds spacing before and after the content.

`<h1>` to `<h6>` - Headings:

These tags define six levels of headings, where `<h1>` represents the highest level (most important) and `<h6>` the lowest. They are used for titles and subtitles.

`<a>` - Anchor (Hyperlink):

The `<a>` tag is used to create hyperlinks, allowing you to link to other web pages or resources.



HTML Tags Inside the `<body>` Element

`` - Unordered List

This tag is used to create an unordered (bulleted) list. You can nest `` tags inside `` to define list items.

`` - List Item

Used inside `` or `` tags, `` defines individual list items within a list.

`
` - Line Break

The `
` tag forces a line break within your content, useful for creating new lines without starting a new paragraph.



HTML Tags Inside the `<body>` Element

`` - Image

Use this tag to embed images on your web page, specifying the source (src) and alternative text (alt) for accessibility.

`<div>` - Division

The `<div>` tag is a versatile container for grouping and styling content. It's often used with CSS for layout purposes.

`` - Inline Span

Similar to `<div>`, but for inline elements. Use `` to apply styles or scripting to specific portions of text.

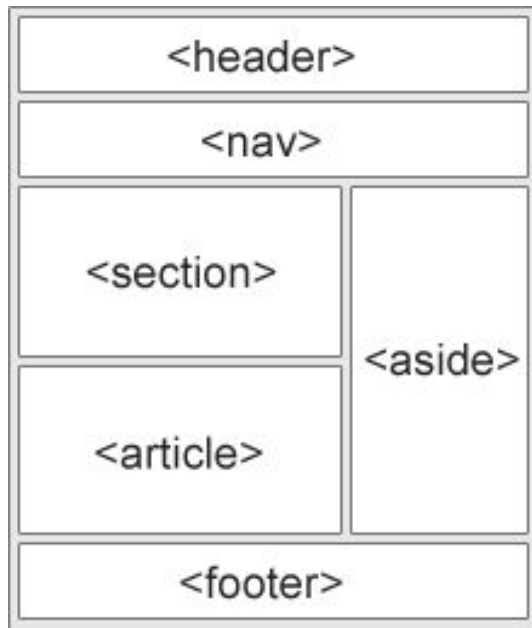
Intro HTML:semantic

Semantics are a bit like the blueprint of your website, its structure, and its framework. Semantic markup is formed using various HTML tags as seen below.

Adhering to semantic structure is important not only for helping web browsers but also for improving accessibility, search engine optimization, and the overall understanding of your site. It's about organizing your HTML effectively.

1. `<header>`: It represents the header of a section or a page and can contain elements like the logo, title, navigation, and more. This aids search engines and screen readers in clearly identifying the header section of a page.
2. `<nav>`: This tag is used to represent a navigation bar.
3. `<main>`: It represents the main content of a page. This tag is important as it allows search engines to easily identify the primary content of your page.
4. `<article>`: This tag is used to represent independent and meaningful content that can be reused or distributed separately, such as a blog post or an article. It helps in organizing and identifying standalone content.
5. `<section>`: It's used to divide content into thematic sections, aiding in structuring content and facilitating the understanding of context and information hierarchy.
6. `<aside>`: This tag represents related content or additional information in relation to the main content of the page. It's often used for things like sidebars or advertisements.
7. `<footer>`: It represents the footer of a section or a page, typically containing information like contact details, additional navigation links, copyright notices, and more.

Adhering to semantic markup not only makes your website more accessible and SEO-friendly but also ensures that it is well-organized and easy to understand by both humans and search engines.





Intro HTML:id & class

1-id:

The `id` attribute allows you to assign a unique identifier that can be given to any of your HTML tags. This identifier can be used for internal links, applying CSS styles by referencing the `id`, and also in the Document Object Model (DOM) to manipulate the content and attributes of the HTML using JavaScript.

You can also utilize it in your CSS, as you will see later on.

```
<body>
  <div id="exemple">
    <p>le paragraphe est dans une div aux bordures rouges</p>
  </div>
</body>
```

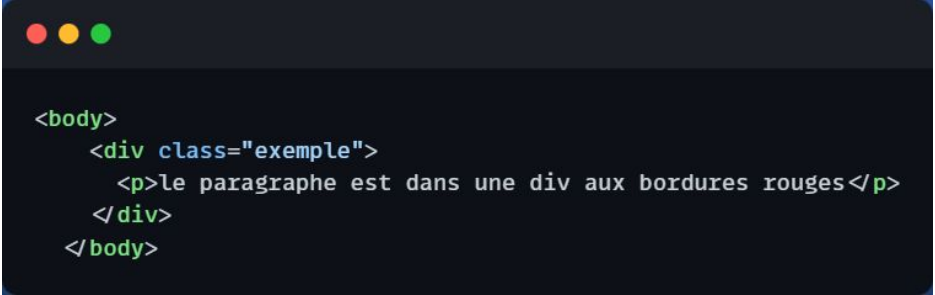
```
#exemple {
  border: 2px solid red;
}
```



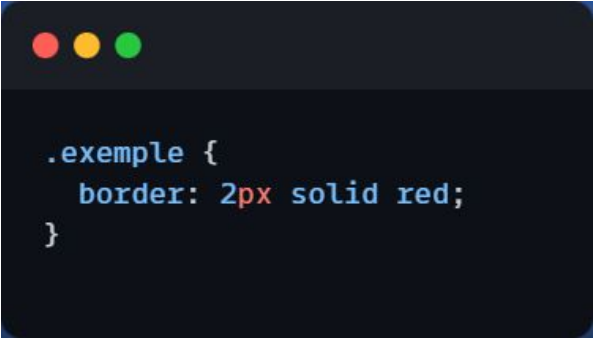
Intro HTML: id & class

2-class:

The `class` attribute serves similar purposes to `id`, but with some key differences. You can assign multiple classes to a single HTML element, and you can also apply the same class to multiple tags. While it is technically possible to assign the same `id` to multiple elements, it's generally not recommended because it can lead to confusion for internal links and DOM manipulation. `class`, on the other hand, is more flexible and commonly used for styling and scripting purposes across multiple elements.



```
<body>
  <div class="exemple">
    <p>le paragraphe est dans une div aux bordures rouges</p>
  </div>
</body>
```



```
.exemple {
  border: 2px solid red;
}
```




Intro HTML:link CSS to HTML

In order for your CSS to be applied to your HTML and work correctly, it's imperative to link them by adding a simple line of code in the ``<head>`` section of your HTML document.

You can link multiple CSS files to a single HTML file, and you can also use a single CSS file across multiple HTML pages.

The `rel="stylesheet"` attribute informs the browser that the referenced file is a style sheet (CSS) to be applied to the HTML page.



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



Intro to CSS: What is CSS?

CSS is the language we use to style an HTML document. It allows us to set colors, borders, adjust the layout of our elements, and much more.

Within CSS, you'll encounter numerous properties, and it's natural to feel a bit overwhelmed at the beginning.

However, don't worry; here, you'll learn the fundamentals that will propel you further in your web design journey.

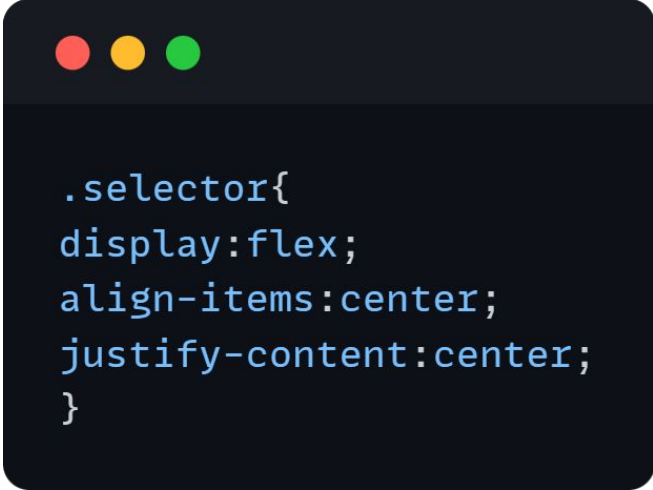




Intro to CSS:How to Write CSS code(syntax)

The syntax of CSS is wonderfully straightforward. You choose what you want to modify (the selector). You'll discover the three primary types of selectors shortly (tag, id, class).

Then, you open a block of declarations where you specify the desired properties. Here's an example:



```
.selector{  
  display:flex;  
  align-items:center;  
  justify-content:center;  
}
```



Intro to CSS:Selectors

Here, we'll introduce you to the three most commonly used and simplest selectors:

1. **Tag Name Selector**: Select elements based on their tag name. For example, `nav`, `p`, `ul`, and so on.
2. **ID Selector**: Select elements by their unique identifier. Start with a `#` followed by the ID name. For example, `#home`.
3. **Class Selector**: Select elements by their class. Begin with a period `.` followed by the class name. For example, `.title`.

In addition to these basic selectors, CSS also provides more advanced selectors for more precise targeting:

- **Attribute Selectors**: These select elements based on the values of their attributes. For example, `[type="text"]` selects elements with the `type` attribute set to "text."
- **Descendant Selectors**: These select elements that are direct or indirect descendants of another element. For example, `div p` selects all `p` elements that are direct descendants of a `div` element.
- **Pseudo-Class Selectors**: These select elements in specific states. For example, `:hover` selects an element when the cursor is placed over it.
- **Combination Selectors**: These combine multiple selectors to achieve more specific selections. For example, `div p` selects all `p` elements that are direct descendants of a `div` element, while `div > p` selects only direct descendants.

Understanding these selectors will allow you to target and style elements in your HTML documents effectively.



Intro to CSS:Selectors

1-class:

```
.exemple {  
  border: 2px solid red;  
}
```

2-id:

```
#exemple {  
  border: 2px solid red;  
}
```

3-tag:

```
section {  
  display:flex;  
  align-items:center;  
  justify-content:center;  
}
```



Intro to CSS:Selectors

4-attribute selectors:

Here, you are applying a gray text color and a 5px padding to all input elements of type "text."

```
<input type="text" name="username">  
<input type="password" name="password">  
<input type="submit" value="Submit">
```

```
input[type="text"] {  
  color: gray;  
  padding: 5px;  
}
```




Intro to CSS:Selectors

5-Descendant selector:

In this case, `div p` will apply a blue text color to all descendants of the `div`, whether they are direct or indirect. So, all three paragraphs will have blue text.

- **Direct Descendant**: An element that is immediately inside another element, one level below in the HTML structure. In this example, paragraphs 1 and 3 are direct descendants.
- **Indirect Descendant**: An element that is inside another element, but not necessarily at an immediately lower level. It can be nested deeper in the HTML structure. In this example, paragraph 2 is an indirect descendant.

```
<div class="container">
  <h1>Titre</h1>
  <p>Paragraphe 1</p>
  <div>
    <p>Paragraphe 2</p>
  </div>
  <p>Paragraphe 3</p>
</div>
```

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



Intro to CSS:Selectors

6-Pseudo-classes Selector:

A CSS pseudo-class is a selector that allows you to target specific states or characteristics of an HTML element. It is used to apply styles to an element when certain conditions are met. We'll explore pseudo-classes later.

So, following the example, the button will have a pointer cursor, white text, and a blue color when the cursor hovers over it. This effect is achieved using the `:hover` pseudo-class, which applies the specified styles when the mouse cursor hovers over the button element.

```
<button>Appuyer ici</button>
```

```
button:hover {  
  cursor:pointer;  
  background-color: blue;  
  color: white;  
}
```



Intro to CSS:Selectors

7-Combinations selectors:

Indeed, there are many CSS selectors that allow for more precise targeting of elements for specific styling. Here's an overview of the various selector combinations:

1. **Child Selector**: ``parent > child`` selects the direct child elements of a specified parent element.
2. **Descendant Selector**: ``ancestor descendant`` selects descendant elements of a specified ancestor element, regardless of the depth level.
3. **Adjacent Sibling Selector**: ``element1 + element2`` selects ``element2`` that immediately follows ``element1``.
4. **General Sibling Selector**: ``element1 ~ element2`` selects all ``element2`` elements that follow ``element1``, regardless of immediate proximity.
5. **Multiple Elements Selector**: ``element1, element2`` selects multiple elements by specifying multiple selectors separated by commas.
6. **Negation Selector**: ``:not(selector)`` selects elements that do not match the specified selector.

These selector combinations provide a wide range of options for selecting and styling specific elements within your HTML documents.



Intro to CSS:Selectors

Here are some more advanced CSS selectors for targeting elements based on their position within their siblings:

1. **`**`:nth-child(n)` Selector`**`**`:nth-child(n)`` selects elements based on their position among their siblings. You can specify a numerical value for ``n`` to select elements at a specific position.
2. **`**`:nth-last-child(n)` Selector`**`**`:nth-last-child(n)`` selects elements based on their position among their siblings, counting from the end of the list.
3. **`**`:first-child` Selector`**`**`:first-child`` selects elements that are the first child of their parent.
4. **`**`:last-child` Selector`**`**`:last-child`` selects elements that are the last child of their parent.

These selectors offer precise control over styling elements in relation to their sibling elements, making it useful for more complex styling scenarios.

Intro to CSS:Selectors

Here's an example where we apply everything we've seen so far

```
<body>
  <div>
    <h1>Titre</h1>
    <p>Paragraphe 1</p>
    <section>
      <p>Paragraphe 2</p>
    </section>
    <p>Paragraphe 3</p>
  </div>

  <h2>Sous-titre</h2>
  <p>Paragraphe suivant le sous-titre</p>
  <p>Paragraphe suivant le sous-titre</p>

  <ul>
    <li>Élément 1</li>
    <li>Élément 2</li>
    <li>Élément 3</li>
    <li>Élément 4</li>
    <li>Élément 5</li>
  </ul>

  <div class="special">test</div>
  <p>test ul - p</p>
  <p>test ul - p</p>
  <p>test ul - p</p>
  <p>test ul - p</p>
</body>
```

```
/* Exemple 1 - Sélecteur d'enfant direct */
div > p {
  color: blue;
}

/* Exemple 2 - Sélecteur de descendant */
div p {
  font-weight: bold;
  text-decoration: underline;
}

/* Exemple 3 - Sélecteur d'élément + élément */
h2 + p {
  font-size: 40px;
  border: 2px solid red;
}

/* Exemple 4 - Sélecteur d'élément ~ élément */
ul ~ p {
  text-decoration: underline;
  color: brown;
}

/* Exemple 6 - Sélecteur de plusieurs éléments différents en meme temps */
h1,
h2 {
  color: red;
}

/* Exemple 7 - Sélecteur d'élément :not() */
div:not(.special) {
  border: 1px solid gray;
}

/* Exemple 8 - Sélecteur d'élément :nth-child() */
ul li:nth-child(2) {
  background-color: lightgray;
}

/* Exemple 9 - Sélecteur d'élément :nth-last-child() */
ul li:nth-last-child(2) {
  color: purple;
}

/* Exemple 10 - Sélecteur d'élément :first-child */
ul li:first-child {
  background-color: cadetblue;
}

/* Exemple 11 - Sélecteur d'élément :last-child */
ul li:last-child {
  text-align: right;
}
```



Intro to CSS: Priorities in CSS

When multiple attributes are applied with different selectors to the same element, the browser follows a set of priorities to determine which attributes will be applied. Here they are:

1. **Styles with !important**: Any attribute with `!important` has the highest priority and will override all other code for the same attribute.

So, if an attribute is marked with `!important`, it takes precedence over any conflicting styles, and its value will be applied.



```
p {  
  color: red !important;  
}
```



Intro to CSS: Priorities in CSS

2. **Inline Styles**:

Styles defined directly within the HTML element's attributes have higher priority than styles defined in external or internal style sheets. Inline styles are applied directly to the element they are defined on.

3. **ID Selectors**:

Styles applied to elements targeted by an ID selector have higher priority than class selectors or element selectors. ID selectors are more specific and, as a result, take precedence.

4. **Class, Attribute, and Pseudo-Class Selectors**:

```
<p style="color: blue;">Texte en bleu</p>
```

```
#monId {  
  color:blue;  
}
```

```
.nomClasse {  
  color:blue;  
}
```

Intro to CSS: Priorities in CSS

5. **Combined Element Selectors
(Descendant, Child, Sibling)**

6. **Element Selectors**

7. **Universal Selectors (*)**

8. **Pseudo-Element Selectors (::before,
::after)**

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

```
body {  
  background-color: blue;  
}
```

```
* {  
  margin: 0;  
  padding: 0;  
}
```

```
p::before {  
  content: "Avant le paragraphe";  
}
```




Intro to CSS: CSS basic attributes

1-color:

We use it to apply a color to a text and there is 4 different types:

- **Color Name** (for eg: blue, black, azur,...)
- In **hexadecimal** (for eg #FF0000)
- In **RGB** (Red Green Blue)
- In **HSL** (Hue Saturation Brightness)

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

```
div p {  
  color: #0000FF;  
}
```

```
div p {  
  color: rgb(0, 0, 255);  
}
```

```
div p {  
  color: hsl(240, 100%, 50%);  
}
```



Intro to CSS: CSS basic attributes

2-background-color:

Do the same job as the color but for backgrounds

2-width & height:

these two properties are used to specify the width and height of a tag

For the units used you gonna see in the following slides



```
.home {  
background-color:blue;  
height : 400px;  
width : 100%;  
}
```

Intro to CSS:box model(margin,border,padding,content)

Every HTML element (each tag) follows the same structure, and it's this structure that allows us to control various specifications, including the space between elements and their borders. Understanding this structure helps us comprehend how our HTML is organized and how to better position our elements.

So, for each HTML element, the order of properties includes:

Margin,Border,Padding,Content

This order determines the layout and spacing of elements in HTML and allows for precise control over their appearance and positioning.





Intro to CSS:box model(margin,border,padding,content)

1-Margin:

Margins are used to create space around elements, outside of any defined border. They allow you to specify the distance between that element and other adjacent elements. Margins are essential for controlling the layout and spacing between elements on a webpage.

A-Button without

margin:

welcome back

read more

B-Button with

margin:

welcome back

read more



Intro to CSS:box model(margin,border,padding,content)

2-Border:

It is the line that surrounds the content and padding. It can be made visible or invisible, and by default, it doesn't appear. It can have different thicknesses, colors, and styles.

In CSS, this "line" is referred to as the "border," and it plays a significant role in defining the appearance and structure of elements on a webpage.

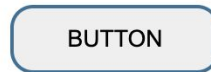
A-Button without

border:



B-Button with

border:



Intro to CSS:box model(margin,border,padding,content)

3-Padding:

It is used to create transparent space around the content of an element, inside any defined border. Many people confuse it with margin, but they are not the same:

In CSS, this transparent space is referred to as "padding," and it's distinct from margins. Padding defines the space between an element's content and its border, whereas margins define the space outside the border, affecting the distance between elements.

Padding creates space inside the specified element, while margin applies space outside of it.

In CSS, this fundamental distinction is crucial for controlling the layout and spacing of elements within a webpage. Padding affects the internal space of an element, while margins affect the space outside of the element.

A-Button without

Padding:

hello world

B-Button with

Padding:

hello world

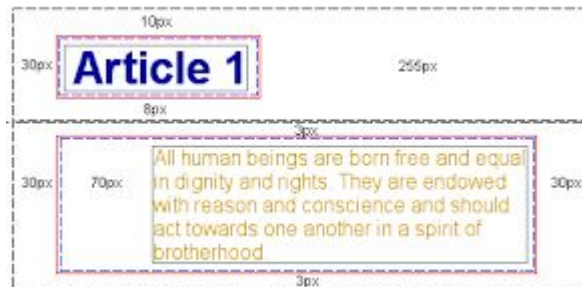
Intro to CSS:box model(margin,border,padding,content)

4-Content:

It is the actual content of the element, such as text, images, or other HTML elements it contains. It's what you place inside your HTML tags.

For example:

Here, we have two different tags. The first one has the content "Article 1," and the second one contains a paragraph written in yellow.





End of the first part of the workshop!

If you have any questions don't hesitate!

salimghalem40@gmail.com

Thanks to you all!