



Politecnico
di Torino

Introduzione alle Applicazioni Web

CSS

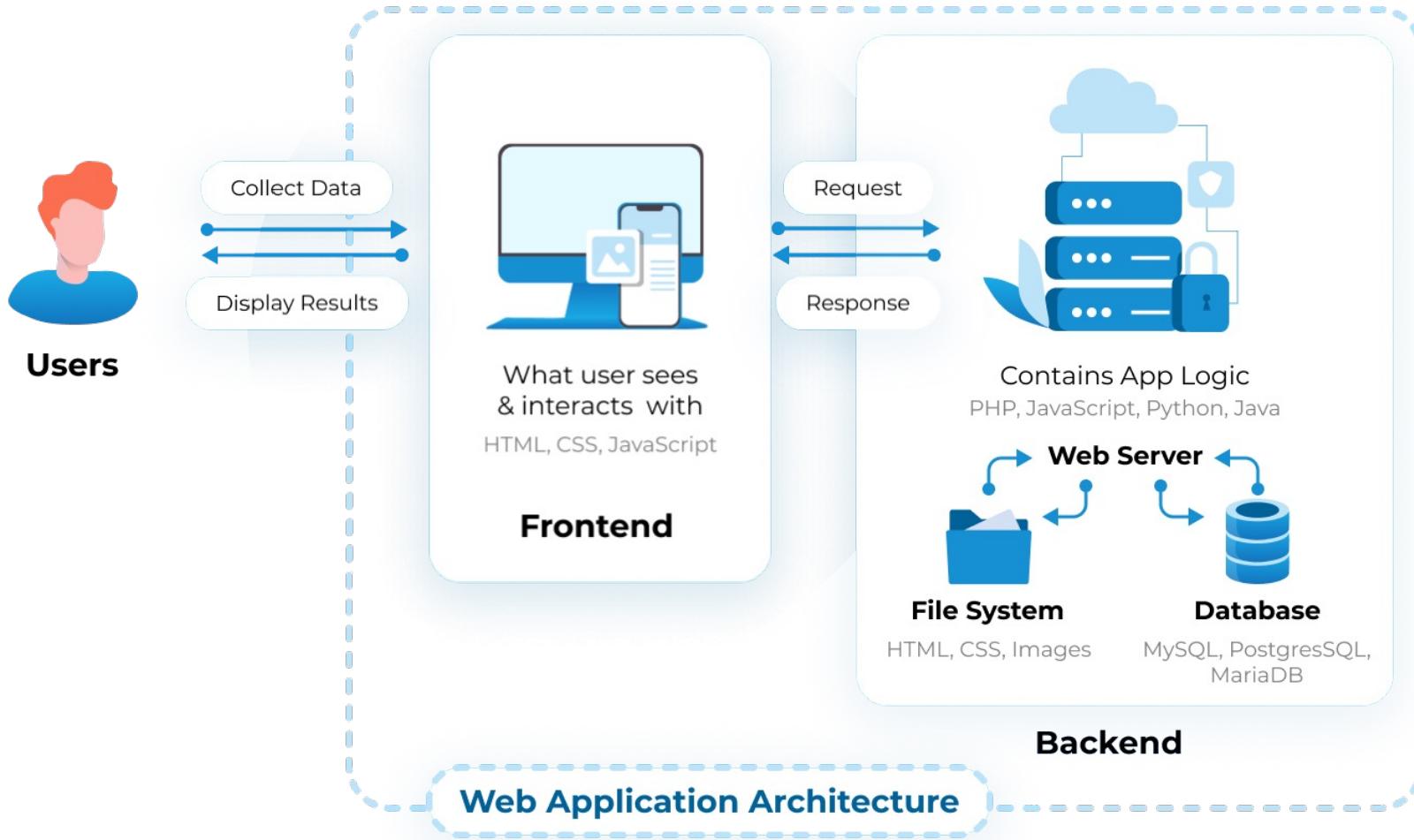
Juan Pablo Sáenz



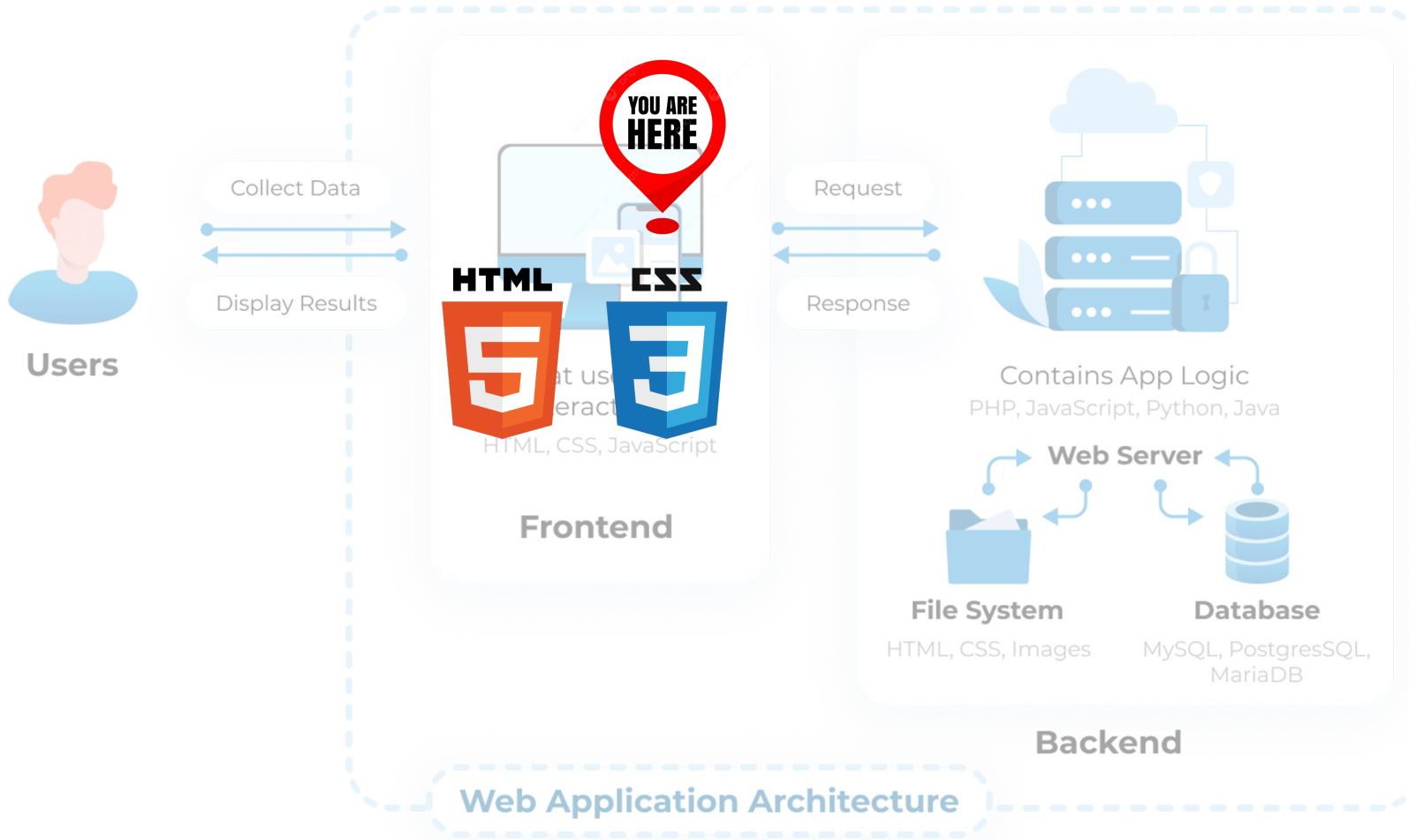
Goals

- Style web content to enhance **visual presentation**.
- Understand the fundamentals of **Cascading Style Sheets (CSS)**
- Explore key CSS concepts and **best practices**
- **Apply CSS** effectively on web pages

📍 CSS: where are we?



📍 CSS: where are we?



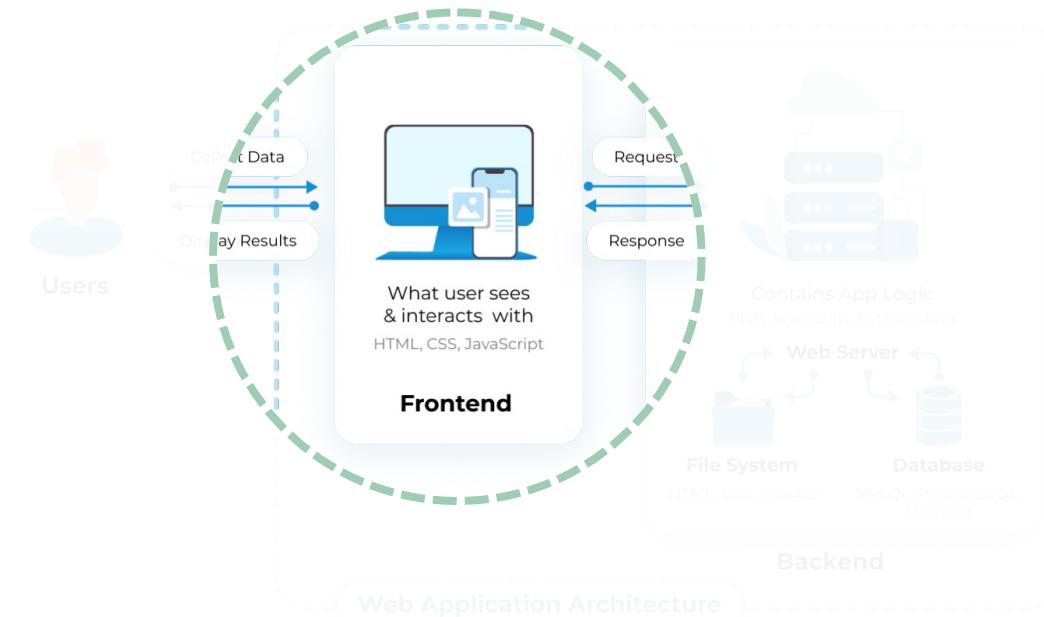


Web architecture components: Frontend

Frontend: what the user sees and interacts with

Languages:

- **HTML:** a markup language used to **structure content on the web**. It defines elements like headings, paragraphs, images, and links.
- **CSS:** a style sheet language used to **control the presentation of HTML elements**, such as colors, fonts, and layouts.
- **JavaScript:** a programming language used to **add interactivity and dynamic behavior** to web pages.



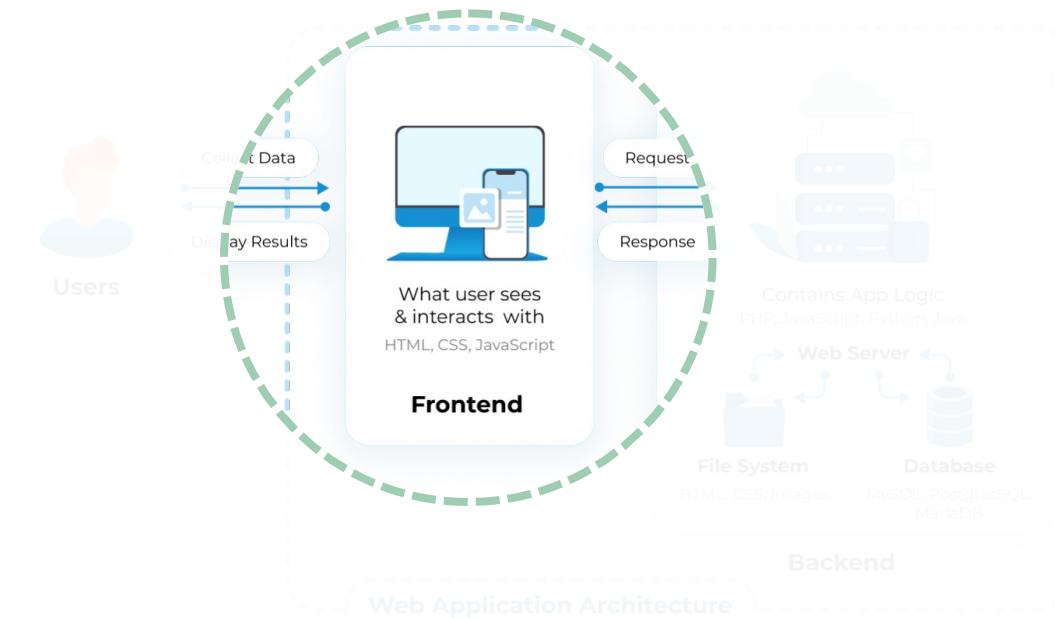


Web architecture components: Frontend

Frontend: what the user sees and interacts with

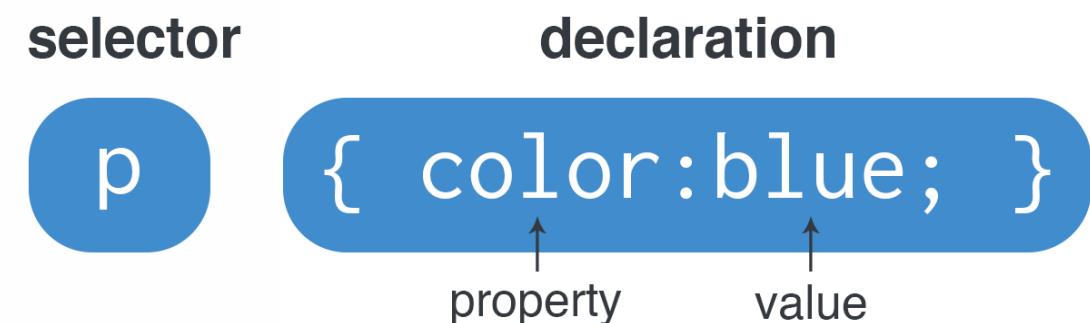
Applications:

- **Browser:** an application that **retrieves, interprets, and displays web content**, including HTML, CSS, and JavaScript.



CSS Syntax

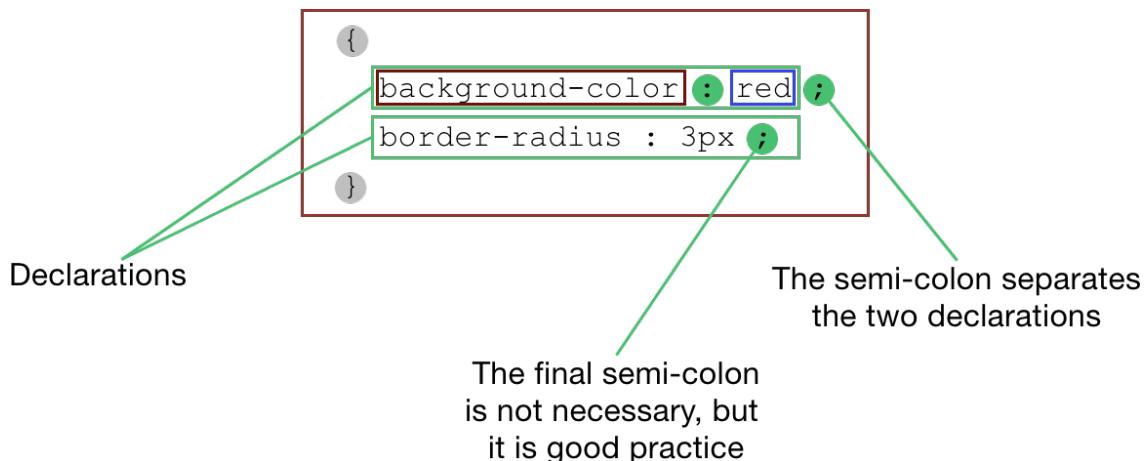
- CSS **uses rules** to style HTML elements.
 - A rule defines a **visual property** for one or more **elements**.
 - Each rule consists of a **selector** and **declarations** (styles).



<https://devdojo.com/guide/css/syntax>

CSS Syntax

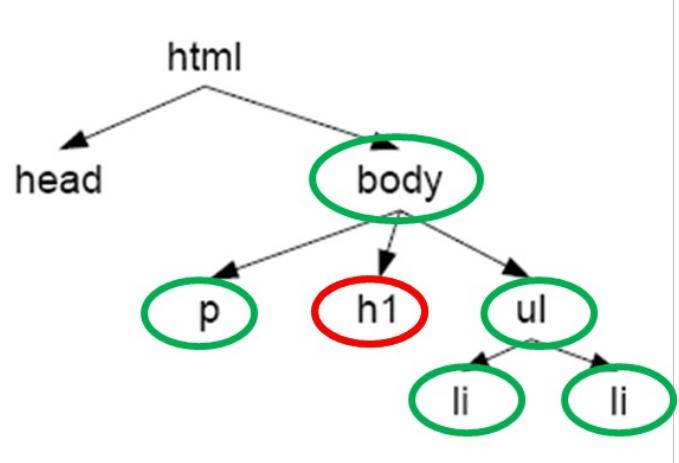
- CSS **uses rules** to style HTML elements.
- A rule defines a **visual property** for one or more **elements**.
- Each rule consists of a **selector** and **declarations** (styles).



```
header, p.intro {  
    background-color: red;  
    border-radius: 3px;  
}
```

CSS Syntax

- HTML documents have a **tree structure**.
- **Styles are inherited** along the tree.
- When rules conflict, the **most specific** one takes **precedence**.



```
body {  
    color: green  
}  
  
h1 {  
    color: red  
}
```

CSS Properties

- Animation Properties
- Background Properties
- Border Properties
- Color Properties
- Dimension Properties
- Generated Content Properties
- Flexible Box Layout
- Font Properties
- List Properties
- Margin Properties
- Multi-column Layout Properties
- Outline Properties
- Padding Properties
- Print Properties
- Table Properties
- Text Properties
- Transform Properties
- Transitions Properties
- Visual formatting Properties

CSS Properties



The **MDN Web Docs** site provides information about Open Web technologies, including **HTML**, **CSS**, and APIs for both Web sites and progressive web apps.

- <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>

CSS Units

There are two types of length units

- **Absolute** (fixed)

The most common fixed unit is **pixel** (also pt, pc, in, cm, mm).

⚠ They are **relative** to the **viewing device** 😬.

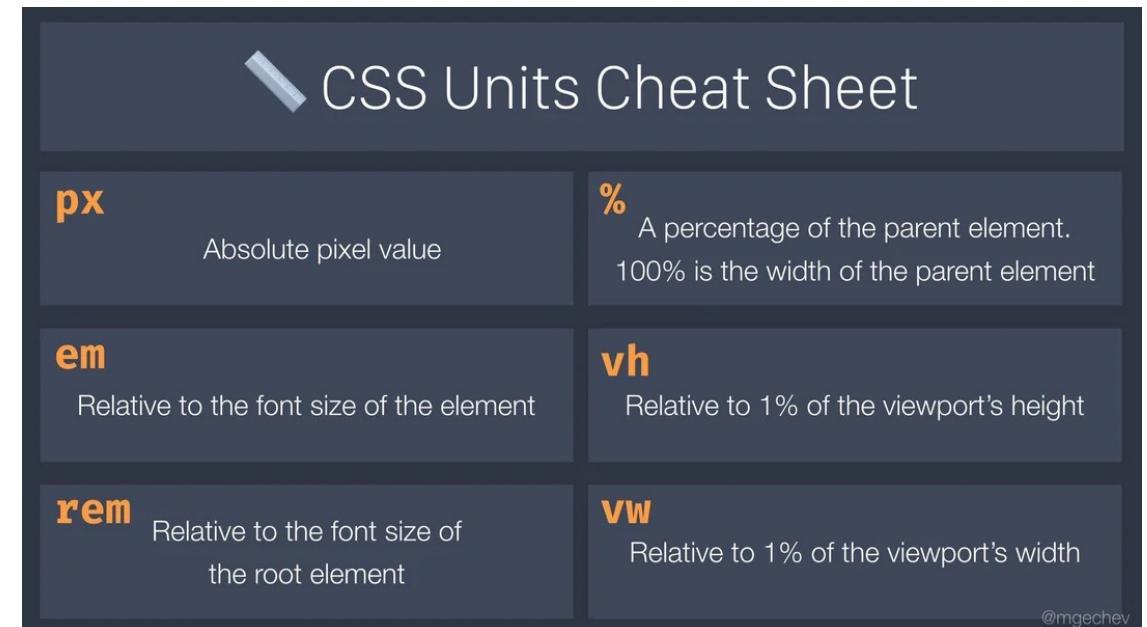
- **Relative**

```
header {  
    width: 1000px;  
}
```

CSS Units

Relative

- **em**: relative to the **font size** of the element. 2em means 2 times the **font size** of the **current element**.
- **rem**: relative to the **font size** of the **root element** of the HTML page (<html>).
- **vw**: relative to **1%** of the **width** of the **viewport**.
- **vh**: relative to **1%** of the **height** of the **viewport**.
- **%: percentage** relative to the **parent element**.



@mgechev

CSS Units

Relative

- **em**: relative to the **font size** of the element. 2em means 2 times the **font size** of the **current element**.
- **rem**: relative to the **font size** of the **root element** of the HTML page (<html>).
- **vw**: relative to **1%** of the **width** of the **viewport**.
- **vh**: relative to **1%** of the **height** of the **viewport**.
- **%: percentage** relative to the **parent element**.

JULIA EVANS
@bork

CSS has 2 kinds of units:
absolute & relative

absolute: px, pt, pc,
in, cm, mm

relative: em, rem,
vw, vh, %

CSS units

rem

the root element's
font size

1rem is the same
everywhere in the
document. rem is a
good unit for setting
font sizes!

em

the current element's
font size

heading
tiny text
these 2 elements
have different
values of 1em

0 is the same
in all units

```
.btn {  
    margin: 0;  
}
```

also, 0 is different from none.
border: 0 sets the border width
and border: none sets the style

1 inch = 96 px

on a screen, 1 CSS "inch"
isn't really an inch, and
1 CSS "pixel" isn't really
a screen pixel.
look up "device pixel
ratio" for more.

rem & em help with
accessibility

```
.modal {  
    width: 20rem;  
}
```

this scales nicely if the user
increases their browser's
default font size

CSS Units

Suggestions:

- Prefer **relative units** over absolute ones when possible.
- **rem** is nowdays preferred over **em**.

CSS Selectors

Patterns for **selecting elements** to style.

There are three main types of **selectors** and two
'pseudo-selectors':

1. **Element** selector: `element`

```
/* Element selector: used to apply  
the same style to all instances of  
a specific element in a document */  
  
header {  
    width: 1000px;  
}
```

@mgechev

CSS Selectors

Patterns for **selecting elements** to style.

There are three main types of **selectors** and two '**pseudo-selectors**:

2. Class selector: .class

```
/* Class selector: apply the same style to  
all elements belonging to a specific class  
*/  
  
.bluetext {  
    color: blue;  
}
```

```
<body>  
    <h1>Hola amigos cómo están</h1>  
    <p class="bluetext">IAW</p>  
    <p>Esto es un <a href="demo.html">link</a>  
    </p>  
</body>
```

CSS Selectors

Patterns for **selecting elements** to style.

There are three main types of **selectors** and two '**pseudo-selectors**:

3. ID selector: #id

```
/* ID selector: apply a style to a
specific element in the document */

#greetings {
    color: gray;
}
```

```
<body>
    <h1 id="greetings">Hola amigos cómo están
    </h1>
    <p class="bluetext">IAW</p>
    <p class="bluetext">Esto es un <a
    href="demo.html">link</a>
    </p>
</body>
```

CSS Selectors

Patterns for **selecting elements** to style.

There are three main types of **selectors** and two '**pseudo-selectors**:

4. Attribute selector: [name=val]

```
/* The element has an href attribute */  
a[href] {}  
  
/* The element has this exact href attribute value */  
a[href="https://www.polito.it/didattica"] {}  
  
/* The href attribute value includes the string .com */  
a[href*=".it"] {}  
  
/* The href value starts with the string https: */  
a[href^="https:"] {}  
  
/* The href value ends with the string /dev-tips */  
a[href$="/didattica"] {}
```

CSS Selectors

Patterns for **selecting elements** to style.

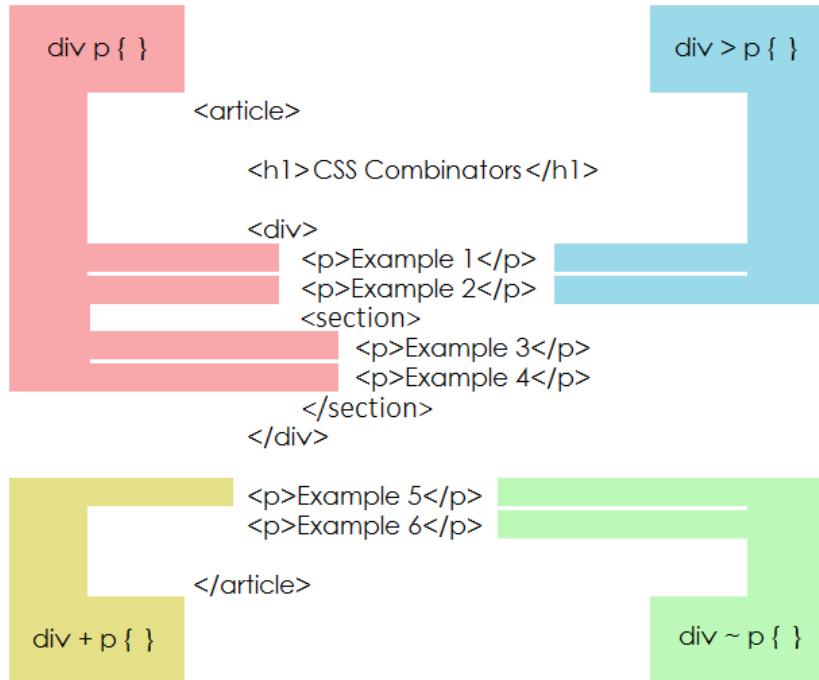
There are three main types of **selectors** and two '**pseudo-selectors**:

5. Pseudo selector: :something

Used to style an element based on something other than the structure of the document.

```
/* Makes all unvisited links blue */  
a:link {color:blue;}  
  
/* Makes all visited links green */  
a:visited {color:green;}  
  
/* Makes links red when hovered or activated */  
a:hover, a:active {color:red;}  
  
/* Makes table rows red when hovered over */  
tr:hover {background-color: red;}  
  
/* Makes input elements yellow when focus is applied */  
input:focus {background-color:yellow;}
```

CSS Combinators



```
/* Selects all <div> elements and all <p> elements */
div, p {}

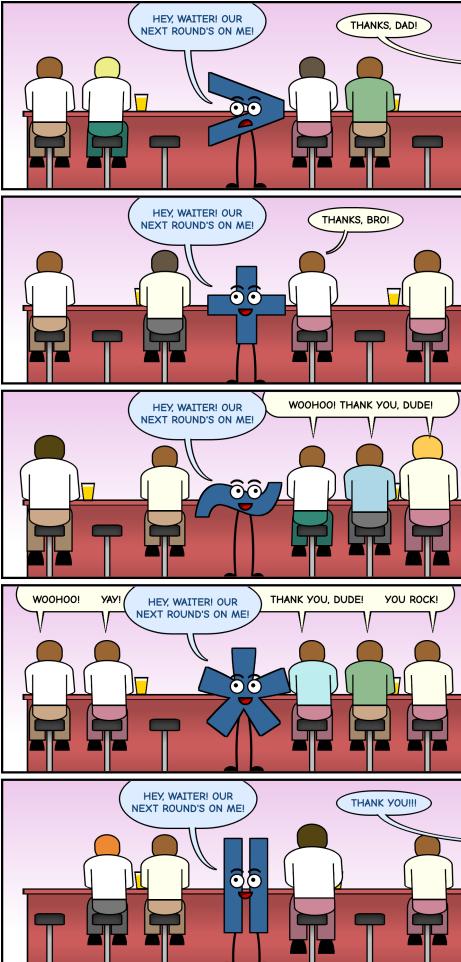
/* Selects all <p> elements inside <div> elements */
div p {}

/* Selects every <p> element that are direct children
   of a <div> element */
div > p {}

/* Selects the first <p> element that is placed
   immediately after <div> elements */
div + p {}

/* Selects all <ul> elements that are preceded by a
   <p> element */
p ~ ul {}
```

CSS Combinators



```
/* Selects all <div> elements and all <p> elements */
div, p {}

/* Selects all <p> elements inside <div> elements */
div p {}

/* Selects every <p> element that are direct children
of a <div> element */
div > p {}

/* Selects the first <p> element that is placed
immediately after <div> elements */
div + p {}

/* Selects all <ul> elements that are preceded by a
<p> element */
p ~ ul {}
```

CSS: Display property

- Controls element display (**block** or **inline**)
- Changing an inline element to block, or vice versa, helps **adjust page layout**.



We'll see an example of how to use it!

```
li {display: inline;}
```

```
span {display: block;}
```

CSS: Display property

The property **display** allows to hide an element, too.

- The element will be hidden, and the page will be displayed **as if the element is not there**.

The property **visibility** also can hide an element, but the element will **still take up the same space** as before.

- The element will be hidden, but still affects the layout.

```
h1.hidden {  
    display: none;  
}  
  
h1.hidden {  
    visibility: hidden;  
}
```

Applying CSS

You can **apply CSS styles to an HTML document** in three main ways:

1. **Inline** CSS 💀:

Add the **style** attribute directly to an element

2. Internal CSS:

Use a **<style>** block inside the **<head>** section of the HTML file

3. External CSS:

Link to an **external .css file** using **<link>** in the **<head>**.

```
<p style="color: blue; font-size: 16px;">  
    Hello, world!  
</p>
```

Applying CSS

You can **apply CSS styles to an HTML document** in three main ways:

1. Inline CSS:

Add the **style** attribute directly to an element

2. Internal CSS:

Use a **<style>** block inside the **<head>** section of the HTML file

3. External CSS:

Link to an **external .css file** using **<link>** in the **<head>**.

```
<head>
  <style>
    p {
      color: blue;
      font-size: 16px;
    }
  </style>
</head>
```

Applying CSS

You can **apply CSS styles to an HTML document** in three main ways:

1. Inline CSS:

Add the **style** attribute directly to an element

2. Internal CSS:

Use a **<style>** block inside the **<head>** section of the HTML file

3. External CSS 🙌 ✅:

Link to an **external .css file** using **<link>** in the **<head>**.

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

```
/* styles.css */
p {
  color: blue;
  font-size: 16px;
}
```

Applying CSS

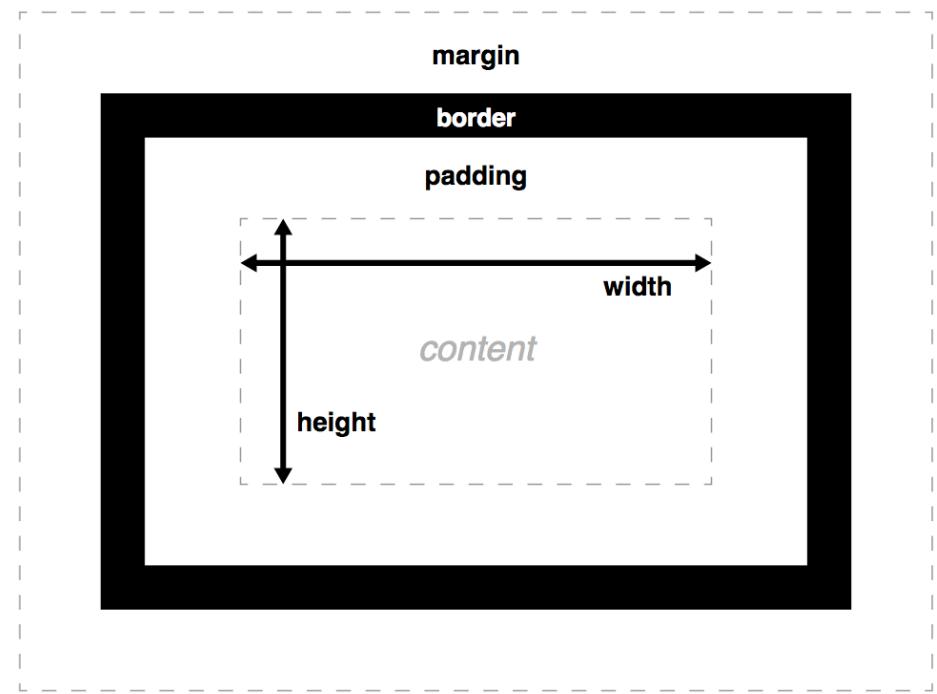
💡 Best Practice:

Use **external CSS** for better **maintainability** and **reusability!** 🚀

Let's see it in practice

CSS Box model

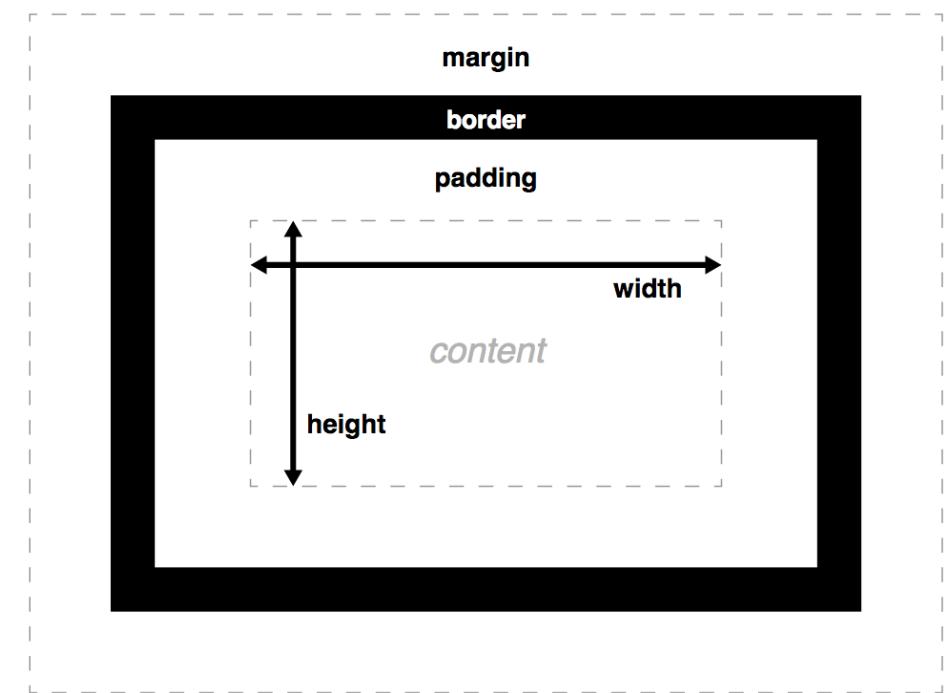
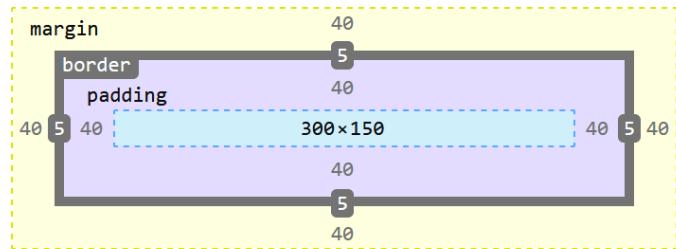
- A fundamental concept in CSS.
- Every element on the page is treated as a **rectangular box**.



https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_box_model

CSS Box model

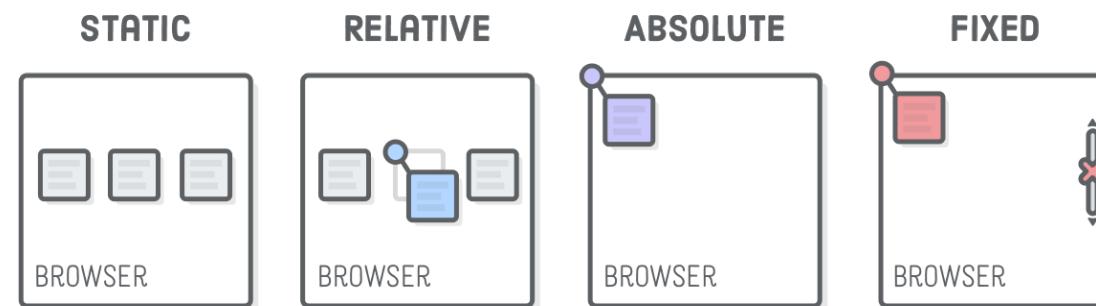
- **Total element width =**
width + left padding + right padding + left border + right border + left margin + right margin
- **Total element height =**
height + top padding + bottom padding + top border + bottom border + top margin + bottom margin
- Each property can be set independently.



https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_box_model

CSS Positioning schemes

- **Static:** The default position, follows the normal document flow.
- **Relative:** Positioned relative to **its original position** in the normal flow.
- **Absolute:** Positioned using the top, left, right, and bottom properties, relative to the **nearest positioned ancestor**.
- **Fixed:** Positioned relative to the **viewport**, remaining fixed even when scrolling.



<https://internetengineeringishard.com/html-and-css/advanced-positioning/>

CSS Positioning schemes: Relative

Relative: Positioned relative to **its original position** in the normal flow.

- An element can be shifted **relative** to its normal flow position by applying vertical and/or horizontal offsets.



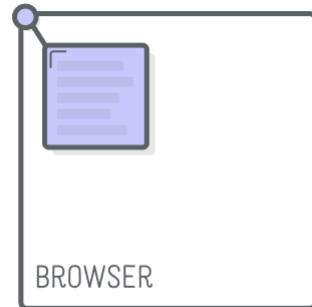
```
.item-relative {  
    position: relative;  
    left: 20px;  
    top: 20px;  
}
```

<https://internetingishard.com/html-and-css/advanced-positioning/>

CSS Positioning schemes: Absolute

Absolute: Positioned using the top, left, right, and bottom properties, relative to the **nearest positioned ancestor**.

- It **removes** the element from the document flow, meaning it takes up no space.
- Other elements in the normal flow will behave as if the absolutely positioned element doesn't exist.



ABSOLUTE POSITIONING

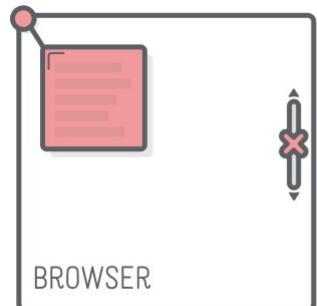
<https://internetingishard.com/html-and-css/advanced-positioning/>

```
.item-absolute {  
    position: absolute;  
    left: 20px;  
    top: 20px;  
}
```

CSS Positioning schemes: Fixed

Fixed: Positioned relative to the **viewport**, remaining fixed even when scrolling.

- Like absolute positioning, the element is **removed** from the normal flow, relative to the entire browser window.
- The key difference is that fixed elements remain in place **when the page scrolls**.



FIXED POSITIONING

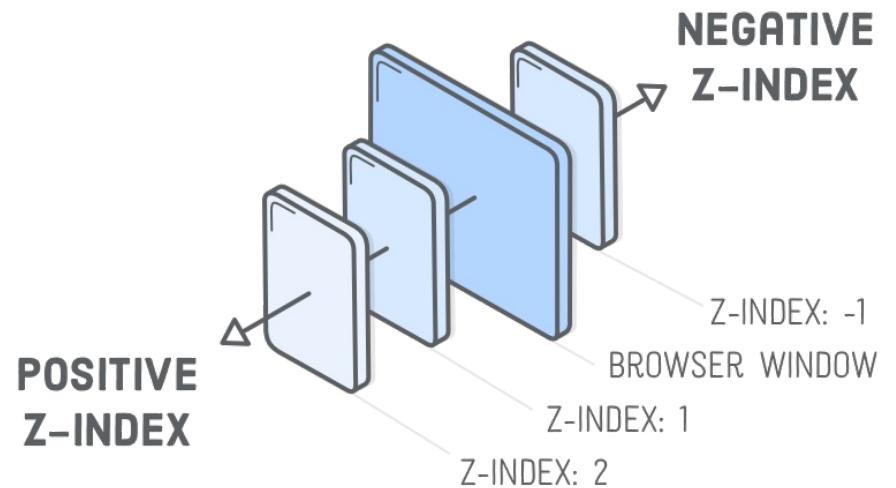
<https://internetingishard.com/html-and-css/advanced-positioning/>

```
.item-fixed {  
    position: fixed;  
    left: 20px;  
    top: 20px;  
}
```

CSS Positioning: z-index

In case of **overlaps**:

The **z-index** property determines the stack order of elements, specifying which should appear **in front of** or **behind** others.



<https://internetgingishard.com/html-and-css/advanced-positioning/>



Licenza

- These slides are distributed under a Creative Commons license "**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**"
- **You are free to:**
 - **Share** – copy and redistribute the material in any medium or format
 - **Adapt** – remix, transform, and build upon the material
 - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
 - **Attribution** – You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - **NonCommercial** – You may not use the material for commercial purposes.
 - **ShareAlike** – If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
 - **No additional restrictions** – You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- <https://creativecommons.org/licenses/by-nc-sa/4.0/>