



# INTRODUCTION TO CSS

---

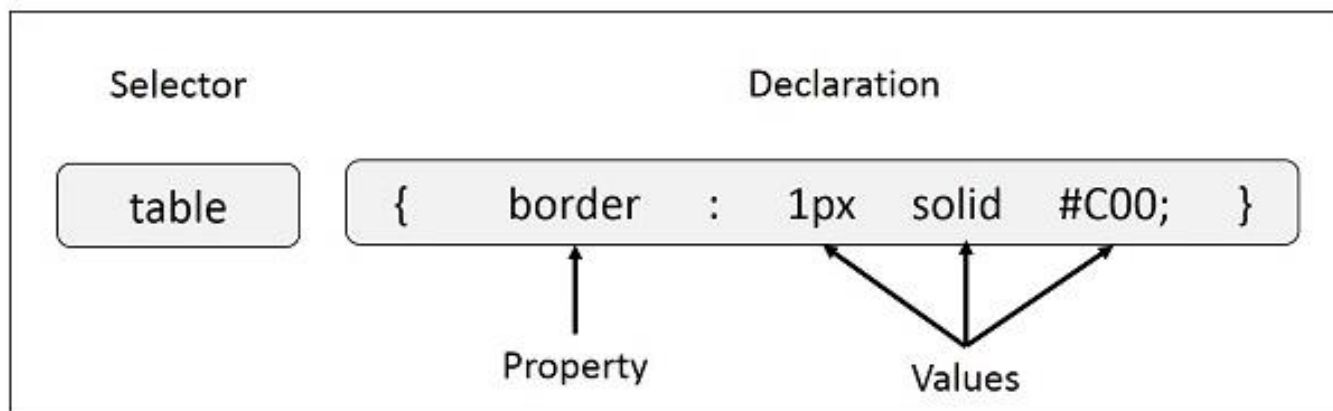
Presented by  
Eng./Abanoub Nabil  
Teaching assistant at ITI



# Introduction...

- **CSS** is used to control the style of a web document in a simple and easy way.
- **CSS** is the acronym for "**Cascading Style Sheet**".
- It is used with various **HTML elements** to stylize and format them.
- CSS **describes** how **HTML elements** should be displayed.

# CSS Syntax





# CSS Syntax Example.

```
h2 {  
  color: #000000;  
}
```



# CSS Syntax Example.

```
h2{  
  color: #000000;  
}
```

SELECTOR



# CSS Syntax Example.

```
h2 {  
  color: #000000;  
}
```

**CURLY BRACKETS**



# CSS Syntax Example.

```
h2 {  
  color: #000000;  
}
```

PROPERTY



# CSS Syntax Example.

```
h2 {  
  color: #000000;  
}
```

VALUE





# CSS Selectors.

- CSS selectors are used to "**find**" (or select) the HTML elements you want to style.

**We can divide CSS selectors into five categories:**

- 1- **Simple selectors** (select elements based on name, id, class).
- 2- **Combination selectors** (select elements based on a specific relationship between them).
- 3- **Pseudo-class selectors** (select elements based on a certain state)
- 4- **Pseudo-elements selectors** (select and style a part of an element)
- 5- **Attribute selectors** (select elements based on an attribute or attribute value)



# CSS Selectors (element Selector)

**h2 {**

**font-style: bold;**

**}**

**p {**

**font-family: Helvetica;**

**}**

```
<body>  
  <div id="container">
```

```
    <h2 class="headline">Hello World</h2>
```

```
    <p class="byline">By Jeremy Rue.</p>
```

```
    <p>In today's news, a class at UC Berkeley  
    learned the importance of CSS in designing  
    and building webpages.</p>
```

```
  <div id="sidebar">
```

```
    <h2>Side Bar Story</h2>
```

```
    <p class="byline">By John Doe.</p>
```

```
    <p>This is a related story.</p>
```

```
  </div>
```

```
</div>
```

```
</body>
```



# CSS Selectors (element Selector)

```
h2 {  
  
  font-style: bold;  
  
}
```

```
p {  
  
  font-family: Helvetica;  
  
}
```

```
<body>  
  <div id="container">  
  
    <h2 class="headline">Hello World</h2>  
  
    <p class="byline">By Jeremy Rue.</p>  
  
    <p>In today's news, a class at UC Berkeley  
    learned the importance of CSS in designing  
    and building webpages.</p>  
  
    <div id="sidebar">  
      <h2>Side Bar Story</h2>  
      <p class="byline">By John Doe.</p>  
      <p>This is a related story.</p>  
    </div>  
  
  </div>  
</body>
```



# CSS Selectors (class Selector)

**.headline {**

**text-decoration: underline;**

**}**

**.byline {**

**font-size: 8px;**

**}**

```
<body>  
  <div id="container">
```

```
    <h2 class="headline">Hello World</h2>
```

```
    <p class="byline">By Jeremy Rue.</p>
```

```
    <p>In today's news, a class at UC Berkeley  
    learned the importance of CSS in designing  
    and building webpages.</p>
```

```
  <div id="sidebar">
```

```
    <h2>Side Bar Story</h2>
```

```
    <p class="byline">By John Doe.</p>
```

```
    <p>This is a related story.</p>
```

```
  </div>
```

```
</div>
```

```
</body>
```



# CSS Selectors (class Selector)

```
.headline {  
  
  text-decoration: underline;  
  
}
```

```
.byline {  
  
  font-size: 8px;  
  
}
```

```
<body>  
  <div id="container">  
  
    <h2 class="headline">Hello World</h2>  
  
    <p class="byline">By Jeremy Rue.</p>  
  
    <p>In today's news, a class at UC Berkeley  
    learned the importance of CSS in designing  
    and building webpages.</p>  
  
    <div id="sidebar">  
      <h2>Side Bar Story</h2>  
      <p class="byline">By John Doe.</p>  
      <p>This is a related story.</p>  
    </div>  
  
  </div>  
</body>
```



# CSS Selectors (id Selector)

```
#container {  
  font-size: 30px;  
}
```

```
#sidebar {  
  font-size: 8px;  
}
```

```
<body>  
  <div id="container">  
    <h2 class="headline">Hello World</h2>  
    <p class="byline">By Jeremy Rue.</p>  
    <p>In today's news, a class at UC Berkeley  
    learned the importance of CSS in designing  
    and building webpages.</p>  
    <div id="sidebar">  
      <h2>Side Bar Story</h2>  
      <p class="byline">By John Doe.</p>  
      <p>This is a related story.</p>  
    </div>  
  </div>  
</body>
```



# CSS Selectors (id Selector)

```
#container {  
  
  font-size: 30px;  
  
}
```

```
#sidebar {  
  
  font-size: 8px;  
  
}
```

```
<body>  
  <div id="container">  
  
    <h2 class="headline">Hello World</h2>  
  
    <p class="byline">By Jeremy Rue.</p>  
  
    <p>In today's news, a class at UC Berkeley  
    learned the importance of CSS in designing  
    and building webpages.</p>  
  
    <div id="sidebar">  
      <h2>Side Bar Story</h2>  
      <p class="byline">By John Doe.</p>  
      <p>This is a related story.</p>  
    </div>  
  </div>  
</body>
```



# CSS Selectors (Universal Selector)

```
<style>
* {
  text-align: center;
  color: blue;
}
</style>
```

Hello world!

Every element on the page will be affected by the style.

Me too!

And me!





# CSS Selectors (Grouping Selector)

```
<html>
<head>
<style>
h1, h2 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
```

```
<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

Hello World!

Smaller heading!

This is a paragraph.



# CSS Selectors

## All CSS Simple Selectors

Selector	Example	Example description
<u><a href="#">#id</a></u>	#firstname	Selects the element with id="firstname"
<u><a href="#">.class</a></u>	.intro	Selects all elements with class="intro"
<u><a href="#">element.class</a></u>	p.intro	Selects only <p> elements with class="intro"
<u><a href="#">*</a></u>	*	Selects all elements
<u><a href="#">element</a></u>	p	Selects all <p> elements
<u><a href="#">element,element,..</a></u>	div, p	Selects all <div> elements and all <p> elements



# CSS Selectors (Combinators)

- A combinator is something that explains the relationship between the selectors.
- A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.
- There are four different combinators in CSS:
  - 1- descendant selector (space)
  - 2- child selector (>)
  - 3- adjacent sibling selector (+)
  - 4- general sibling selector (~)



# CSS Selectors (Combinators) cont.

- **Descendant Selector**
- The descendant selector matches all elements that are descendants of a specified element.
- The following example **selects all `<p>` elements inside `<div>` elements:**



# CSS Selectors (Combinators) cont.

- **Descendant Selector**

```
<html>
<head>
<style>
div p {
  background-color: lightgreen;
  font-style: italic;
}
</style>
</head>
<body>
```

```
<h2>Descendant Selector</h2>
```

```
<p>The descendant selector matches all elements that are descendants of
a specified element.</p>
```

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>
```

```
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
```

```
</body>
</html>
```



# CSS Selectors (Combinators) cont.

- **Descendant Selector**

## Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

*Paragraph 1 in the div.*

*Paragraph 2 in the div.*

*Paragraph 3 in the div.*

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.



# CSS Selectors (Combinators) cont.

- **Child Selector (>)**
- The child selector selects all elements that are the children of a specified element.
- The following example selects all **<p> elements that are children of a <div> element:**



# CSS Selectors (Combinators) cont.

```
<html>
<head>
<style>
div > p {
  background-color: yellow;
}
</style>
</head>
<body>
```

<h2>Child Selector</h2>

<p>The child selector (>) selects all elements that are the children of a specified element.</p>

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section> <!-- not Child but Descendant -->
  <p>Paragraph 4 in the div.</p>
</div>

<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>

</body>
</html>
```





# CSS Selectors (Combinators) cont.

## Child Selector

The child selector (`>`) selects all elements that are the children of a specified element.

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4 in the div.

Paragraph 5. Not in a div.

Paragraph 6. Not in a div.



# CSS Selectors (Combinators) cont.

## Adjacent Sibling Selector (+)

- The adjacent sibling selector is used to select an element that is directly after another specific element.
- Sibling elements must have the same parent element, and "adjacent" means "immediately following".
- The following example **selects the first <p> element that are placed immediately after <div> elements:**



# CSS Selectors (Combinators) cont.

```
<html>
<head>
<style>
div + p {
  background-color: yellow;
}
</style>
</head>
<body>
```

<h2>Adjacent Sibling Selector</h2>

<p>The + selector is used to select an element that is directly after another specific element.</p>

<p>The following example selects the first p element that are placed immediately after div elements:</p>

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>
```

```
<p>Paragraph 3. After a div.</p>
<p>Paragraph 4. After a div.</p>
```

```
<div>
  <p>Paragraph 5 in the div.</p>
  <p>Paragraph 6 in the div.</p>
</div>
```

```
<p>Paragraph 7. After a div.</p>
<p>Paragraph 8. After a div.</p>
```

```
</body>
</html>
```

---



# CSS Selectors (Combinators) cont.

## Adjacent Sibling Selector

The + selector is used to select an element that is directly after another specific element.

The following example selects the first p element that are placed immediately after div elements:

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. After a div.

Paragraph 4. After a div.

Paragraph 5 in the div.

Paragraph 6 in the div.

Paragraph 7. After a div.

Paragraph 8. After a div.



# CSS Selectors (Combinators) cont.

## General Sibling Selector (~)

- The combiner '~' allows you to **select** nodes which follow an element and which have the same parent.
- The following example **selects all <p> elements that are siblings of <div> elements and follow it.**



# CSS Selectors (Combinators) cont.

```
<html>
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>
```

```
<h2>General Sibling Selector</h2>
```

```
<p>The general sibling selector (~) selects all elements that are siblings of a specified element.</p>
```

```
<p>Paragraph 1.</p>
```

```
<div>
  <p>Paragraph 2.</p>
</div>
```

```
<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>
```

```
</body>
</html>
```



# CSS Selectors (Combinators) cont.

## General Sibling Selector

The general sibling selector (~) selects all elements that are siblings of a specified element.

Paragraph 1.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.



# CSS Selectors (Combinators) cont.

## All CSS Combinator Selectors

Selector	Example	Example description
<a href="#"><u><i>element element</i></u></a>	div p	Selects all <p> elements inside <div> elements
<a href="#"><u><i>element&gt;element</i></u></a>	div > p	Selects all <p> elements where the parent is a <div> element
<a href="#"><u><i>element+element</i></u></a>	div + p	Selects the first <p> element that are placed immediately after <div> elements
<a href="#"><u><i>element1~element2</i></u></a>	p ~ ul	Selects every <ul> element that are preceded by a <p> element





# CSS Selectors ( Pseudo-classes)

- **Pseudo-classes are** used to target elements based on state information that is not stored in the document tree.
- **For example**, it can be used to:
  - 1- Style an element when a user **mouse over** it
  - 2- Style **visited and unvisited** links differently
  - 3- Style an element when it gets **focus**
- **Syntax**
- The syntax of pseudo-classes:

```
selector:pseudo-class {  
    property: value;  
}
```



# CSS Selectors ( Pseudo-classes)cont.

- **Anchor Pseudo-classes**

- ```
/* unvisited link */
a:link {
    color: red;
}

/* visited link */
a:visited {
    color: green;
}

/* mouse over link */
a:hover {
    color: hotpink;
}

/* selected link */
a:active {
    color: blue;
}
```



# CSS Selectors ( Pseudo-classes)cont.

- **Pseudo-classes and CSS Classes**
- Pseudo-classes can be combined with CSS classes:
- When you hover over the link in the example, it will change color:

```
a.highlight:hover {  
    color: #ff0000;  
}
```

---



# CSS Selectors ( Pseudo-classes)cont.

- **Simple Tooltip Hover**
- Hover over a <div> element to show a <p> element (like a tooltip):

```
<html>
<head>
<style>
p {
  display: none;
  background-color: yellow;
  padding: 20px;
}

div:hover p {
  display: block;
}
</style>
</head>
<body>

<div>Hover over me to show the p element
  <p>Tada! Here I am!</p>
</div>
|
</body>
</html>
```



# CSS Selectors ( Pseudo-classes)cont.

- **CSS - The :first-child Pseudo-class**
- The **:first-child** pseudo-class matches a specified element that is the first child of another element.
- **Match the first <p> element**
- In the following example, the selector matches any <p> element that is the first child of any element:

```
<style>
p:first-child {
  color: blue;
}
</style>
</head>
<body>
```

```
<p>This is some text.</p>
<p>This is some text.</p>
```

This is some text.

This is some text.



# CSS Selectors ( Pseudo-classes)cont.

- **Match the first `<i>` element in all `<p>` elements**
- In the following example, the selector matches the first `<i>` element in all `<p>` elements:

```
<head>
<style>
p i:first-child {
  color: blue;
}
</style>
</head>
<body>
```

I am a *strong* person. I am a *strong* person.

I am a *strong* person. I am a *strong* person.

```
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

</body>
```



# CSS Selectors ( Pseudo-classes)cont.

- **Input (focus)**

- ```
<style>
input:focus {
  background-color: yellow;
}

</style>
</head>
<body>

<form action="/action_page.php" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
</form>

</body>
```



# CSS Selectors ( Pseudo-classes)cont.

- **All CSS Pseudo Classes**

- 

Selector	Example	Example description
<u>:active</u>	a:active	Selects the active link
<u>:checked</u>	input:checked	Selects every checked <input> element
<u>:disabled</u>	input:disabled	Selects every disabled <input> element
<u>:empty</u>	p:empty	Selects every <p> element that has no children
<u>:enabled</u>	input:enabled	Selects every enabled <input> element
<u>:first-child</u>	p:first-child	Selects every <p> elements that is the first child of its parent
<u>:first-of-type</u>	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
<u>:focus</u>	input:focus	Selects the <input> element that has focus
<u>:hover</u>	a:hover	Selects links on mouse over





# CSS Selectors ( Pseudo-classes)cont.

- **All CSS Pseudo Classes**

- 

<u>:in-range</u>	input:in-range	Selects <input> elements with a value within a specified range
<u>:invalid</u>	input:invalid	Selects all <input> elements with an invalid value
<u>:lang(<i>language</i>)</u>	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
<u>:last-child</u>	p:last-child	Selects every <p> elements that is the last child of its parent
<u>:last-of-type</u>	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
<u>:link</u>	a:link	Selects all unvisited links
<u>:not(<i>selector</i>)</u>	:not(p)	Selects every element that is not a <p> element
<u>:nth-child(<i>n</i>)</u>	p:nth-child(2)	Selects every <p> element that is the second child of its parent
<u>:nth-last-child(<i>n</i>)</u>	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child



# CSS Selectors ( Pseudo-classes)cont.

- **All CSS Pseudo Classes**

<u>:nth-of-type(n)</u>	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
<u>:only-of-type</u>	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
<u>:only-child</u>	p:only-child	Selects every <p> element that is the only child of its parent
<u>:optional</u>	input:optional	Selects <input> elements with no "required" attribute
<u>:out-of-range</u>	input:out-of-range	Selects <input> elements with a value outside a specified range
<u>:read-only</u>	input:read-only	Selects <input> elements with a "readonly" attribute specified
<u>:read-write</u>	input:read-write	Selects <input> elements with no "readonly" attribute
<u>:required</u>	input:required	Selects <input> elements with a "required" attribute specified
<u>:root</u>	root	Selects the document's root element
<u>:target</u>	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
<u>:valid</u>	input:valid	Selects all <input> elements with a valid value
<u>:visited</u>	a:visited	Selects all visited links



# CSS Selectors ( Pseudo-elements).

- **CSS Pseudo-elements**
- What are Pseudo-Elements?
- A CSS pseudo-element is used to style specified parts of an element.
- **For example**, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element

## Syntax

The syntax of pseudo-elements:

```
selector::pseudo-element {  
  property: value;  
}
```



# CSS Selectors ( Pseudo-elements).

## The **::first-line** Pseudo-element

- The ::first-line pseudo-element is used to add a special style to the first line of a text.
- The following example formats the first line of the text in all <p> elements:

```
p::first-line {  
    color: #ff0000;  
    font-variant: small-caps;  
}
```



# CSS Selectors ( Pseudo-elements).

- **Note:** The `::first-line` pseudo-element can only be applied to block-level elements.
- **The following properties apply to the `::first-line` pseudo-element:**
  - font properties
  - color properties
  - background properties
  - word-spacing
  - letter-spacing
  - text-decoration
  - vertical-align
  - text-transform
  - line-height
  - clear



# CSS Selectors ( Pseudo-elements).

- **The ::first-letter Pseudo-element**
- The ::first-letter pseudo-element is used to add a special style to the first letter of a text.
- The following example formats the first letter of the text in all <p> elements:

```
p::first-letter {  
    color: #ff0000;  
    font-size: xx-large;  
}
```

---



# CSS Selectors ( Pseudo-elements).

**Note:** The `::first-letter` pseudo-element can only be applied to block-level elements.

The following properties apply to the `::first-letter` pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear




# CSS Selectors ( Pseudo-elements).


- **CSS - The ::before Pseudo-element**

- The **::before** pseudo-element can be used to insert some content before the content of an element.
- The following example inserts an image before the content of each `<h1>` element:

```
<html>
<head>
<style>
h1::after {
  content: url(smiley.gif);
}
</style>
</head>
<body>
```

**This is a heading** 

The ::after pseudo-element inserts content after the content of an element.

**This is a heading** 

```
<h1>This is a heading</h1>
<p>The ::after pseudo-element inserts content after the content of an element.</p>

<h1>This is a heading</h1>

</body>
</html>
```





# CSS Selectors ( Pseudo-elements).

- **CSS - The ::selection Pseudo-element**

- The ::selection pseudo-element matches the portion of an element that is selected by a user.
- *The following CSS properties can be applied to ::selection: color, background, cursor, and outline.*
- The following example makes the selected text red on a yellow background:

```
::selection {  
    color: red;  
    background: yellow;  
}
```



# CSS Selectors ( Pseudo-elements).

## All CSS Pseudo Elements

Selector	Example	Example description
<u>::after</u>	p::after	Insert something after the content of each <p> element
<u>::before</u>	p::before	Insert something before the content of each <p> element
<u>::first-letter</u>	p::first-letter	Selects the first letter of each <p> element
<u>::first-line</u>	p::first-line	Selects the first line of each <p> element
<u>::selection</u>	p::selection	Selects the portion of an element that is selected by a user



# CSS Selectors (Attribute Selectors).

- **Style HTML Elements With Specific Attributes**
- It is possible to style HTML elements that have specific attributes or attribute values.
- **CSS [attribute] Selector**
- The [attribute] selector is used to select elements with a specified attribute.
- The following example selects all <a> elements with a target attribute:

```
a[target] {  
    background-color: yellow;  
}
```



# CSS Selectors (Attribute Selectors).

```
<style>  
• a[target] {  
  background-color: yellow;  
}  
</style>  
</head>  
<body>
```

```
<h2>CSS [attribute] Selector</h2>
```

```
<p>The links with a target attribute gets a yellow background:</p>
```

```
<a href="https://www.w3schools.com">w3schools.com</a>
```

```
<a href="http://www.disney.com" target="_blank">disney.com</a>
```

```
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
```

## CSS [attribute] Selector

The links with a target attribute gets a yellow background:

[w3schools.com](https://www.w3schools.com) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)



# CSS Selectors (Attribute Selectors).

- **CSS [attribute="value"] Selector**
- The **[attribute="value"]** selector is used to select elements with a specified attribute and value.
- The following example selects all <a> elements with a target="\_blank" attribute:

```
a[target="_blank"] {  
    background-color: yellow;  
}
```



# CSS Selectors (Attribute Selectors).

- **CSS [attribute~="value"] Selector**

- The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.
- The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

```
<style>
[title~=flower] {
  border: 5px solid yellow;
}
</style>
</head>
<body>
```

```
<h2>CSS [attribute~="value"] Selector</h2>
```

```
<p>All images with the title attribute containing the word "flower" get a yellow border.</p>
```

```



```

# CSS Selectors (Attribute Selectors).

- **CSS [attribute~="value"] Selector**

- The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.
- The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":





# CSS Selectors (Attribute Selectors).

- **CSS [attribute]="value" Selector**
- The `[attribute]="value"` selector is used to select elements with the specified attribute starting with the specified value.
- The following example selects all elements with a class attribute value that begins with "top":
- **Note:** The value has to be a whole word, either alone, like `class="top"`, or followed by a hyphen( - ), like `class="top-text"`!





# CSS Selectors (Attribute Selectors).

- **CSS [attribute]="value" Selector**
- The `[attribute]="value"` selector is used to select elements with the specified attribute starting with the specified value.
- The following example selects all elements with a class attribute value that begins with "top":
- **Note:** The value has to be a whole word, either alone, like `class="top"`, or followed by a hyphen( - ), like `class="top-text"`!

```
<style>
[class|=top] {
    background: yellow;
}
</style>
</head>
<body>

<h2>CSS [attribute]="value" Selector</h2>

<h1 class="top-header">Welcome</h1>
<p class="top-text">Hello world!</p>
<p class="topcontent">Are you learning CSS?</p>
```



# CSS Selectors (Attribute Selectors).

**Welcome**

Hello world!

Are you learning CSS?



# CSS Selectors (Attribute Selectors).

- **CSS [attribute^="value"] Selector**
- The [attribute^="value"] selector is used to select elements whose attribute value begins with a specified value.
- The following example selects all elements with a class attribute value that begins with "top":
- **Note:** The value does not have to be a whole word!

```
<head>
<style>
[class^="top"] {
    background: yellow;
}
</style>
</head>
<body>

<h2>CSS [attribute^="value"] Selector</h2>

<h1 class="top-header">Welcome</h1>
<p class="top-text">Hello world!</p>
<p class="topcontent">Are you learning CSS?</p>
```



# CSS Selectors (Attribute Selectors).

- **CSS `[attribute^="value"]` Selector**

**Welcome**

Hello world!

Are you learning CSS?



# CSS Selectors (Attribute Selectors).

- **CSS [attribute\$="value"] Selector**

- The [attribute\$="value"] selector is used to select elements whose attribute value ends with a specified value.
- The following example selects all elements with a class attribute value that ends with "test":
- **Note:** The value does not have to be a whole word!

```
<head>
<style>
[class$="test"] {
    background: yellow;
}
</style>
</head>
<body>

<h2>CSS [attribute$="value"] Selector</h2>

<div class="first_test">The first div element.</div>
<div class="second">The second div element.</div>
<div class="my-test">The third div element.</div>
<p class="mytest">This is some text in a paragraph.</p>
```



# CSS Selectors (Attribute Selectors).

- **CSS [attribute\$="value"] Selector**
- The [attribute\$="value"] selector is used to select elements whose attribute value ends with a specified value.
- The following example selects all elements with a class attribute value that ends with "test":
- **Note:** The value does not have to be a whole word!

The first div element.

The second div element.

The third div element.

This is some text in a paragraph.



# CSS Selectors (Attribute Selectors).

- **CSS [attribute\*="value"] Selector**

- The [attribute\*="value"] selector is used to select elements whose attribute value contains a specified value.
- The following example selects all elements with a class attribute value that contains "te":
- **Note:** The value does not have to be a whole word!

```
<style>
[class*="te"] {
    background: yellow;
}
</style>
</head>
<body>

<h2>CSS [attribute*="value"] Selector</h2>

<div class="first_test">The first div element.</div>
<div class="second">The second div element.</div>
<div class="my-test">The third div element.</div>
<p class="mytest">This is some text in a paragraph.</p>

</body>
```



# CSS Selectors (Attribute Selectors).

- **CSS [attribute\*="value"] Selector**
- The [attribute\*="value"] selector is used to select elements whose attribute value contains a specified value.
- The following example selects all elements with a class attribute value that contains "te":
- **Note:** The value does not have to be a whole word!

## CSS [attribute\*="value"] Selector

The first div element.

The second div element.

The third div element.

This is some text in a paragraph.





# CSS Selectors (Attribute Selectors).

## All CSS Attribute Selectors

Selector	Example	Example description
<code>[<u>attribute</u>]</code>	<code>[target]</code>	Selects all elements with a target attribute
<code>[<u>attribute=value</u>]</code>	<code>[target=_blank]</code>	Selects all elements with target="_blank"
<code>[<u>attribute~=value</u>]</code>	<code>[title~=flower]</code>	Selects all elements with a title attribute containing the word "flower"
<code>[<u>attribute =value</u>]</code>	<code>[lang =en]</code>	Selects all elements with a lang attribute value starting with "en"
<code>[<u>attribute^=value</u>]</code>	<code>a[href^="https"]</code>	Selects every <a> element whose href attribute value begins with "https"
<code>[<u>attribute\$=value</u>]</code>	<code>a[href\$=".pdf"]</code>	Selects every <a> element whose href attribute value ends with ".pdf"
<code>[<u>attribute*=value</u>]</code>	<code>a[href*="w3schools"]</code>	Selects every <a> element whose href attribute value contains the substring "w3schools"



# How To Add CSS

- **There are three ways of inserting a style sheet:**
  - 1- External CSS
  - 2- Internal CSS
  - 3- Inline CSS



# How To Add CSS(cont.)

- **External CSS**
- With an external style sheet, you can change the look of an entire website by changing just one file!
- Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

```
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



# How To Add CSS(cont.)

- **Internal CSS**

- An internal style sheet may be used if one single HTML page has a unique style.
- The internal style is defined inside the <style> element, inside the head section.

```
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```



# How To Add CSS(cont.)

- **Inline CSS**
- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

```
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```



# CSS Comments

```
/* This is a single-line comment */  
p {  
    color: red;  
}
```

```
/* This is  
a multi-line  
comment */
```

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



# CSS Color

- **Background Color**

## Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>  
<p style="background-color:Tomato;">Lorem ipsum...</p>
```



# CSS Color (cont.)

- Text Color

## Example

```
<h1 style="color:Tomato;">Hello World</h1>  
<p style="color:DodgerBlue;">Lorem ipsum...</p>  
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```





# CSS Color (cont.)

- **Border Color**

## Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>  
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>  
<h1 style="border:2px solid Violet;">Hello World</h1>
```



# CSS Color (cont.)

- **Border Color**

## Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>  
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>  
<h1 style="border:2px solid Violet;">Hello World</h1>
```



# CSS background-repeat

- By default, the background-image property repeats an image both horizontally and vertically.
- Some images should be repeated only horizontally or vertically, or they will look strange, like this:

```
body {  
    background-image: url("gradient_bg.png");  
    background-repeat: repeat-x;  
}
```

---

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
}
```



# CSS background-position

- The background-position property is used to specify the position of the background image.

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
}
```

---



# CSS background-attachment

The background-attachment property specifies whether the background image should **scroll or be fixed** (will not scroll with the rest of the page):

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
    background-attachment: fixed;  
}
```



# CSS Background Shorthand

- When using the shorthand property the order of the property values is:
- background-color
- background-image
- background-repeat
- background-attachment
- background-position

```
body {  
    background: #ffffff url("img_tree.png") no-repeat right top;  
}
```



# CSS Borders

- **CSS Border Width**

- The border-width property specifies the width of the four borders.
- The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

```
p.one {  
    border-style: solid;  
    border-width: 5px;  
}  
  
p.two {  
    border-style: solid;  
    border-width: medium;  
}  
  
p.three {  
    border-style: dotted;  
    border-width: 2px;  
}  
  
p.four {  
    border-style: dotted;  
    border-width: thick;  
}
```



# CSS Borders

- **CSS Border Color**
- The border-color property is used to set the color of the four borders.

```
p.one {  
    border-style: solid;  
    border-color: red;  
}
```

```
p.two {  
    border-style: solid;  
    border-color: green;  
}
```

```
p.three {  
    border-style: dotted;  
    border-color: blue;  
}
```





# CSS Borders

- **CSS Border - Individual Sides**
- From the examples on the previous pages, you have seen that it is possible to specify a different border for each side.
- In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

```
p {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```



# CSS Margins

## Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- `margin-top`
- `margin-right`
- `margin-bottom`
- `margin-left`

All the margin properties can have the following values:

- `auto` - the browser calculates the margin
- `length` - specifies a margin in px, pt, cm, etc.
- `%` - specifies a margin in % of the width of the containing element
- `inherit` - specifies that the margin should be inherited from the parent element



# CSS Margins

## Margin - Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one property.

The `margin` property is a shorthand property for the following individual margin properties:

- `margin-top`
- `margin-right`
- `margin-bottom`
- `margin-left`

So, here is how it works:

If the `margin` property has four values:

- **`margin: 25px 50px 75px 100px;`**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px



# CSS Margins

If the `margin` property has three values:

- **`margin: 25px 50px 75px;`**
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px

## Example

```
p {  
  margin: 25px 50px 75px;  
}
```



# CSS Margins

If the `margin` property has two values:

- `margin: 25px 50px;`
  - top and bottom margins are 25px
  - right and left margins are 50px

## Example

```
p {  
    margin: 25px 50px;  
}
```

# CSS Margins

If the `margin` property has one value:

- `margin: 25px;`
  - all four margins are 25px

## Example

```
p {  
  margin: 25px;  
}
```



# CSS Padding

This element has a padding of 70px.



# CSS Padding

CSS has properties for specifying the padding for each side of an element:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element
- *inherit* - specifies that the padding should be inherited from the parent element

**Note:** Negative values are not allowed.





# CSS Padding

## Example

```
div {  
  padding-top: 50px;  
  padding-right: 30px;  
  padding-bottom: 50px;  
  padding-left: 80px;  
}
```



# CSS Padding

## Padding - Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one property.

The `padding` property is a shorthand property for the following individual padding properties:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

So, here is how it works:

If the `padding` property has four values:

- **`padding: 25px 50px 75px 100px;`**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px



# CSS Padding

## Example

```
div {  
    padding: 25px 50px 75px 100px;  
}
```



# CSS Padding

If the `padding` property has three values:

- **`padding: 25px 50px 75px;`**
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px

## Example

```
div {  
  padding: 25px 50px 75px;  
}
```



# CSS Padding

If the `padding` property has two values:

- **`padding: 25px 50px;`**
  - top and bottom paddings are 25px
  - right and left paddings are 50px

## Example

```
div {  
    padding: 25px 50px;  
}
```



# CSS Padding

If the `padding` property has one value:

- `padding: 25px;`
  - all four paddings are 25px

## Example

```
div {  
    padding: 25px;  
}
```

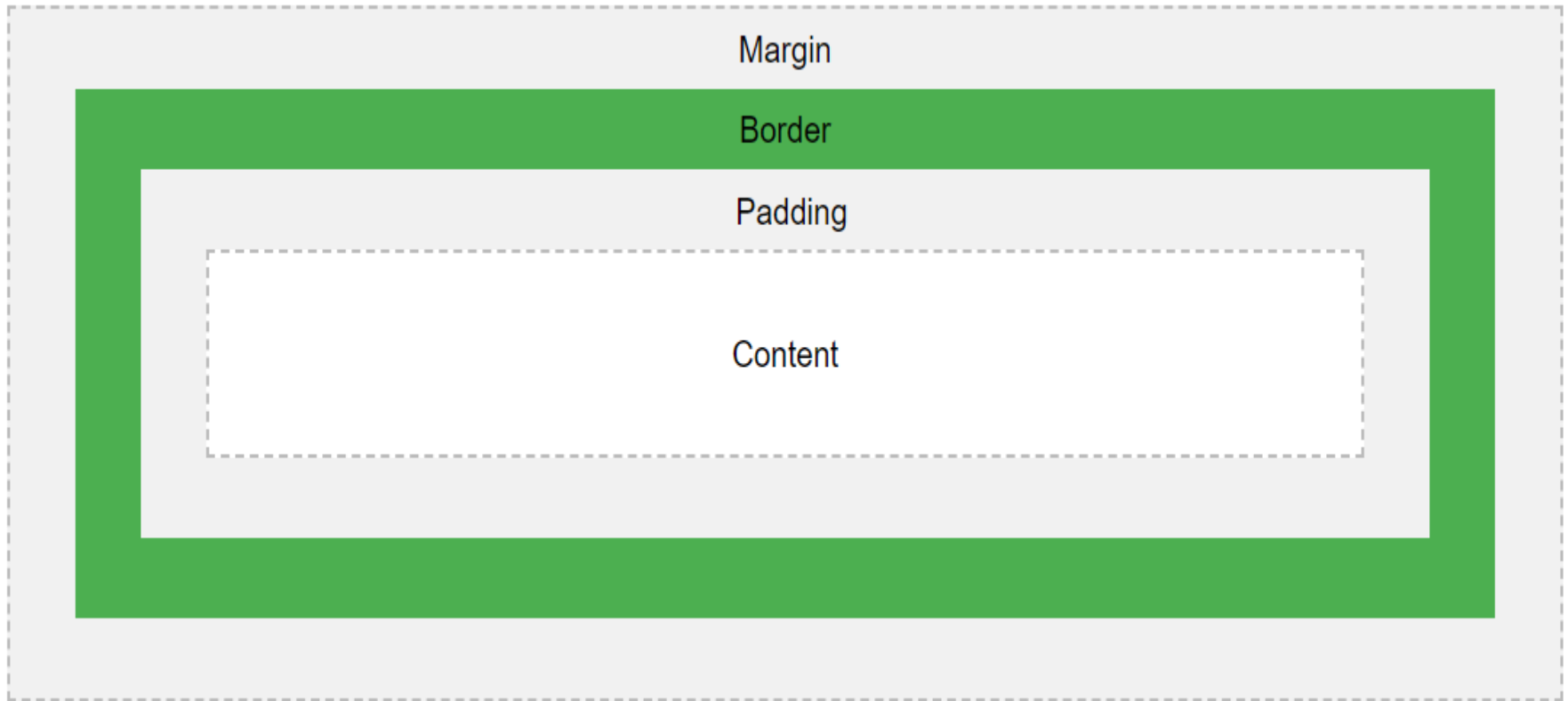


# CSS Box Model

- **The CSS Box Model**
- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



# CSS Box Model







# CSS Box Model

Assume we want to style a `<div>` element to have a total width of 350px:

## Example

```
div {  
  width: 320px;  
  padding: 10px;  
  border: 5px solid gray;  
  margin: 0;  
}
```



# CSS Box Model

Here is the calculation:

```
320px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= 350px
```



# CSS Layout - The display Property

The display property is the most important CSS property for controlling layout.

- **The display Property**
- The display property specifies **if/how** an element is displayed.
- Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.



# CSS Layout - The display Property

## Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The `<div>` element is a block-level element.

Examples of block-level elements:

- `<div>`
- `<h1>` - `<h6>`
- `<p>`
- `<form>`
- `<header>`
- `<footer>`
- `<section>`



# CSS Layout - The display Property

## Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline `<span>` element inside a paragraph.

Examples of inline elements:

- `<span>`
  - `<a>`
  - `<img>`
-



# CSS Layout - The display Property

- **Override The Default Display Value**
- As mentioned, every element has a default display value. However, you can override this.
- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.
- A common example is making inline `<li>` elements for horizontal menus:



# CSS Layout - The display Property

```
<html>
<head>
<style>
li {
  display: inline;
}
</style>
</head>
<body>
```

Display a list of links as a horizontal menu:

[HTML](#) [CSS](#) [JavaScript](#)

```
<p>Display a list of links as a horizontal menu:</p>
```

```
<ul>
  <li><a href="/html/default.asp" target="_blank">HTML</a></li>
  <li><a href="/css/default.asp" target="_blank">CSS</a></li>
  <li><a href="/js/default.asp" target="_blank">JavaScript</a></li>
</ul>

</body>
</html>
```



# CSS Layout - The position Property

- The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

## **There are five different position values:**

- 1- static
- 2- relative
- 3- fixed
- 4- absolute
- 5- sticky





# CSS Layout - The position Property

- The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

## There are five different position values:

- 1- static
- 2- relative
- 3- fixed
- 4- absolute
- 5- sticky

Elements are then positioned using the **top, bottom, left, and right properties**. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.



# CSS Layout - The position Property

- **position: static;**
- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with position: static; is not positioned in any special way; it is always positioned according to the **normal flow of the page:**



# CSS Layout - The position Property

```
<html>
<head>
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
```

```
<h2>position: static;</h2>
```

<p>An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:</p>

```
|
<div class="static">
  This div element has position: static;
</div>
```

```
</body>
</html>
```

**position: static;**

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This div element has position: static;



# CSS Layout - The position Property

- **position: fixed;**
- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.
- Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:



# CSS Layout - The position Property

- **position: relative;**
- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element **will cause** it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.



# CSS Layout - The position Property

- **position: relative;**
- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element **will cause** it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.



# CSS Layout - The position Property

```
<head>
<style>
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
```

```
<h2>position: relative;</h2>
```

```
<p>An element with position: relative; is positioned relative to its normal position:</p>
```

```
<div class="relative">
This div element has position: relative;
</div>

</body>
</html>
```



# CSS Layout - The position Property

## **position: relative;**

An element with `position: relative;` is positioned relative to its normal position:

This div element has `position: relative;`





# CSS Layout - The position Property

- **position: absolute;**
- An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like `fixed`).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- **Note:** A "positioned" element is one whose position is anything except `static`.
- Here is a simple example:

This `<div>` element has `position: relative;`

This `<div>` element has  
`position: absolute;`



# CSS Layout - The position Property

```
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
```

```
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

```
</style>
</head>
<body>
```

```
<h2>position: absolute;</h2>
```

<p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):</p>

```
<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>
```

```
</body>
```

# CSS Layout - The position Property

## • Overlapping Elements

- When elements are positioned, they can overlap other elements.
- The **z-index** property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- An element can have a positive or negative stack order:

```
<html>
<head>
<style>
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
</style>
</head>
<body>
```



**This is a heading**

Because the image has a z-index of -1, it will be placed behind the text.

```
<h1>This is a heading</h1>

<p>Because the image has a z-index of -1, it will be placed behind the text.</p>

</body>
</html>
```



# CSS Layout - Overflow

- The CSS overflow property controls what happens to content that is too big to fit into an area.

The `overflow` property has the following values:

- `visible` - Default. The overflow is not clipped. The content renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content
- `auto` - Similar to `scroll`, but it adds scrollbars only when necessary

**Note:** The `overflow` property only works for block elements with a specified height.



# CSS Layout - Overflow

- **overflow: visible**

```
<head>
<style>
div {
  background-color: #eee;
  width: 200px;
  height: 50px;
  border: 1px dotted black;
  overflow: visible;
}
</style>
</head>
<body>

<h2>CSS Overflow</h2>
<p>By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:</p>

<div>You can use the overflow property when you want to have better control of the layout.
The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```

By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.



# CSS Layout - Overflow

- **overflow: scroll**

```
<html>
<head>
<style>
div {
  background-color: #eee;
  width: 200px;
  height: 100px;
  border: 1px dotted black;
  overflow: scroll;
}
</style>
</head>
<body>
```

```
<h2>CSS Overflow</h2>
```

```
<p>Setting the overflow value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):</p>
```

```
<div>You can use the overflow property when you want to have better control of the layout.
```

## CSS Overflow

Setting the overflow value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what

# CSS Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element.

---

## Transparent Image

The `opacity` property can take a value from 0.0 - 1.0. The lower value, the more transparent:



opacity 0.2



opacity 0.5



opacity 1  
(default)



# CSS Specificity

## What is Specificity?

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out.

Think of specificity as a score/rank that determines which style declarations are ultimately applied to an element.

The universal selector (\*) has low specificity, while ID selectors are highly specific!

**Note:** Specificity is a common reason why your CSS-rules don't apply to some elements, although you think they should.





# CSS Specificity

## Specificity Hierarchy

Every selector has its place in the specificity hierarchy. There are four categories which define the specificity level of a selector:

**Inline styles** - An inline style is attached directly to the element to be styled. Example: `<h1 style="color: #ffffff;">`.

**IDs** - An ID is a unique identifier for the page elements, such as `#navbar`.

**Classes, attributes and pseudo-classes** - This category includes `.classes`, `[attributes]` and pseudo-classes such as `:hover`, `:focus` etc.

**Elements and pseudo-elements** - This category includes element names and pseudo-elements, such as `h1`, `div`, `:before` and `:after`.

---



# CSS Specificity

## How to Calculate Specificity?

Memorize how to calculate specificity!

Start at 0, add 1000 for style attribute, add 100 for each ID, add 10 for each attribute, class or pseudo-class, add 1 for each element name or pseudo-element.

Consider these three code fragments:

### Example

```
A: h1  
B: #content h1  
C: <div id="content"><h1 style="color: #ffffff">Heading</h1></div>
```

The specificity of A is 1 (one element)

The specificity of B is 101 (one ID reference and one element)

The specificity of C is 1000 (inline styling)

Since  $1 < 101 < 1000$ , the third rule (C) has a greater level of specificity, and therefore will be applied.



# CSS Specificity

## Specificity Rules

**Equal specificity: the latest rule counts** - If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled, and therefore will be applied:

### Example

```
h1 {background-color: yellow;}  
h1 {background-color: red;}
```



# CSS Specificity

## Specificity Rules

**Equal specificity: the latest rule counts** - If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled, and therefore will be applied:

### Example

```
h1 {background-color: yellow;}  
h1 {background-color: red;}
```



# Self study

1- CSS Outline

[https://www.w3schools.com/Css/css\\_outline.asp](https://www.w3schools.com/Css/css_outline.asp)

2- CSS Text

[https://www.w3schools.com/Css/css\\_text.asp](https://www.w3schools.com/Css/css_text.asp)

3-CSS Align

[https://www.w3schools.com/Css/css\\_align.asp](https://www.w3schools.com/Css/css_align.asp)