

# **Introduction to CSS**

**Denis Hoti**

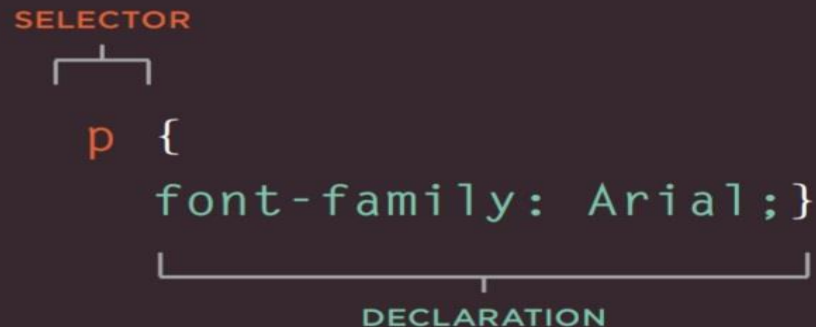
**OJQ THY**

# CSS

- CSS e përdorim që ti bëjmë STYLE web-it tonë.
- CSS mund ta shënojmë në dy mënyra brenda file të HTML dhe të krijojmë një file të CSS.
- File i CSS ruhet me prapashtesen “.css”.
- Për të lidhur një file të CSS në atë të HTML përdorim:
  - `<link rel="stylesheet" type="text/css" href="emri.css">`

# CSS ASSOCIATES STYLE RULES WITH HTML ELEMENTS

CSS works by associating rules with HTML elements. These rules govern how the content of specified elements should be displayed. A CSS rule contains two parts: a **selector** and a **declaration**.



The diagram illustrates the structure of a CSS rule. It shows the word "SELECTOR" in orange above a bracket that points to the "p" in the code "p { font-family: Arial; }". The "p" is also in orange. Below the opening curly brace, the text "font-family: Arial;" is shown in green. A bracket underneath this text points to the word "DECLARATION" in green.

```
SELECTOR
└─┬─
   p {
     font-family: Arial; }
           └──────────┬──────────
                     DECLARATION
```

This rule indicates that all <p> elements should be shown in the Arial typeface.

**Selectors** indicate which element the rule applies to. The same rule can apply to more than one element if you separate the element names with commas.

**Declarations** indicate how the elements referred to in the selector should be styled. Declarations are split into two parts (a property and a value), and are separated by a colon.

# CSS PROPERTIES AFFECT HOW ELEMENTS ARE DISPLAYED

CSS declarations sit inside curly brackets and each is made up of two parts: a **property** and a **value**, separated by a colon. You can specify several properties in one declaration, each separated by a semi-colon.

```
h1, h2, h3 {  
  font-family: Arial;  
  color: yellow;  
}
```



PROPERTY VALUE

This rule indicates that all `<h1>`, `<h2>` and `<h3>` elements should be shown in the Arial typeface, in a yellow color.

**Properties** indicate the aspects of the element you want to change. For example, color, font, width, height and border.

**Values** specify the settings you want to use for the chosen properties. For example, if you want to specify a color property then the value is the color you want the text in these elements to be.

SELECTOR	MEANING	EXAMPLE
UNIVERSAL SELECTOR	Applies to all elements in the document	<code>* {}</code> Targets all elements on the page
TYPE SELECTOR	Matches element names	<code>h1, h2, h3 {}</code> Targets the <code>&lt;h1&gt;</code> , <code>&lt;h2&gt;</code> and <code>&lt;h3&gt;</code> elements
CLASS SELECTOR	Matches an element whose <code>class</code> attribute has a value that matches the one specified after the period (or full stop) symbol	<code>.note {}</code> Targets any element whose <code>class</code> attribute has a value of <code>note</code> <code>p.note {}</code> Targets only <code>&lt;p&gt;</code> elements whose <code>class</code> attribute has a value of <code>note</code>
ID SELECTOR	Matches an element whose <code>id</code> attribute has a value that matches the one specified after the pound or hash symbol	<code>#introduction {}</code> Targets the element whose <code>id</code> attribute has a value of <code>introduction</code>
CHILD SELECTOR	Matches an element that is a direct child of another	<code>li&gt;a {}</code> Targets any <code>&lt;a&gt;</code> elements that are children of an <code>&lt;li&gt;</code> element (but not other <code>&lt;a&gt;</code> elements in the page)
DESCENDANT SELECTOR	Matches an element that is a descendent of another specified element (not just a direct child of that element)	<code>p a {}</code> Targets any <code>&lt;a&gt;</code> elements that sit inside a <code>&lt;p&gt;</code> element, even if there are other elements nested between them
ADJACENT SIBLING SELECTOR	Matches an element that is the next sibling of another	<code>h1+p {}</code> Targets the first <code>&lt;p&gt;</code> element after any <code>&lt;h1&gt;</code> element (but not other <code>&lt;p&gt;</code> elements)
GENERAL SIBLING SELECTOR	Matches an element that is a sibling of another, although it does not have to be the directly preceding element	<code>h1~p {}</code> If you had two <code>&lt;p&gt;</code> elements that are siblings of an <code>&lt;h1&gt;</code> element, this rule would apply to both

# INHERITANCE

## HTML

chapter-10/inheritance.html

```
<div class="page">
  <h1>Potatoes</h1>
  <p>There are dozens of different potato
    varieties.</p>
  <p>They are usually described as early, second
    early and maincrop potatoes.</p>
</div>
```

## CSS

```
body {
  font-family: Arial, Verdana, sans-serif;
  color: #665544;
  padding: 10px;}
.page {
  border: 1px solid #665544;
  background-color: #efefef;
  padding: inherit;}
```

## RESULT

### Potatoes

There are dozens of different potato varieties.

They are usually described as early, second early and maincrop potatoes.

If you specify the `font-family` or `color` properties on the `<body>` element, they will apply to most child elements. This is because the value of the `font-family` property is **inherited** by child elements. It saves you from having to apply these properties to as many elements (and results in simpler style sheets).

You can compare this with the `background-color` or `border` properties; they are **not inherited** by child elements. If these were inherited by all child elements then the page could look quite messy.

You can force a lot of properties to inherit values from their parent elements by using `inherit` for the value of the properties. In this example, the `<div>` element with a class called `page` inherits the padding size from the CSS rule that applies to the `<body>` element.

# FOREGROUND COLOR

## color

The `color` property allows you to specify the color of text inside an element. You can specify any color in CSS in one of three ways:

### RGB VALUES

These express colors in terms of how much red, green and blue are used to make it up. For example: `rgb(100,100,90)`

### HEX CODES

These are six-digit codes that represent the amount of red, green and blue in a color, preceded by a pound or hash `#` sign. For example: `#ee3e80`

### COLOR NAMES

There are 147 predefined color names that are recognized by browsers. For example: `DarkCyan`

We look at these three different ways of specifying colors on the next double-page spread.

CSS3 has also introduced another way to specify colors called HSLA, which you will meet near the end of this chapter on page 255-256.

chapter-11/foreground-color.html

CSS

```
/* color name */
h1 {
  color: DarkCyan;}
/* hex code */
h2 {
  color: #ee3e80;}
/* rgb value */
p {
  color: rgb(100,100,90);}
```

RESULT

## Marine Biology

### The Composition of Seawater

Almost anything can be found in seawater. This includes dissolved materials from Earth's crust as well as materials released from organisms. The most important components of seawater that influence life forms are salinity, temperature, dissolved gases (mostly oxygen and carbon dioxide), nutrients, and pH. These elements vary in their composition as well as in their influence on marine life.

Above each CSS rule in this example you can see how CSS allows you to add comments to your CSS files. Anything between the `/*` symbols and the `*/` symbols will not be interpreted by the browser. They are shown in grey above.

The use of comments can help you to understand a CSS file (and organise it, by splitting a long document into sections). Here, we have used comments to indicate which method is used to specify each of the different types of colors.

# BACKGROUND COLOR

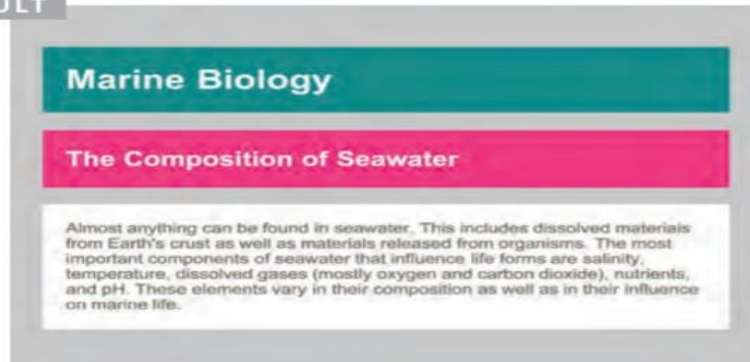
## background-color

### CSS

chapter-11/background-color.html

```
body {  
  background-color: rgb(200,200,200);}   
h1 {  
  background-color: DarkCyan;}   
h2 {  
  background-color: #ee3e80;}   
p {  
  background-color: white;} 
```

### RESULT



CSS treats each HTML element as if it appears in a box, and the `background-color` property sets the color of the background for that box.

You can specify your choice of background color in the same three ways you can specify foreground colors: RGB values, hex codes, and color names (covered on the next page).

If you do not specify a background color, then the background is transparent.

By default, most browser windows have a white background, but browser users can set a background color for their windows, so if you want to be sure that the background is white you can use the `background-color` property on the `<body>` element.

We have also used the `padding` property to separate the text from the edges of the boxes. This makes it easier to read and you will learn more about this property on page 313.

# CSS3: OPACITY

## opacity, rgba

### CSS

chapter-11/opacity.html

```
p.one {  
  background-color: rgb(0,0,0);  
  opacity: 0.5;}  
p.two {  
  background-color: rgb(0,0,0);  
  background-color: rgba(0,0,0,0.5);}
```

### RESULT



### RESULT IN OLDER BROWSER



CSS3 introduces the `opacity` property which allows you to specify the opacity of an element and any of its child elements. The value is a number between 0.0 and 1.0 (so a value of 0.5 is 50% opacity and 0.15 is 15% opacity).

The CSS3 `rgba` property allows you to specify a color, just like you would with an RGB value, but adds a fourth value to indicate opacity. This value is known as an alpha value and is a number between 0.0 and 1.0 (so a value of 0.5 is 50% opacity and 0.15 is 15% opacity). The `rgba` value will only affect the element on which it is applied (not child elements).

Because some browsers will not recognize RGBA colors, you can offer a fallback so that they display a solid color. If there are two rules that apply to the same element, the latter of the two will take priority. To create the fallback, you can specify a color as a hex code, color name or RGB value, followed by the rule that specifies an RGBA value. If the browser understands RGBA colors it will use that rule. If it doesn't, it will use the RGB value.

At the time of writing, the `opacity` and `rgba` properties are only supported by the most recent browsers.

# CSS Border Style

- The border-style property specifies what kind of border to display.
- The following values are allowed:
- **dotted** - Defines a dotted border
- **dashed** - Defines a dashed border
- **solid** - Defines a solid border
- **double** - Defines a double border
- **groove** - Defines a 3D grooved border. The effect depends on the border-color value
- **ridge** - Defines a 3D ridged border. The effect depends on the border-color value
- **inset** - Defines a 3D inset border. The effect depends on the border-color value
- **outset** - Defines a 3D outset border. The effect depends on the border-color value
- **none** - Defines no border
- **hidden** - Defines a hidden border

# Example

Demonstration of the different border styles:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed
solid double;}
```

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

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

- `border-width`
- `border-style` (required)
- `border-color`

## Example

```
p {  
  border: 5px solid red;  
}
```

Result:



Some text

# Left Border

```
p {  
  border-left: 6px solid red;  
  background-color: lightgrey;  
}
```

Result:



Some text

# Bottom Border

```
p {  
  border-bottom: 6px solid red;  
  background-color: lightgrey;  
}
```

Result:

Some text





**PYETJE?**

**Faleminderit për vëmendjen!**