**CSS**

**Introduction :**

CSS, a shorthand for Cascading Style Sheets, is one of the main building blocks of the Web. CSS has improved a lot in the past few years and it’s evolving fast.

**CSS** (an abbreviation of **Cascading Style Sheets**) is the language that we use to style an HTML file, and tell the browser how should it render the elements on the page. People realized the need for a way to style pages, in a way that would work across all browsers. After the initial idea proposed in 1994, CSS got its first release in 1996,

A CSS **rule set** has one part called **selector**, and the other part called **declaration**. The

declaration contains various **rules**, each composed by a **property**, and a **value**.

In this example, p is the selector, and applies one rule which sets the value 20px to the

font-size property:

p {

font-size: 20px;

}

Multiple rules are stacked one after the other:

p {

font-size: 20px;

}

a {

color: blue;

}

A selector can target one or more items:

p, a {

font-size: 20px;

}

and it can target HTML tags, like above, or HTML elements that contain a certain class

attribute with .my-class , or HTML elements that have a specific id attribute with #my-id .

**Semicolons:**

Every CSS rule terminates with a semicolon. Semicolons are **not** optional, but always use Semicolons for consistency and to avoid errors if you add another property and forget to add the semicolon on the previous line.

**Adding CSS to an HTML page**

CSS is attached to an HTML page in different ways.

**1: Using the link tag (External)**

The link tag is the way to include a CSS file. one CSS file is included by all the pages of your site, and changing one line on that file affects the presentation of all the pages in the site.

To use this method, you add a link tag with the href attribute pointing to the CSS file you

want to include. You add it inside the head tag of the site (not inside the body tag):

<link rel="stylesheet" type="text/css" href="myfile.css">

The rel and type attributes are required too, as they tell the browser which kind of file we

are linking to.

**2: using the style tag (Internal)**

Instead of using the link tag to point to separate stylesheet containing our CSS, we can add

the CSS directly inside a style tag. This is the syntax:

<style>

...our CSS...

</style>

Using this method we can avoid creating a separate CSS file.

**3: inline styles**

Inline styles are the third way to add CSS to a page. We can add a style attribute to any

HTML tag, and add CSS into it.

<div style="">...</div>

Example:

<div style="background-color: yellow">...</div>

**Selectors:**

HTML elements have 2 attributes which are very commonly used within CSS to associate styling to a specific element on the page: class and id .

There is one big difference between those two: inside an HTML document you can repeat the same class value across multiple elements, but you can only use an id once. Using classes you can select an element with 2 or more specific class names, something not possible using ids.

Classes are identified using the . symbol, while ids using the # symbol.

**Specificity**. The more specific rule will win. If two or more rules have the same specificity, the one that appears last wins.

Any inline style will have precedence over any rule defined in an external CSS file, or inside the style tag in the page header.

Example:

<p style="color: red">Test</p>

Even if any other rule in the CSS defines the color, this inline style rule is going to be applied.

Except for one case - if !important is used.

**Importance**

Specificity does not matter if a rule ends with !important :

p {

font-size: 20px!important;

}

That rule will take precedence over any rule with more specificity. Adding !important in a CSS rule is going to make that rule be more important than any other rule, according to the specificity rules.

**Inheritance**

When you set some properties on a selector in CSS, they are inherited by all the children of

that selector. You don't need to apply the font-family to every single tag of your page. You set the body tag font, and every children inherits it, along with other properties. The background-color property, on the other hand, makes little sense to be inherited.

**Colors**

We have a white background, black color, and blue links. That's it. Luckily CSS gives us the ability to add colors to our designs. We have these properties:

1.color

2.background-color

3.border-color

All of them accept a **color value**, which can be in different forms.

**RGB and RGBa**

You can use the rgb() function to calculate a color from its RGB notation, which sets the

color based on its red-green-blue parts. From 0 to 255:

p {

color: rgb(255, 255, 255);

background-color: rgb(0, 0, 0);

}

rgba() lets you add the alpha channel to enter a opacity part. That can be a number from

0 to 1:

**Hexadecimal notation**

Another option is to express the RGB parts of the colors in the hexadecimal notation, which is

composed by 3 blocks.

Black, which is rgb(0,0,0) is expressed as #000000 or #000

**Units**

One of the things you'll use every day in CSS are units. They are used to set lengths, paddings, margins, align elements and so on.Things like px , em , rem , or percentages.

**Pixels**

The most widely used measurement unit.

**Percentages**

Another very useful measure, percentages let you specify values in percentages of that parent

element's corresponding property.

**Backgrounds**

The background of an element can be changed using several CSS properties:

background-color,background-image,background-clip,background-position,background-origin,background-repeat,background-attachment,background-size.

**Comments**

CSS gives you the ability to write comments in a CSS file, or in the style tag in the page header. The format is the /\* this is a comment \*/ C-style (or JavaScript-style, if you prefer) comments.

CSS does not have inline comments, like // in C or JavaScript.

**Fonts**

Today’s you can load any kind of font on your pages.CSS has gained many nice capabilities over the years in regards to fonts.

**Typography**

We already talked about fonts, but there's more to styling text. Some properties in below.

text-transform,text-decoration,text-align,vertical-align,line-height,text-indent,text-align-last

**Box Model**

Every CSS element is essentially a box. Every element is a generic box. From the inside to the outside, we have: 1) the content area.2) padding.3) border.4) margin.

**Positioning**

Positioning is what makes us determine where elements appear on the screen, and how they appear. You can move elements around, and position them exactly where you want.

We have one main CSS property: position.

It can have 5 types:

1)static : This is the default value of an element.

2)relative: Top,Right,Left,Bottom.

3)absolute:

4)fixed

5)sticky

**Tables**

Tables in the past were greatly overused in CSS, as they were one of the only ways we could

create a fancy page layout.

This is a basic table:

<table>

<thead>

<tr>

<th scope="col">Name</th>

<th scope="col">Age</th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">Flavio</th>

<td>36</td>

</tr>

<tr>

<th scope="row">Roger</th>

<td>7</td>

</tr>

<tr>

<th scope="row">Syd</th>

<td>6</td>

</tr>

</tbody>

</table>

By default it's not very attractive. The browser provides some standard styles, and that's it:

**Name Age**

Flavio 36

Roger 7

Syd 6

We can use CSS to style all the elements of the table.

**Centering**

Centering things in CSS is a task that is very different if you need to center horizontally or

vertically.

**Logic operators**

1)We can combine rules using and

2)We can perform an "or" type of logic operation using commas, which combines multiple media

queries:

3)We can use not to negate a media query.

**Media queries**

You need to wrap them in a @media () {} structure.

Example:

@media screen and (max-width: 200px) {

/\* enter some CSS \*/

}

**Filters**

Filters allow us to perform operations on elements.

Things you normally do with Photoshop or other photo editing software, like changing the

opacity or the brightness, and more.You use the filter property. This property can be used on *any* element:

.a {

filter: <something>;

}

**Animations**

CSS Animations are a great way to create visual animations,

An animation is applied to an element using the animation property.

.b {

animation: <something>;

}