Programming Javascript for Web and Mobile

Pepe García

2020-04-20

Programming Javascript for Web and Mobile

CSS rules

CSS rules are formed by keys and values, separated by a colon, and finished by a semicolon:

```
key: value;
```

CSS rules. Styling text

How can we give style to text

```
color: #000;
font-family: Georgia, Times;
font-size: 22px;
font-style: italic;
font-weight: bold;
text-decoration: underline;
text-align: justify;
```

CSS rules. Styling text

How can we give style to text

The box model

All tags in HTML will behave as boxes. These boxes have some common properties that we need to be aware of.

The box model

The box model

- **width** and **height** apply to the content of the box
- The **padding** is the distance from the content to the border
- The **border** sits between the margin and padding
- The **margin** is the distance from the border to the other elements

The box model

```
background: #000;
width: 100px;
padding: 20px;
margin: 20px;
border: 5px solid orange;
```

some CSS rules that we can apply to boxes:

CSS rules

https://github.com/mcsbt-web-programming-2020/session-1

Let's fix exercise 2!

CSS selectors

Selectors define the elements to which a CSS rule apply.

For example, if we want to apply a CSS rule to all images, we can use the **img** selector.

element selectors

In element selectors we use the element name to select it:

```
h1 {
  color: #EEE;
}

p {
  text-align: justify;
}
```

class selectors

We can give a class to HTML elements so we can refer to it from CSS later on:

```
<p class="red-text">this text will be red</p>
```

```
.red-text {
  color: red;
}
```

HTML:

CSS:

id selectors

We use the id attribute to give a unique identifier to a tag in the document. We can refer to it from the CSS later:

```
<h1 id="title">title</h1>
```

```
#title {
  font-size: 50px;
  color: #000;
}
```

HTML:

CSS:

CSS dinner!

https://flukeout.github.io/

Practice

Let's do exercise3 from

https://github.com/mcsbt-web-programming-2020/session-1

Display

the CSS display property describes how an element's box behaves

All HTML values have a default display property.

display:block

Block elements are those that start on a new line and fill their whole container, from left to right.

Some block elements are <**div**>, <**li**>

display:block

display.html

display:inline

Inline elements do not break the flow of text in a paragraph. Some examples of inline-by-default elements are <**a**>, and <**span**>

display:inline

display.html

display:none

display:none makes the element not to be displayed in the screen.

display:none

Let's clean up https://www.huffingtonpost.es

Horizontal centering

One of the first things we'll want to do when creating the layout of a webpage is centering the content horizontally.

This can be done using **margin** and **width**:

```
body {
  width: 800px;
  margin: auto;
}
```

Horizontal centering

but, what happens when we resize the window?

Horizontal centering

max-width allows us to express the maximum width we want for a box, so it resizes in case of smaller screens.

```
body {
  max-width: 800px;
  margin: auto;
}
```

Position

we use the **position** css property to make more complex layouts.

There are a number of different values for it:

position:relative

The relative position allows us to place the element related to where it would be placed by default.

We can move them with: **top, bottom, right,** and **left.**

position:fixed

with the **position:fixed** attribute we can make **** elements that stay in the same position. These are elements whose position doesn't change even when scrolling.

position:absolute

with **position:absolute** we can place elements like with **fixed** but relative to their *nearest positioned parent*

position

position.html