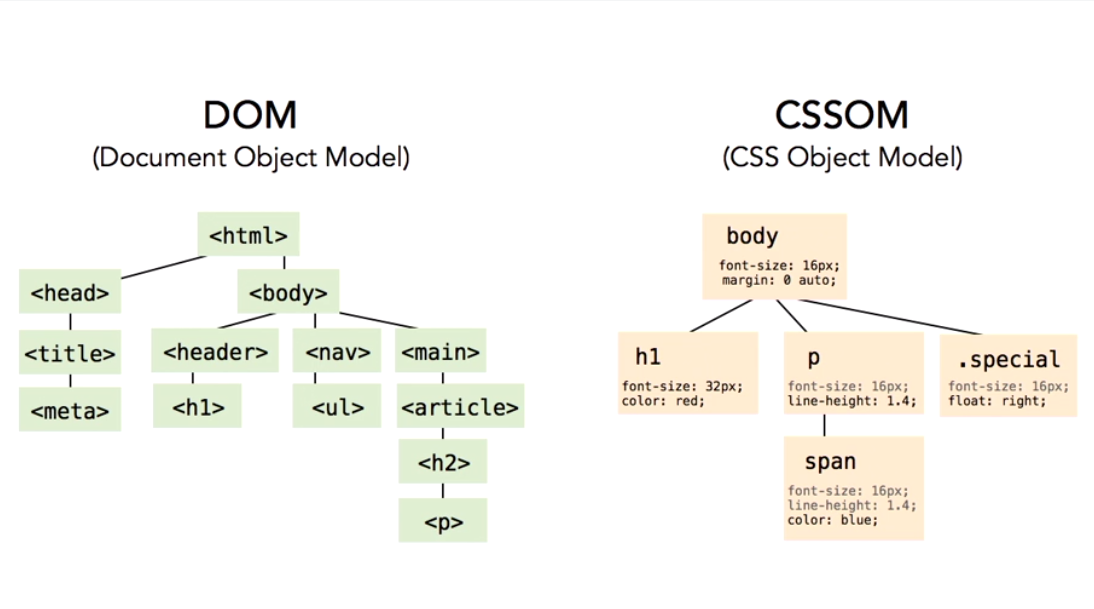
**CSS**

When a browser first encounters a webpage,it builds two things: the Document Object Model or DOM and the CSS Object Model. The DOM is essentially a tree-like structure that describes the relationship of a page's elements to one another.

The CSS Object Model breaks down any stylesheet content in a similar fashion with rules being described in a tree-like representation depending upon how the styles were authored. The CSS Object Model is then combined with the DOM and any user-specific preferences and then rendered in the browser. To construct the CSS Object Model, a browser must parse through all of the selectors in a stylesheet, and then group the styles in a tree-like fashion that is similar to the DOM. Because CSS is render blocking, this step is critically important.



**Universal selector**

Matches all element types in the current document. While sometimes referred to as the ”wildcard” selector, this term is not entirely accurate. The universal selector matches all element types, regardless of structure. While useful, care must be taken when using the universal selector in that it can unintentionally overwrite inheritance in styles, and it can negatively impact browser performance if overused.

Select all elements \*{}

It will not effect if the styles are defined earlier.

Select all para div \* em{}

**Attribute Selectors:**

a[href^="http://"] {}

a[href$=".pdf"] {}

input[type=”text”]:focus{}

[class\*="tip"] {}

**pseudo-class selectors :**

Again it's a pseudo-class selector so no spaces between the colon and the element that it's targeting.

a:link{}, :hover, :visited

**Link Targeting:**

anchor tags - lovehate lvht – link, visited, hover, active

**Form Elements:**

**fieldset**

**Focus:**

Input: focus{}

Essentially this selector is saying hey, any label element which is directly preceded by an input type of check box, that's been checked.

input[type="checkbox"]:checked + label{

color: red;

}

Fragment Identifier

a link will be used to target a specific region of the page using what's known as a fragment identifier. Using the target pseudo-class selector we can give these regions unique styling when the URL contains the targeted fragment IT.

**child selectors**

**literal child selectors**, and there are three of those. There is first-child, last-child, and only-child. Now the reason I call them literal child selectors is because they only match if they are literally the first, last, or only elements within a parent, so, the example that I have here in the first paragraph that you see where I'm matching a p:first-child is only gonna return a match if that paragraph is the first-child within the parent, meaning it comes before any other content.

So it's not looking for the first paragraph, it's looking to see if that paragraph is the first-child. Now, for syntax, they're all very similar. You can see I'm gonna scroll down here a little bit into the syntax section. Here's first-child, last-child, and only-child. Now, you can see they're pretty normal pseudo-class selectors. If you want to make 'em element specific, you just place the simple selector prior to the colon, and that's about it. Now to experiment with these, I've created a little section down here at the bottom of the page for our example, where I have different paragraphs nested in different locations within the syntax.

p:first-child is only gonna return a match if that paragraph is the first-child within the parent, meaning it comes before any other content

the first-of-type, last-of-type, and only-of-type. So those are incredibly similar to the last, first, and only child selectors. But in this case instead of matching the literal first, last, or only child, they matched based upon the type of element that you're targeting.

Empty elements:

the Empty Pseudo-Class Selector can actually target and style them