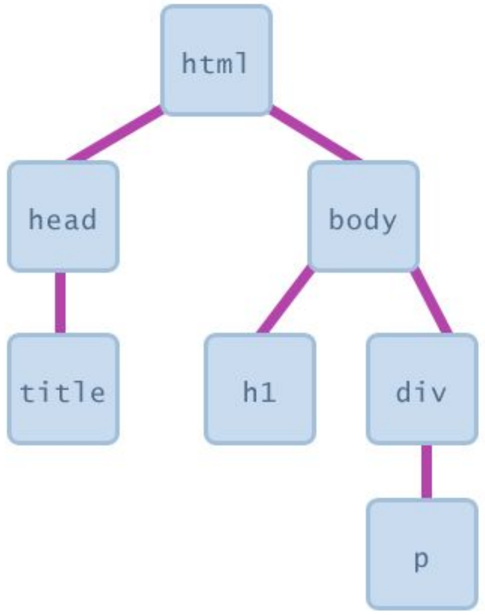
Javascript
HTML DOM

The HTML DOM

Every element on an HTML page is accessible in JavaScript through the **DOM: Document Object Model**

- The DOM is the tree of nodes corresponding to HTML elements on a page.
- Can modify, add and remove nodes on the DOM, which will modify, add, or remove the corresponding element on the page.



The HTML DOM

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

JavaScript can change all the HTML elements in the page
JavaScript can change all the HTML attributes in the page
JavaScript can change all the CSS styles in the page
JavaScript can remove existing HTML elements and attributes
JavaScript can add new HTML elements and attributes
JavaScript can react to all existing HTML events in the page
JavaScript can create new HTML events in the page

The HTML DOM Document Object

The document object represents your web page.

If you want to access any element in an HTML page, you always start with accessing the document object.

Then you can do a lot of things with the document object:

Action	**Example**
Finding HTML Elements	document.querySelector(*CSS selector*);
Adding and Deleting Elements	document.createElement(*element*);
Changing HTML Elements	element.innerHTML = *new html content;*
Adding Events Handlers	element.addEventListener('event', *handler*);

Finding HTML Elements

If you want to find **the first** HTML elements that matches a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelector() method.

For example this javascript statement will return the **first paragraph element of class main**:

```
document.querySelector("p.main");
```

```
<body>
   <p>my first paragraph</p>
   <p class="main">my first main paragraph</p>
   <p class="main">my second main paragraph</p>
   <a href="http://www.google.com">google</a>
</body>
```

Finding HTML Elements

If you want to find **all** HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

For example this javascript statement will return **all paragraph elements of class main**:

```
document.querySelectorAll("p.main");
```

```
<body>
   <p>my first paragraph</p>
   <p class="main">my first main paragraph</p>
   <p class="main">my second main paragraph</p>
   <a href="http://www.google.com">google</a>
</body>
```

Finding HTML Elements

querySelectorAll() method will return a list of all HTML elements that match the specified CSS query.

```
const pars = document.querySelectorAll("p.main");
```

```
<body>
   <p>my first paragraph</p>
   <p class="main">my first main paragraph</p>
   <p class="main">my second main paragraph</p>
   <a href="http://www.google.com">google</a>
</body>
```

pars[0]

pars[1]

Changing HTML Elements

The HTML DOM allows JavaScript to change the content of HTML elements.

The easiest way to modify the content of an HTML element is by using the **innerHTML** property.

To change the content of an HTML element, use this syntax:

This is the element you want to change the html inside of it

element.innerHTML = *new HTML*

this is the new html code or text you want to put inside the element

Changing HTML Elements

For example this javascript code changes the text inside the h1 element:

```
let header = document.querySelector("h1");
header.innerHTML = "My new heading";
```

<h1>My old heading</h1> → <h1>My new heading</h1>

Changing HTML Elements

You can also change the value of an HTML attribute.

element.attribute = *new value*

This is the element you want to change an attribute of

This is the attribute you want to change

this is the new value you want to assign to the specified attribute of the given element

Changing HTML Elements

For example this javascript code changes the href attribute of an <a> element with id myLink:

```
let myLink = document.querySelector("#myLink");
myLink.href = "http://www.newwebsite.com";
```

<a href="http://www.oldwebsite.com" id="myLink">A link to my website</a>

<a href="http://www.newwebsite.com" id="myLink">A link to my website</a>

Changing CSS properties

To change the style of an HTML element, use this syntax:

this is the style property you want to change like font-size

element.style.*property* = *new style*

This is the element you want to change the style of it

this is the new value for the style property you want to change

[complete list of Object style properties]

Changing CSS properties

For example this javascript code changes the font size of the second <p> element with class par to twice it's default value:

```
let pars = document.querySelectorAll("p.par");
pars[1].style.fontSize = "2em";
```

```
<a href="http://www.oldwebsite.com" id="myLink">A link to my website</a>
```

$$\Downarrow$$

```
<a href="http://www.newwebsite.com" id="myLink">A link to my website</a>
```

Changing CSS properties

Did you notice?

```
p.par {
  font-size: 2em;
}
```

→

```
pars[1].style.fontSize = "2em";
```

As a general rule of thumb, in order to get the style property name in javascript, you should change the CSS property name to camelCase!

CSS property	**Javascript property**
color	color
background-color	backgroundColor
margin-top	marginTop

Adding HTML Elements

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
document.createElement(element);
```

This is the name of the element you want to create e.g. "p"

```
document.createTextNode(some text);
```

This creates the text that can go inside an html element. e.g. some text inside a <p> or <h1>

```
parentElement.appendChild(childElement);
```

This is the element you want to append the child element to

This is the child element you want to nest inside the parent element

Adding HTML Elements

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
let para = document.createElement("p");
let node = document.createTextNode("This is new.");
para.appendChild(node);

let element = document.querySelector("#div1");
element.appendChild(para);
```

Adding HTML Elements

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
   <p>This is new.</p>
</div>
```

```
let para = document.createElement("p");
let node = document.createTextNode("This is new.");
para.appendChild(node);

let element = document.querySelector("#div1");
element.appendChild(para);
```

Adding HTML Elements

The appendChild() method in the previous example, appended the new element as the last child of the parent.

If you don't want that you can use the **insertBefore()** method:

```
parentElement.insertBefore(newElement, existingElement)
```

This is the parent element you want to insert the new element inside it

This is the new element you want to insert inside the parent element and before the existing element

This is the existing element inside parent element, for which you want to insert the new element before it

Adding HTML Elements

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
let para = document.createElement("p");
let node = document.createTextNode("This is new.");
para.appendChild(node);

let element = document.querySelector("#div1");
let child = document.querySelector("#p1");
element.insertBefore(para, child);
```

Adding HTML Elements

```html
<div id="div1">
   <p>This is new.</p>
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```javascript
let para = document.createElement("p");
let node = document.createTextNode("This is new.");
para.appendChild(node);

let element = document.querySelector("#div1");
let child = document.querySelector("#p1");
element.insertBefore(para, child);
```

Removing Existing HTML Elements

To remove an HTML element, you must know the parent of the element

Then you can use this syntax to remove the element you want:

parentElement.removeChild(*childElement*)

This is the parent element you want to remove one of its children elements

This is the child element you want to remove

Removing Existing HTML Elements

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
let parent = document.querySelector("#div1");
let child = document.querySelector("#p1");
parent.removeChild(child);
```

Removing Existing HTML Elements

```
<div id="div1">
   ~~<p id="p1">This is a paragraph.</p>~~
   <p id="p2">This is another paragraph.</p>
</div>
```

```
let parent = document.querySelector("#div1");
let child = document.querySelector("#p1");
parent.removeChild(child);
```

Replacing HTML Elements

To replace an element, use the replaceChild() method:

parentElement.replaceChild(newElement, *oldElement*)

This is the parent element you want to replace one of its children elements

This is the new child element you want to add to the parent element by replacing the old one

This is the child element you want to replace

Replacing HTML Elements

```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
let newPar = document.createElement("p");
let node = document.createTextNode("This is new.");
newPar.appendChild(node);

let parent = document.querySelector("#div1");
let oldPar = document.querySelector("#p1");
parent.replaceChild(newPar, oldPar);
```

Replacing HTML Elements

```
<div id="div1">
   <p>This is new.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
let newPar = document.createElement("p");
let node = document.createTextNode("This is new.");
newPar.appendChild(node);

let parent = document.querySelector("#div1");
let oldPar = document.querySelector("#p1");
parent.replaceChild(newPar, oldPar);
```

Javascript DOM Events

Javascript can react to HTML DOM events

To achieve this we have to add an event listener that fires when a user causes any event e.g. clicks a button

element.addEventListener(*event*, *eventHandler*)

This is the element you want to capture the events on

This is the event you want to capture e.g. 'click' or 'mouseover'

This is the name of the function you want to call when the event is fired

Javascript DOM Events

You can add many eventHandlers for the same or different events to the same element:

```
const element = document.querySelector("button");

element.addEventListener('click', function1);
element.addEventListener('click', function2);
element.addEventListener('keyup', function3);
```

Javascript DOM Events

You can also remove event handlers that have been attached with the addEventListener() method:

element.removeEventListener(*event*, *eventHandler*)

This is the element you want to remove the eventListener from

This is the event name you want to remove the eventListener for

This is the name of the function you have used as the eventHandler for the eventListener you want to remove

Javascript DOM Events

```
const element = document.querySelector("button");

element.addEventListener('click', function1);
element.addEventListener('click', function2);
element.addEventListener('keyup', function3);

//This removes the second eventListener for 'click'
element.removeEventListener('click', function2);
```