Client-side Javascript

Client-side Javascript consists of three parts:

- The core language (loop, arrays, etc)
- The Browser Object Model (BOM)
- The Document Object Model (DOM)

The Browser Object Model

- An **application programming interface** (API) for browsers:
    - **window** object
    - **navigator** object
        - which browser this is
        - whether cookies are enabled
        - these things may not work in every browser
    - **location** object
        - which URL they're currently looking at
        - can change this without asking the user
    - **screen** object
        - can find out screen resolution
        - this is also not widely supported
    - **history** object
        - you can make a user go back and forward in their history

The Document Object Model

- An API for HTML documents (and XML documents):
    - Allows you to use JavaScript to change the content and appearance of a web page without reloading it.
    - Treats the page as a heirarchy of different types of **node**:
        - Allows you to use JS to remove, replace, insert, or change the nodes.

Nodes

Nodes come in three types:

- **Document** nodes (contains the html)

- **Element** nodes (start tag, content, end tag)
- **Text nodes** (e.g. stored in
  tag)

Finding element nodes in javascript

```
element = document.querySelector('canvas');
```

This code goes to the document and finds all canvases (using CSS selector), and returns the first one, or null if no element matches the CSS selector.

```
elements = document.querySelectorAll('CSS_selector');
```

This returns an array-like collection of all elements which are specified by the CSS selector. An empty array if nothing matches.

Creating new element nodes and new text nodes

To create a new **p** element node:

```
new_p_element = document.createElement('p');
```

To create a new text node:

```
new_text_node = document.createTextNode('Lorem ipsum');
```

Once created, you have to put these things into the tree.

Changing the tree's nodes and values

Changing the nodes in the tree:

```
some_node.appendChild(new_node);
```

This add **new_node** to the end of the **some_node**'s children.
Others:

- insertBefore
- replaceChild

- removeChild

Changing the text in a text node:

```
some_text_node.nodeValue = 'Some new text';
```

Others:

- appendData
- insertData
- deleteData

A more concise approach

You can create a new element and make it a child in one by using:

```
new_p_element.innerHTML;
```

innerHTML is much more powerful than this example suggests.

Properties of element nodes

JavaScript can retrieve and change attribute values by accessing and setting object properties. It's very easy:

```
some_node.id = 'new_id'
```

Note you would find the node using querySelector.

Changing the value of the class attribute

This is an exception. You cannot use `some_node.class` to change the class in JS, because `class` is already a reserved word in javascript.
Instead you have to write:

```
some_node.className = 'new_class'
```

Note

There are many examples in Derek Bridge's notes.

Slight aside

`<div>` is now old-fashioned, stuff like `<section>`, `<article>`, `<nav>` are replacing it.