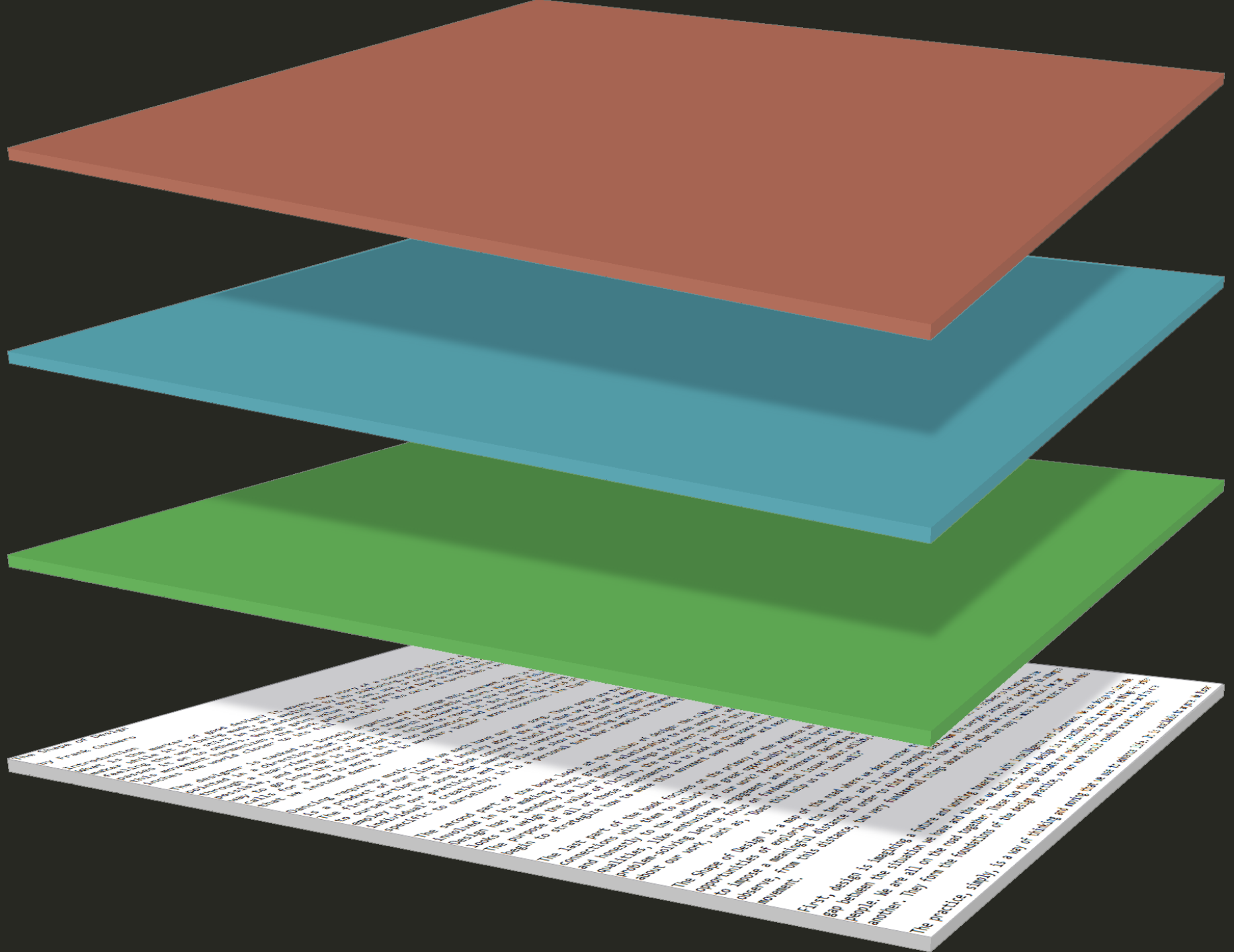
The behavioural layer (JavaScript 2)

Content Management



Behaviour

JavaScript

Presentation

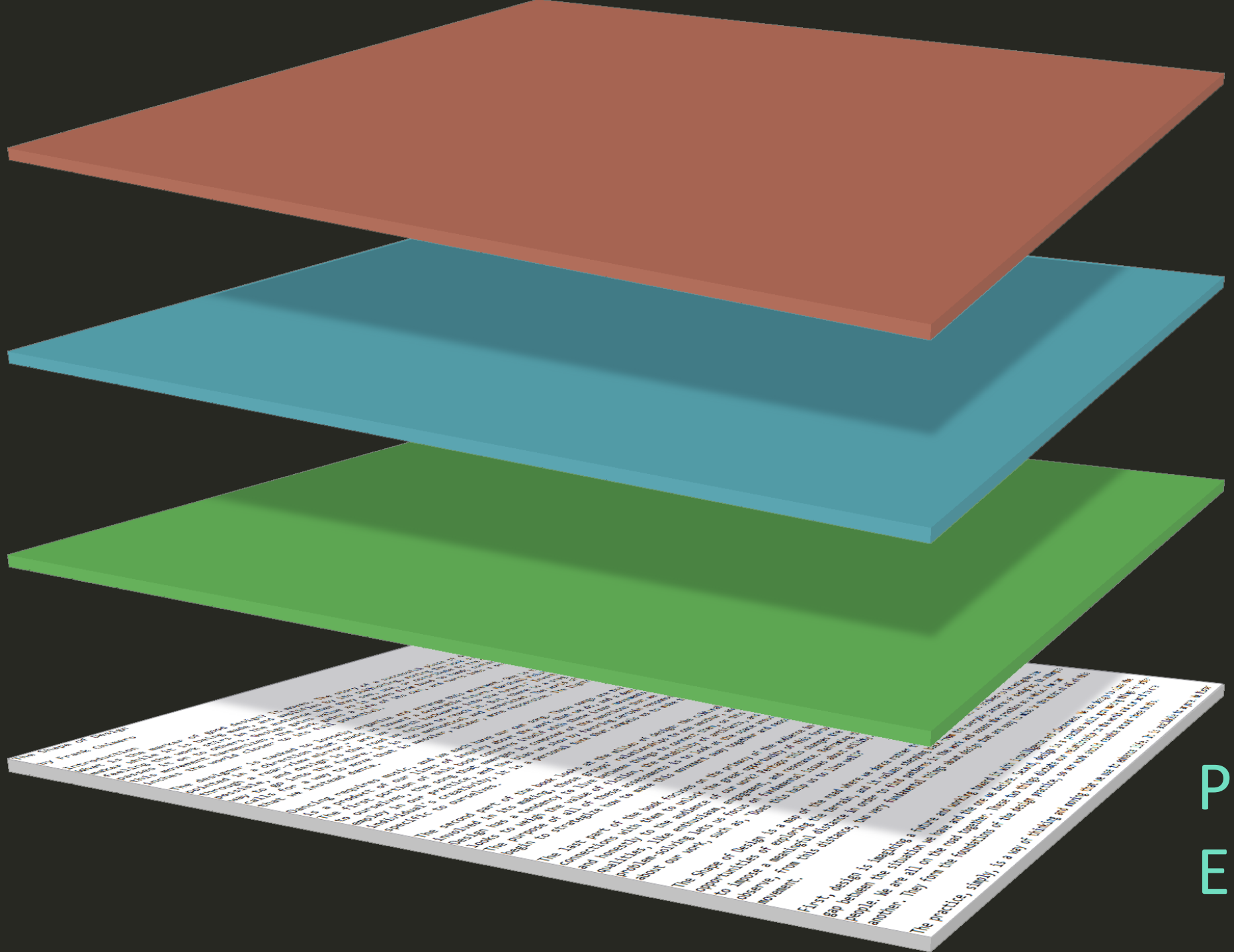
CSS

Structure

HTML

Content

Text, images etc.



Richness of experience

Progressive Enhancement

All three layers of the web standards model combined…

structure | presentation | behaviour | webpage in browser

The browser loads the HTML file and then any linked resources such as the CSS file, the JavaScript file and any media files (e.g. images). A single webpage is built from many separate files.

Controlling the flow of a script

All the scripts we've written so far will run automatically when the document is loaded in the browser. The trick to building successful user interactions is in being able to tell the browser when a certain part of a script should be run. To do this, we create a bit of script called a *function*.

```
function functionName(){
   do something when the function is called
}
```

Building user interactions

```
<script>
1. Create the markup;
2. Add event listeners;
3. Build the functions to run;
</script>
```

Building user interactions is, essentially, a three-step process. The first step is to create the markup we need for the user controls. The second step is to add an event listener on the element used as a control. The third is to add a JavaScript function, that will run when the control is triggered.

Creating markup with JavaScript

```
var ul = document.createElement('ul');
var parent = document.getElementById(parent);
parent.appendChild(ul);
```

Adding an element to a document is a three-step process, represented by the three statements above:

1. Create the element you want and assign to a variable.
2. Select the parent element.
3. Add the new element to its parent.

createElement

1. This method allows us to create any HTML element. However, it isn't immediately added to the document (DOM tree), instead it is held in a variable, ready to be inserted where we want.

Select parent

2. You may use any convenient method to select the parent element. Here we're using getElementById but you could use querySelector, for example. The parent element is assigned to a variable.

appendChild

3. This method will add the specified new element as the *last* child of the selected parent element. If you need to add a new element as the *first* child, you may use the prepend method.

Enhancing markup with JavaScript

```
var element = document.querySelector('li:nth-child(1)');
element.id = 'name';
element.innerHTML = 'Some text';
```

Once an element has been added to the document, it can be selected and manipulated:

1. We can add an ID.
2. We can add text between opening and closing tags.

id

1. This method allows us to add an ID to the specified element. In the example above, we're adding the id "name" to the first list item:

```
<li id="name"></li>
```

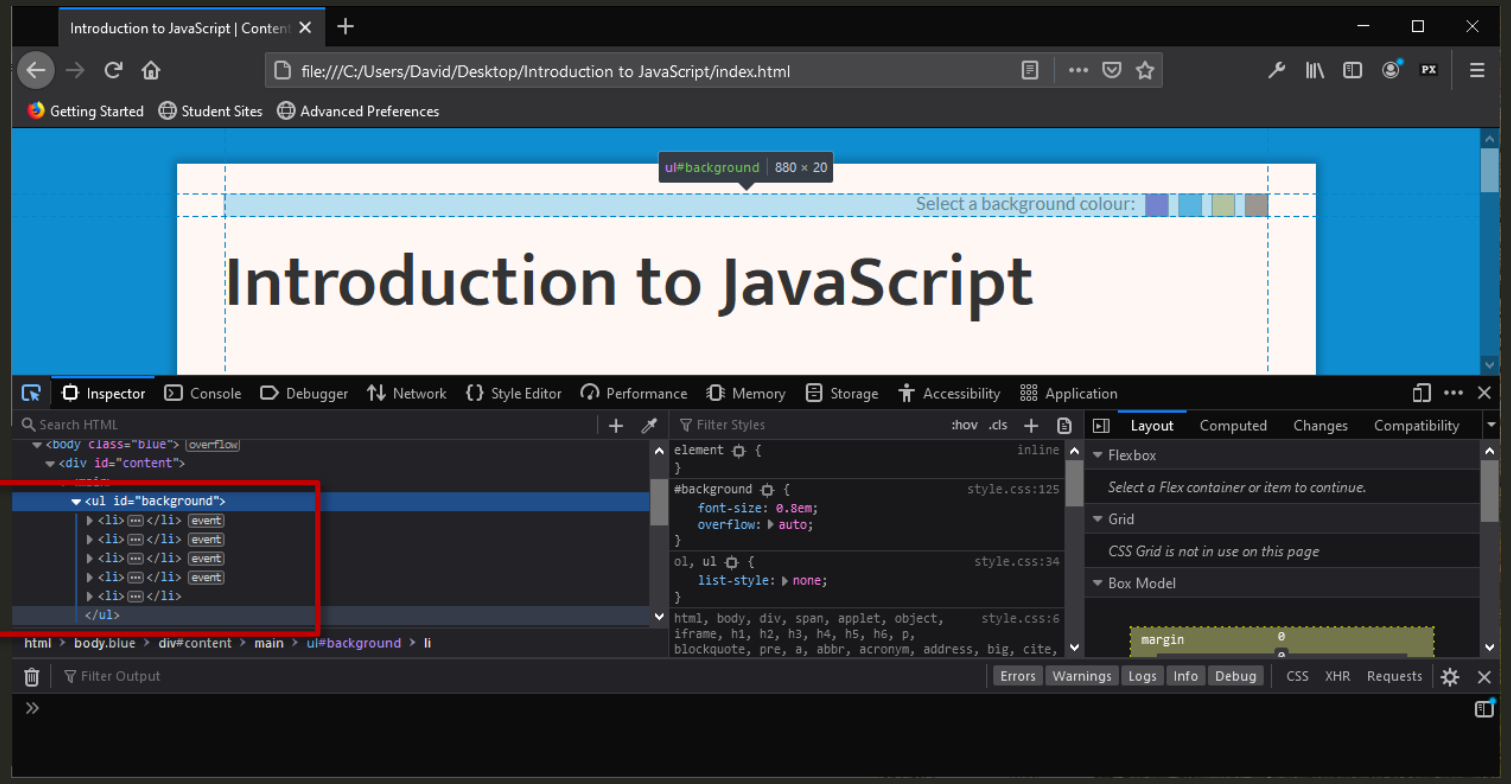
innerHTML

2. In the third statement, we are adding some text to the same element, and the result is:

```
<li id="name">Some text</li>
```

Bear in mind that this generated code is not visible in the document source, it can only be viewed in the inspector…

Viewing generated markup



The inspector panel in Firefox Developer can be used to check any markup generated using JavaScript – you won't see it by viewing source.

Adding an event listener

```
var element = document.querySelector('li:nth-child(1)');
element.addEventListener('click', function);
```

Adding an event listener is a simple two-step process:

1. Select the element to which you want to add an event listener and assign it to a variable.
2. Use the addEventListener method and specify the type of event (in this case a click) and the name of the function to run when the event takes place.

addEventListener

This method is used to add an event listener (sometimes referred to as an event handler) to an element. In the wild, you may see alternative ways of doing this. For example, the following achieves a similar outcome:

```
element.onclick = function
```

Building a function

```
function addClass(){
      element.className = 'highlight';
}
```

function

A function is a self-contained script that is given a name. The function will run when its name is called. In the example above, class="highlight" is added to the element stored in the element variable when the function is run.

className

This method is used to add a class to an element. It works in the same way as the id method we used on an earlier slide.

Although this function works perfectly well, we will usually want to check if an element already has a class before adding another, in case we want to replace the existing class, rather than simply adding another…

Building a smarter function

```
function addClass(){
      if (element.hasAttribute('class')){
            element.removeAttribute('class');
      }
      element.className = 'highlight';
}
```

This function uses an if clause to check if the element has a class attribute. If it does, it removes that attribute. Finally, a new class attribute is added with the value "highlight".

hasAttribute

This method can be used to check if an element has an attribute of a particular type. In this case, we're checking if it has a class attribute.

removeAttribute

This method is used to remove an attribute of a specified type.

A complete script

1

```
function addClass(){
        if (element.hasAttribute('class')){
                element.removeAttribute('class');
        }
        element.className = 'highlight';
} //function ends
```

2

```
var ul = document.createElement('ul');
var parent = document.querySelector(body);
parent.appendChild(ul);
var li = document.createElement('li');
ul.appendChild(li);
li.addEventListener('click', addClass);
```

The first part of this script (1) will only run when the function addClass is called. The rest of the script (2) will run as soon as the document is loaded in the browser.

The script creates an unordered list and assigns it to a variable called "ul". It then selects the body element, assigning it to a variable called "parent". Next, it adds the unordered list at the end of the body. It then creates a list item, assigning it to a variable called "li" and adds it to the unordered list. Finally, the event listener is added to the list item. Now, when the list item is clicked, the addClass function will run.

JavaScript is fun!

JavaScript mat seem daunting if you've never done any scripting before, but like all languages, it has a relatively simple and logical syntax. Once you learn this and get to know some of the common methods used in DOM Scripting, you'll find that making your web pages come to life can be quite rewarding and maybe even fun.

https://developer.mozilla.org/en-US/docs/Web/JavaScript

```
function endSlideshow(){
    el = document.getElementById("slideshow");
    el.style.display = "none";
    return true;
}
```