

Teddywaddy Code Club

Activity 4e

Accessing the DOM



Accessing the DOM

This exercise will further illustrate how JavaScript can work with the Document Object Model (DOM).

For this activity and onwards it is assumed that VS Code is in use.

Go to <http://www.teddywaddy.com.au/resources.html> and download the following files - js03.html, js03.css and unicorn.svg. Then open the HTML and CSS files in VS Code.

Create an empty js03.js file.

Open the html file in Live Server. You should see an animated unicorn. Study the CSS styles to see how the animation works.

```
.tile1 {  
  top: 15%;  
  left: 15%;  
  background-color: #seashell;  
  animation: flipIt 5s infinite;  
}
```

The animation is applied here.

```
@keyframes flipIt {  
  from{  
    transform: rotateY(0deg);  
  }  
  to{  
    transform: rotateY(360deg);  
  }  
}
```

The animation is defined here.

Add a button to the HTML.

```
<body>  
  <div class="container">  
    <h1>JavaScript DOM Example</h1>  
    <div class="tile tile1" id="tile1"></div>  
    <button class="goBtn">Display/Hide</button>  
  </div>  
  <script src="js03.js"></script>  
</body>
```

Add a button. There is already styling for the button in the CSS called goBtn.

This should add a button to the display, but it doesn't do anything yet.

Now we need to connect some JavaScript to the button so that when the button is clicked a JavaScript function will be called.

Modify the button tag as follows.

```
<button class="goBtn" onclick="goBtn()">Display/Hide</button>
```

Now in the JavaScript file, add the following code.

```
JS js03.js > ...
1
2 function goBtn() {
3
4     console.log("goBtn clicked");
5
6     document.getElementById("tile1Img").style.display = "none";
7
8 }
```

The line numbers don't matter, just enter the code. When the button is clicked a message should appear in the console and the unicorn should disappear (but you can't make it come back).

Understanding the code.

Firstly, remember that *document* refers to all the elements of the whole web page.

The diagram shows the code line `document.getElementById("tile1Img").style.display = "none";` with three callout boxes:

- A top callout box points to the `= "none"` part, containing the text: `= "none"` means setting the LH side of the = to the value "none".
- A bottom-left callout box points to `document.getElementById("tile1Img")`, containing the text: Find the element within the web page that has an id of "tile1Img".
- A bottom-right callout box points to `.style.display`, containing the text: The display property within all style properties of the element.

To make the button more functional, change the JavaScript code as follows.

The diagram shows the updated JavaScript code with a callout box:

```
JS js03.js > ...
1 var imgVisible=true;
2
3 function goBtn() {
4
5     console.log("goBtn");
6
7     if (imgVisible) {
8         document.getElementById("tile1Img").style.display = "none";
9         imgVisible=false;
10    } else {
11        document.getElementById("tile1Img").style.display = "block";
12        imgVisible=true;
13    }
14
15 }
```

A callout box points to the `imgVisible` variable, containing the text: A Boolean variable is used to keep track of whether the unicorn is visible or not.

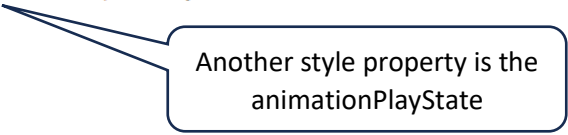
Now add another button to start and stop the animation.

```
<button class="goBtn" onclick="goBtn()">Display/Hide</button>
<button class="pauseContinueBtn" onclick="pauseContinueBtn()">Pause/Continue</button>
```

Once again there is already a CSS style for this button.

Add another function to the JavaScript for the Pause button.

```
function pauseContinueBtn() {  
    document.getElementById("tile1").style.animationPlayState = "paused";  
}
```



Another style property is the animationPlayState

Add another Boolean variable and modify the code so that the new button toggles the animation.