Exercise 10 – Manipulating Styles

Objectives

In this exercise you are going to examine the CSS objects and style settings of a simple dialog box.

Overview

We are going to create a simple dialog box using CSS manipulation. The style object allows us to consistently and effectively manipulate CSS properties for any DOM object.

This exercise is scheduled to run for around **25 minutes**.

Exercise Set Up

1. In VSCode, open the file called **DialogBox.html** and **styles.css** from the **Starter** folder inside **Ex10** of the **Exercises** folder.
2. If you have rebooted your computer or stopped *live-server* from running since the last exercise, then use VSCode's integrated terminal to navigate to the **Exercises** folder and run the command **live-server**.
3. Open your browser (if it doesn't automatically) and navigate to **http://localhost:8080** and follow the path to open **Ex10→Starter→ DialogBox.html**.

Exercise Instructions

Dialog boxes are modal, meaning they freeze out the rest of the page while they are open. We need the **#overlay div** to expand out to fill up 100% of the screen. Then position the inner ** ** inside the container.

1. Study the styles that has been presented in the **styles.css** file

This CSS will expand the overlay to 100% and set the background colour to black. Importantly, the ** ** is absolute in position, so it is out of the normal document flow. Its z-index is set to 1000, which in any normal page would mean it is the highest stack object.

The inner div needs to be positioned at the centre of its container. The margin property can be set to auto, where the browser can calculate the centre of the screen. The margin places the inner div 100px down from the top of its parent. The auto property will dynamically calculate how many pixels of margin are required to the left and the right to keep the div dead centre.

2. Save and test the page in the browser, you should find you have a black background and a central div.
3. Add a property **visibility: hidden;** to the **#overlay** styling in the styles.css.

This will hide the two ** ** elements we are using to create our dialog box when the page first loads.

Now we will get into the JavaScript. If you examine the two **<a>** elements in the page, they both have an onclick attribute that points to an **overlay()** function. When the user clicks on the **<a>** the function fires. This is known as an event handler.

4. Inside **<script>**, create a function called **overlay**.
5. Create 2 variables that point to the ** ** elements we want to manipulate:
   a) Call the first **overlay** targeting the element with the same **id**;
   b) Call the second **dialog** targeting a ** ** that is a *descendent of the element* with **id overlay**.
6. Add a **console.dir** call to display the **overlay** object.
7. Save and access the console in the browser's Developer Tools finding the style property.

The style property is an object literal array in its own right. If you expand the node, you will see the CSS properties the ** ** can support.

We want to control the visibility of the **#overlay**. To achieve this, we need to check whether the object is currently visible.

8. Add the following code at the end of the function:

```
overlay.style.visibility = (overlay.style.visibility == "visible")
? "hidden" : "visible";
```

9. Save and observe the page again.

You should find that clicking on the **<a>** element causes the dialog box to show and clicking it again makes it disappear.

Additional activities

Everything is good as it is, but you might want to improve the layout a little through more CSS manipulation:

1. The black background of the #overlay is a little stark, so let's alter its opacity:

```
if (overlay.style.visibility == "visible") {
                dialog.style.opacity = 0.9;
                overlay.style.opacity = 0.6;
}
```

Perhaps we can utilise a little CSS3 to make things look a bit more modern.

2. Just before the closing bracket of the if statement, add the following code and test it in your browser:

```
dialog.style.borderRadius = "5px";
```

Another useful thing to know about browsers is how their scrollbars work. If there is enough content to fill a screen, a scrollbar will appear. This will either occur in a or the <body>.

3. Add an overflow property, either to the style sheet or the JavaScript, you can hide the scrollbar while the modal box is active.

```
overflow: hidden;
```