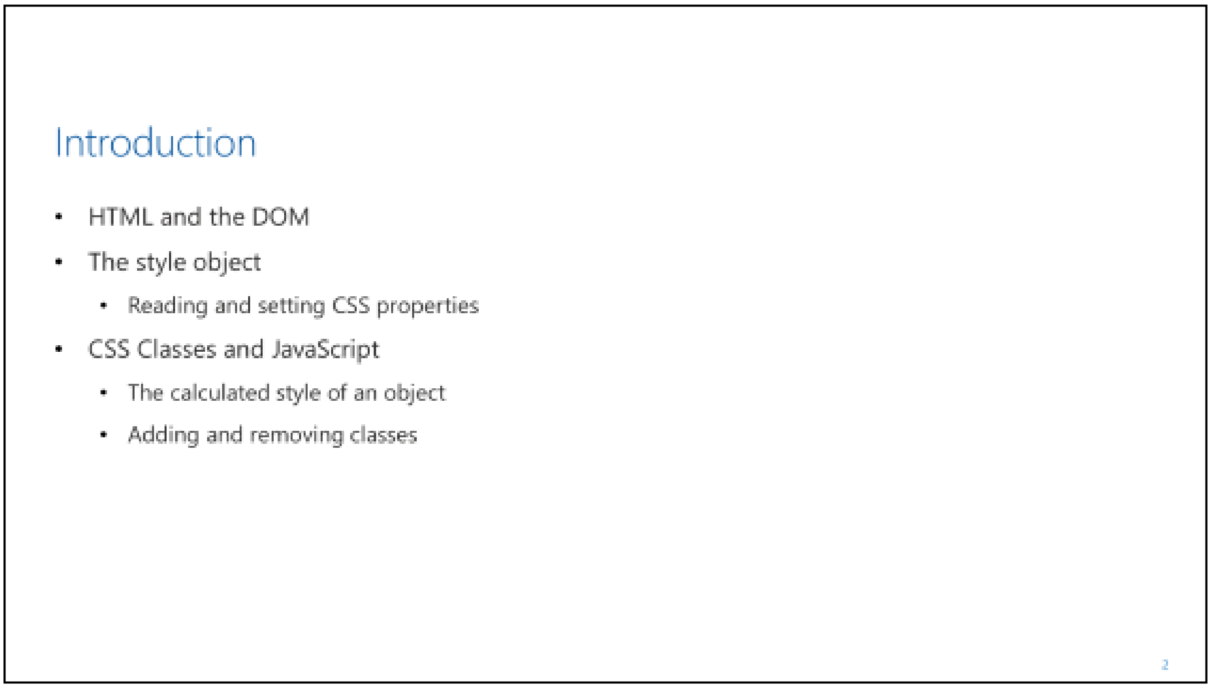
Manipulating Styles

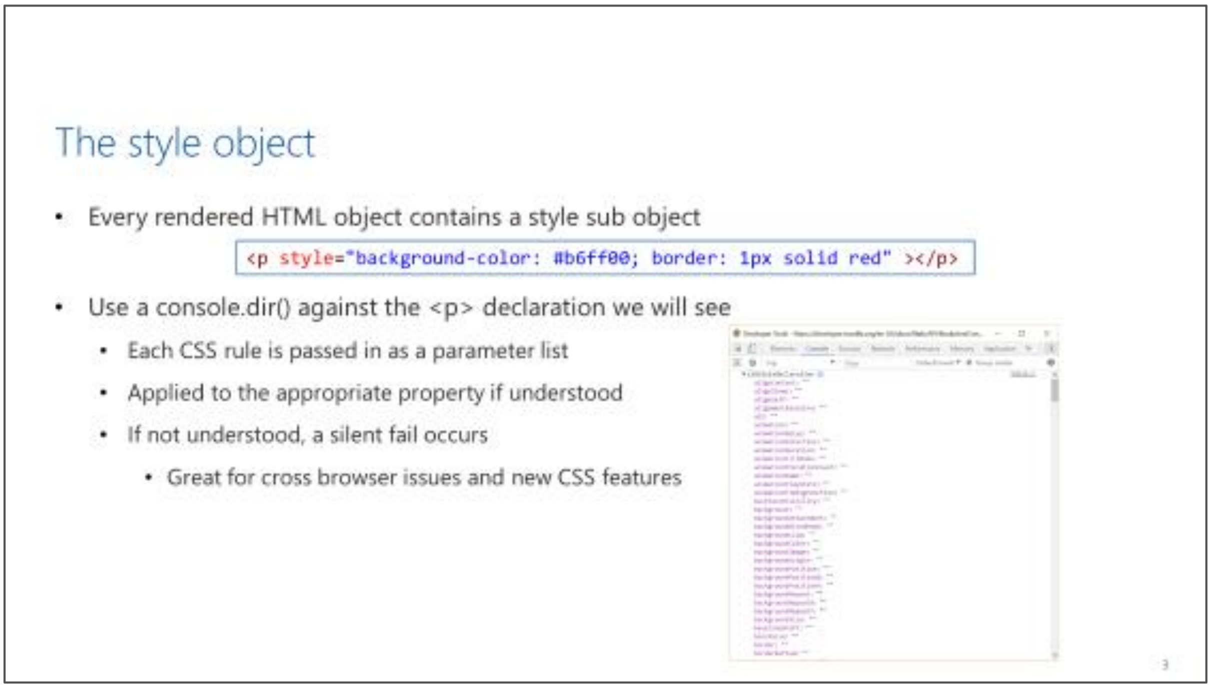
PROGRAMMING WITH JAVASCRIPT



Introduction

- HTML and the DOM
- The style object
  - Reading and setting CSS properties
- CSS Classes and JavaScript
  - The calculated style of an object
  - Adding and removing classes

2



The style object

- Every rendered HTML object contains a style sub object

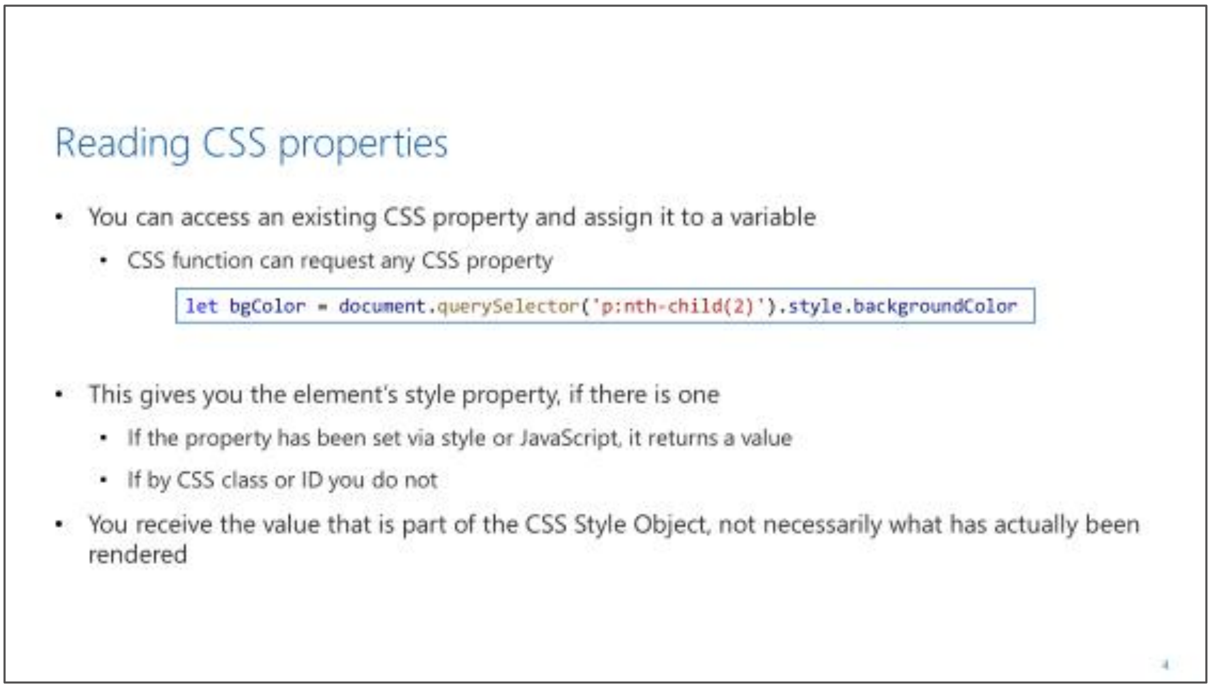
```
<p style="background-color: #b6ff00; border: 1px solid red" ></p>
```

- Use a console.dir() against the <p> declaration we will see
  - Each CSS rule is passed in as a parameter list
  - Applied to the appropriate property if understood
  - If not understood, a silent fail occurs
    - Great for cross browser issues and new CSS features

3

The style property of a HTML element allows us to apply an inline CSS style. The style object is an associative array of key properties and applied styles. As we will see in the next few slides, we can use JavaScript to manipulate the CSS style. Most CSS JavaScript programming occurs using this object and ensures the JavaScript manipulation will always take precedence over any CSS classes or IDs applied to the DOM object.

At execution, the CSS style or JavaScript is examined as a set of parameters and applied to the appropriate style keys. Via HTML rendering, any issues will fail silently. In our own JavaScript code, this is not the case and we should check for the property if we believe it may not be present for any reason.



Reading CSS properties

- You can access an existing CSS property and assign it to a variable
  - CSS function can request any CSS property

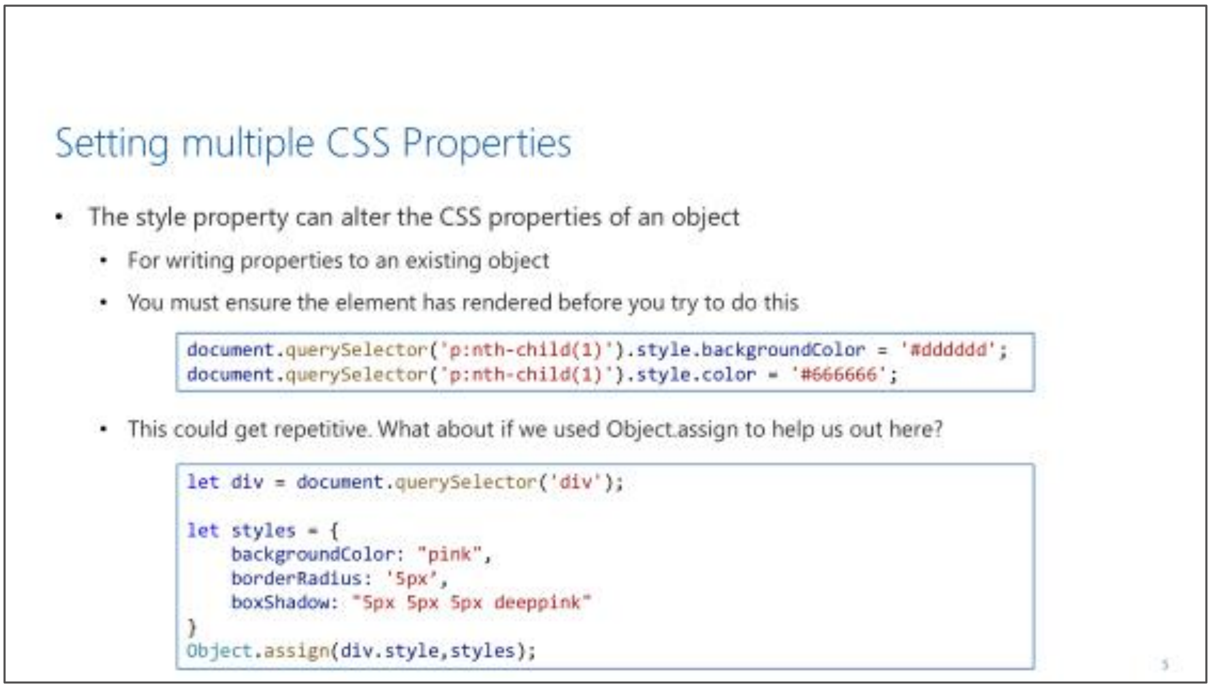
```
let bgColor = document.querySelector('p:nth-child(2)').style.backgroundColor
```

- This gives you the element's style property, if there is one
  - If the property has been set via style or JavaScript, it returns a value
  - If by CSS class or ID you do not
- You receive the value that is part of the CSS Style Object, not necessarily what has actually been rendered

4

CSS properties of multiple elements

You can ask for a CSS property after selecting multiple elements, but this is almost always a bad idea: a function can only return a single result, so you'll still only obtain the property for the first matched element.



Setting multiple CSS Properties

- The style property can alter the CSS properties of an object
  - For writing properties to an existing object
  - You must ensure the element has rendered before you try to do this

```
document.querySelector('p:nth-child(1)').style.backgroundColor = '#dddddd';
document.querySelector('p:nth-child(1)').style.color = '#666666';
```

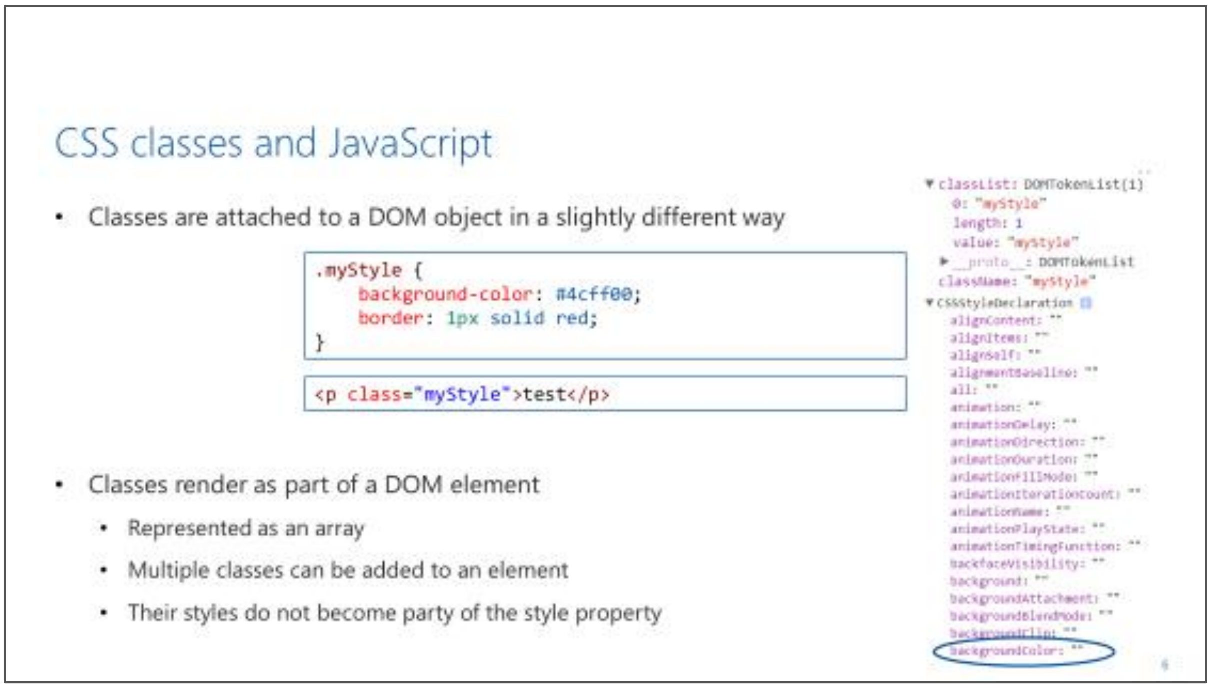
  - This could get repetitive. What about if we used Object.assign to help us out here?

```
let div = document.querySelector('div');

let styles = {
    backgroundColor: "pink",
    borderRadius: '5px',
    boxShadow: "5px 5px 5px deeppink"
}
Object.assign(div.style,styles);
```

5

The advantage of using the style object is its authority in the hierarchy of CSS application – even if a CSS class is applied, this will override those rules. Each style rule must be applied as a separate function call, which can end up being very weighty in code.



CSS classes and JavaScript

- Classes are attached to a DOM object in a slightly different way

```
.myStyle {
    background-color: #4cff00;
    border: 1px solid red;
}
```

```
<p class="myStyle">test</p>
```

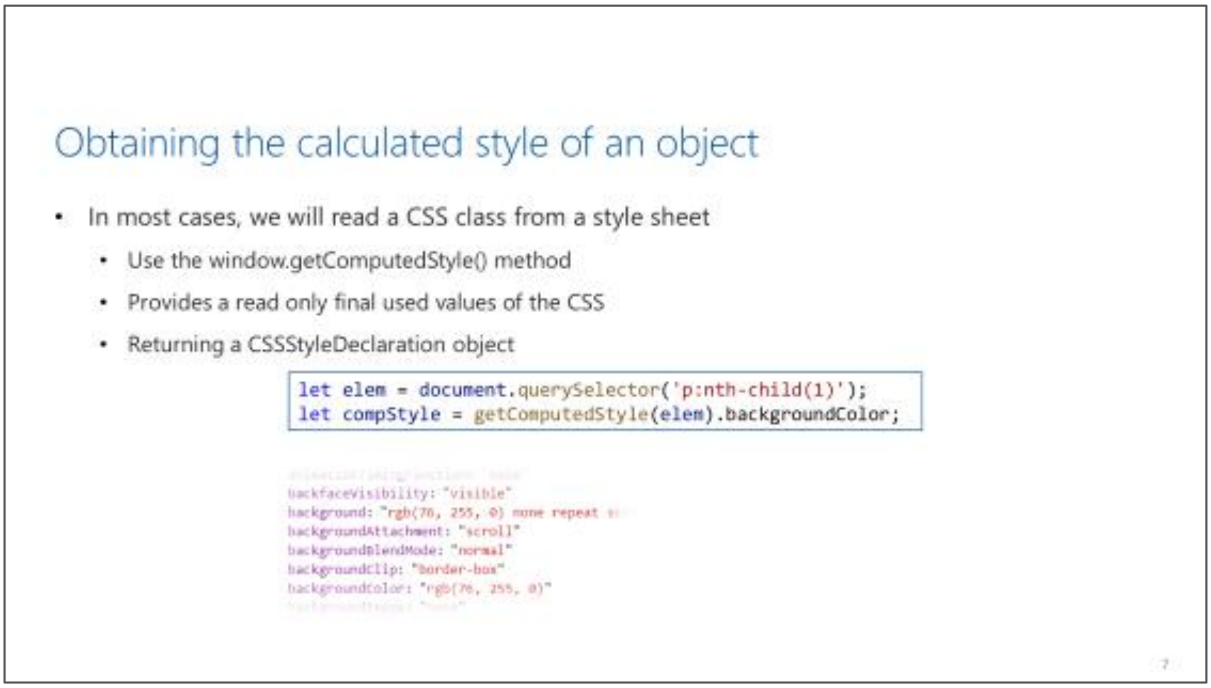
- Classes render as part of a DOM element
  - Represented as an array
  - Multiple classes can be added to an element
  - Their styles do not become party of the style property

```
▼classList: DOMTokenList(1)
   0: "myStyle"
   length: 1
   value: "myStyle"
 ▶__proto__: DOMTokenList
 className: "myStyle"
▼CSSStyleDeclaration
   alignContent: ""
   alignItems: ""
   alignSelf: ""
   alignmentBaseline: ""
   all: ""
   animation: ""
   animationDelay: ""
   animationDirection: ""
   animationDuration: ""
   animationFillMode: ""
   animationIterationCount: ""
   animationName: ""
   animationPlayState: ""
   animationTimingFunction: ""
   backfaceVisibility: ""
   background: ""
   backgroundAttachment: ""
   backgroundBlendMode: ""
   backgroundClip: ""
   backgroundColor: ""
```

6

In all probability, we will be using CSS classes and in that situation, as outlined above. CSS classes apply to the DOM object, but we will not see it in the style property of the object.

Trying to retrieve the a CSS style property via JavaScript will only work when that style property has previously been set via JavaScript or when that style property was defined inline by using the style attribute in HTML. Now this doesn't do you much good if you haven't first set said CSS property in JavaScript and you still want to retrieve it via JavaScript.



Obtaining the calculated style of an object

- In most cases, we will read a CSS class from a style sheet
  - Use the window.getComputedStyle() method
  - Provides a read only final used values of the CSS
  - Returning a CSSStyleDeclaration object

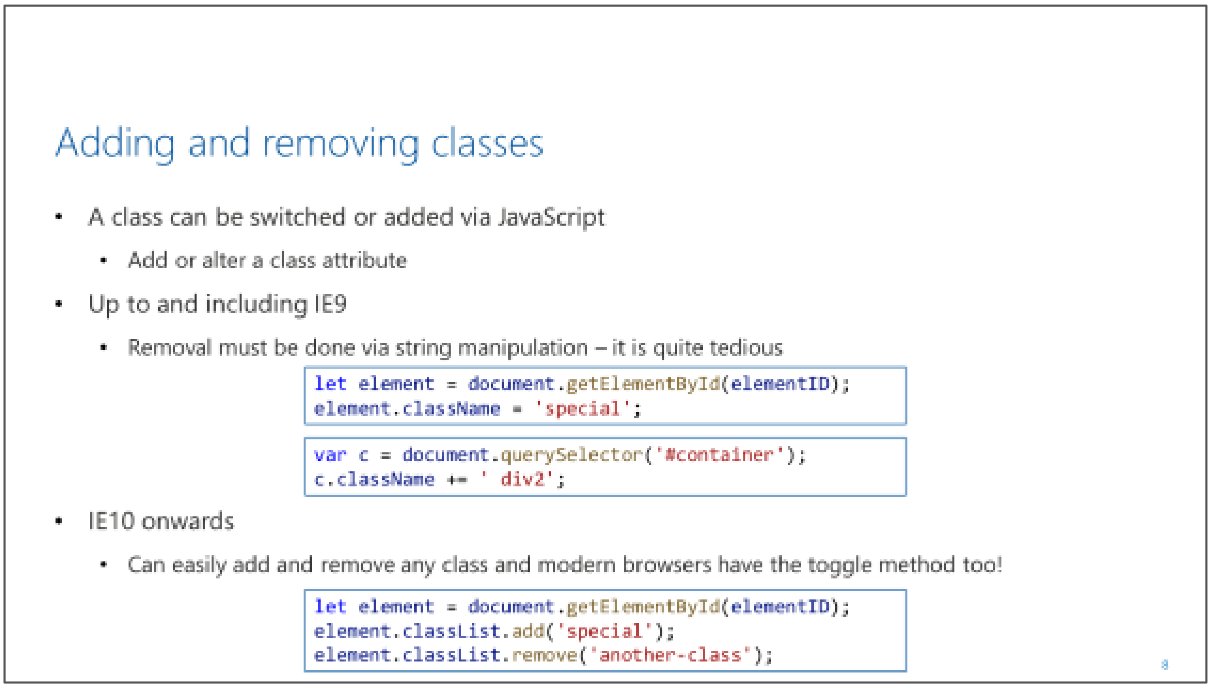
```
let elem = document.querySelector('p:nth-child(1)');
let compStyle = getComputedStyle(elem).backgroundColor;
```

```
backfaceVisibility: "visible"
background: "rgb(76, 255, 0) none repeat s
backgroundAttachment: "scroll"
backgroundBlendMode: "normal"
backgroundClip: "border-box"
backgroundColor: "rgb(76, 255, 0)"
```

7

getComputedStyle() gives the final used values of all the CSS properties of an element, where the returned style is a **CSSStyleDeclaration** object.

The returned object is of the same type as that of the object returned from the element's style property; however, the two objects have different purposes. The object returned from **getComputedStyle** is read-only and can be used to inspect the element's style (including those set by a <style> element or an external stylesheet). The style object should be used to set styles on a specific element.



Adding and removing classes

- A class can be switched or added via JavaScript
  - Add or alter a class attribute
- Up to and including IE9
  - Removal must be done via string manipulation – it is quite tedious

```
let element = document.getElementById(elementID);
element.className = 'special';
```

```
var c = document.querySelector('#container');
c.className += ' div2';
```

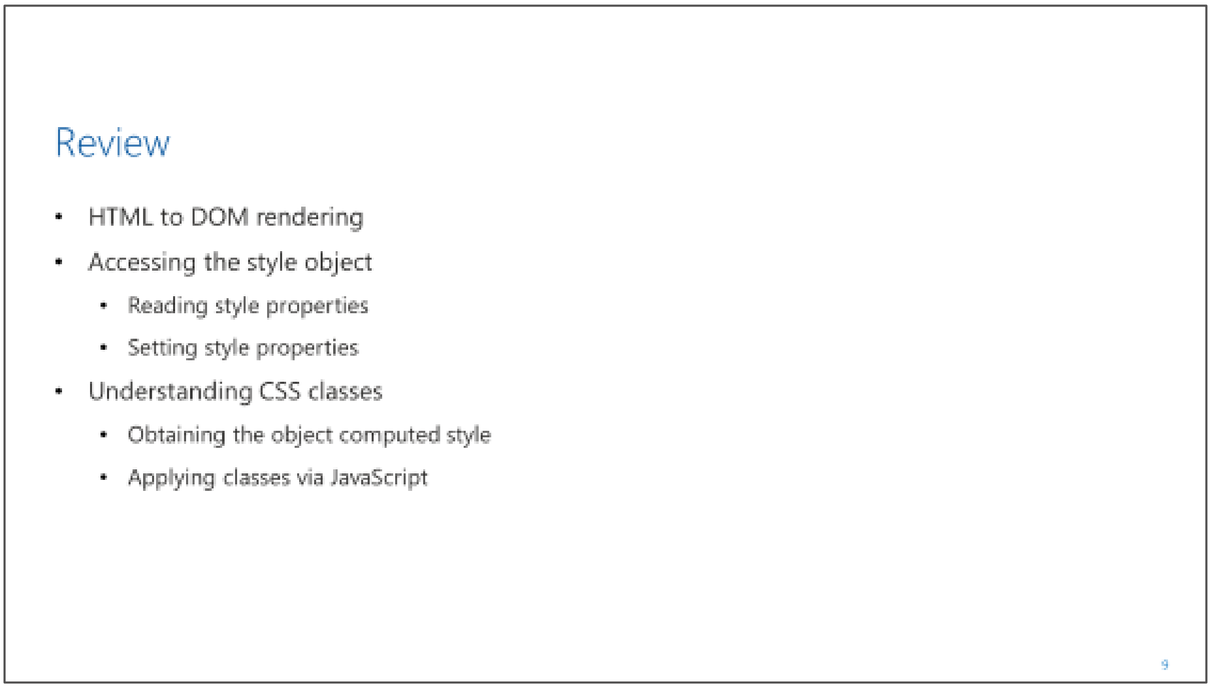
- IE10 onwards
  - Can easily add and remove any class and modern browsers have the toggle method too!

```
let element = document.getElementById(elementID);
element.classList.add('special');
element.classList.remove('another-class');
```

8

All elements can have attributes added or changed via JavaScript. The setAttribute method takes an attribute value and sets it to the new value. Any CSS class that is currently loaded into the page can be referenced in this way.

There is an alternative approach that can be used if you want to apply multiple classes. The DOM element holds all applied classes in the className property. It is simply a string value that we can append to.



Review

- HTML to DOM rendering
- Accessing the style object
  - Reading style properties
  - Setting style properties
- Understanding CSS classes
  - Obtaining the object computed style
  - Applying classes via JavaScript

9

Exercise

- Creating a modal dialog box

10