CSY1018
Web Development

Tom Butler
thomas.butler@northampton.ac.uk

Topic 4

- Finding elements by class name
- A chance to test out everything you've learnt so far

Setting CSS classes using javascript

- You can set a CSS class on a HTML Element using javascript
- This is similar to setting a style, however it uses the syntax:

```
var element = document.getElementById('circle');
element.className = 'nameOfClass';
```

Setting CSS classes using javascript

- This allows you to define a set of CSS rules in the stylesheet that can be applied with a single line of javascript:

```
.semiTransparentAndBlue {
  background-color: blue;
  Opacity: 0.5;
}
```

```
var element = document.getElementById('circle');
element.className = 'semiTransparentAndBlue';
```

Setting CSS classes using javascript

- You can apply more than one CSS class to an element by separating the class names with a space:

```
.semiTransparent {
  opacity: 0.5;
}

.blue {
   background-color: blue;
}
```

```
  var element = document.getElementById('circle');
  element.className = 'semiTransparent blue';
```

Finding Elements using CSS Classes

- So far to find elements on the page, we have been using the code
  - document.getElementById()
- This finds an element on the page using the HTML ID attribute
- It is also possible to find elements on the page using the CSS class name

Finding Elements using CSS Classes

- There is one major difference between IDs and classes:
  - An ID should be unique. There will only be one element on the page with each ID
  - Classes are not, multiple elements can have the same class

Finding Elements using CSS Classes

- document.getElementById() will always retrieve a single element from the page
- However, this is not possible with classes because there may be more than one element on the page with the same class

```
<div class="circle">
</div>
<div class="circle">
</div>
```

Finding Elements using CSS Classes

```
<div class="circle">
  </div>
  <div class="circle">
  </div>
```

- Which element would you expect the following code to find?

```
var element = document.getElementByClassName('circle');
```

Finding Elements using CSS Classes

- Because it's not clear, there is no function
- **getElementByClassName**()
- Instead, the function is
- **getElementsByClassName()**
- That is **getElement**S instead of element (no S)
- This will retrieve more than one element!

```
<div class="circle">
  </div>
  <div class="circle">
  </div>
```

```
var elements = document.getElementsByClassName('circle');
```

- Because more than one element is retrieved, if you want to make changes to one, you have to specify which element that was matched you'd like to change
- This is done using:

```
var elements = document.getElementsByClassName('circle');
elements[0].style.backgroundColor = 'blue';
```

- elements[0] is the first found element
- elements[1] is the second found element
- elements[2] is the third
- etc

```
<div class="circle">

</div>

<div class="circle">

</div>
var elements = document.getElementsByClassName( 'circle' );
elements[0].style.backgroundColor = 'blue';

elements[1].style.backgroundColor = 'green';
```

Exercise 1a

- 90-120 Minutes
- Download Topic3/Exercise1-Exercise from GitHub
- 1) When one of the heads in the sidebar is clicked, set the head of the player to the relevant image. You will need to use getElementsByClassName() to find the player's head, as it doesn't have an ID. Do not give the head element an ID
- Hint: You need to set the head's CSS background-image property e.g.

```
elements[0].style.backgroundImage = 'url(img/head0.png)';
```

- 2a) Do this for all the heads, clicking the element with the ID head0 should set the player's head to head0.png, the element with the ID head1 should set the player's head to head1.png, etc
- 2b) Do the same for the bodies so the body image can be changed on click, apply this to all 4 body images

Exercise 1b

- 3) There are several CSS classes that can be applied to the element with the ID `player`:
  - standLeft
  - standRight
  - standDown
  - StandUp
- 3a) Try setting these in the HTML. You will need to set the class to "character standLeft", "character standRight", etc
- 3b) When the arrow keys are pressed, set the relevant direction on the player element

Exercise 1c

- There are other classes:
  - walkUp
  - walkDown
  - walkLeft
  - walkRight
- 1) When the arrow keys are pressed, the character should walk around the screen and the relevant walking animation should be displayed.
- 2) When the arrow keys are not being pressed the player should be still and facing the direction they were last movement
- 3) You should be able to walk diagonally by pressing two keys at once

Exercise 1d

- 1) When clicking the "X" at the top of the sidebar, slide it off the screen to the right
- 2) When the player is clicked on, re-open the sidebar by sliding it in

Exercise 1e

- Extra exercises (no solutions available):
  - Stop the player leaving the screen
  - Prevent the player from walking over the tree
  - Difficulty: Hard: Make the opponent walk around the screen applying appropriate animations
  - Difficulty: Hard Add more trees that block the player (and opponent), make a more interesting level e.g. a maze

Exercise 1a - solution

```
function myLoadFunction() {

  var element = document.getElementById('head0');
  element.addEventListener('click', setHead0);

  var element = document.getElementById('head1');
  element.addEventListener('click', setHead1);

  var element = document.getElementById('head2');
  element.addEventListener('click', setHead2);

  var element = document.getElementById('head3');
  element.addEventListener('click', setHead3);

  var element = document.getElementById('head');
  element.addEventListener('click', setHead4);
}

document.addEventListener('DOMContentLoaded',
                  myLoadFunction);
```

```
function setHead0() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head0.png)';
}

function setHead1() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head1.png)';
}

function setHead2() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head2.png)';
}

function setHead3() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head3.png)';
}

function setHead4() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head4.png)';
}
```

Repetition

- This solution involves a lot of repeated code
- You have to make a function for each click event
- Each function shares 99% of the same code
- All that changes is a single number!

```
function setHead0() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head0.png)';
}

function setHead1() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head1.png)';
}
```

Repetition

- It would be better if a single function could be written that worked for every head.
- It is possible to assign the same function to the click event for multiple elements:

```
var element = document.getElementById('head0');
element.addEventListener('click', setHead);

var element = document.getElementById('head1');
element.addEventListener('click', setHead);

var element = document.getElementById('head2');
element.addEventListener('click', setHead);

var element = document.getElementById('head3');
element.addEventListener('click', setHead);

var element = document.getElementById('head');
element.addEventListener('click', setHead);
```

```
function setHead() {
  var element = document.getElementsByClassName('head')[0];
  element.style.backgroundImage = 'url(images/head1.png)';
}
```

Repetition

- This will work, each time **any** of the head images is clicked on, it will set the player's head to head1.png
- This isn't quite what we want
- However, the element's ID contains the name of the file we want to set:

```
<p>Select Head</p>
<ul class="heads">
      <li id="head0"></li>
      <li id="head1"></li>
      <li id="head2"></li>
      <li id="head3"></li>
      <li id="head4"></li>
</ul>
```

Repetition

- It is possible to know which element was clicked on inside the event listener
- There is a special variable that always exists when an event listener is fired
- The variable is `this`
- The this variable is available inside every event listener function automatically
- And stores a reference to the element that was clicked on:

```
function clickHead() {
        this.style.backgroundColor = 'red';
}

function myLoadFunction() {

        var element = document.getElementById('head0');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head1');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head2');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head3');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head4');
        element.addEventListener('click', clickHead);
}
```

Reading an element's ID

- It's also possible to read an element's ID using
- element.id
- This can be done either when an element is found using getElementById() or the *this* reference
-

```
function clickHead() {
        alert(this.id);
}

function myLoadFunction() {

        var element = document.getElementById('head0');
        element.addEventListener('click', clickHead);


….
```

Exercise 2

- 10 – 15 minutes
- Reduce the amount of code from Exercise 1
- 1) Add a single function that can be used as the event listener for each head.
  - By Reading the id attribute the function should set the relevant head image when clicked: Clicking head0 should set head0.png, clicking head1 should set head1.png, etc
- 2) Do the same for bodies

Exercise 2 - Solution

```
function clickHead() {
        var elements = document.getElementsByClassName('head');
        elements[0].style.backgroundImage = 'url(images/' + this.id +  '.png)';

}

function myLoadFunction() {

        var element = document.getElementById('head0');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head1');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head2');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head3');
        element.addEventListener('click', clickHead);

        var element = document.getElementById('head4');
        element.addEventListener('click', clickHead);
}
```

Repetition

- There is still a lot of repetition assigning the event listeners to each head element:

```
var element = document.getElementById('head0');
element.addEventListener('click', clickHead);

var element = document.getElementById('head1');
element.addEventListener('click', clickHead);

var element = document.getElementById('head2');
element.addEventListener('click', clickHead);

var element = document.getElementById('head3');
element.addEventListener('click', clickHead);

var element = document.getElementById('head4');
element.addEventListener('click', clickHead);
```

Repetition

- Like **getElementsByClassName()** there is also a function **getElementsByTagName()**
- This allows you to find all the elements with a specific tag name e.g. <p> or <h1>
- All of the elements for the head images are <li> elements

```
<p>Select Head</p>
<ul class="heads">
      <li id="head0"></li>
      <li id="head1"></li>
      <li id="head2"></li>
      <li id="head3"></li>
      <li id="head4"></li>
</ul>
```

Repetition

- It is possible to select all the <li> elements on the page using the code:

```
var elements = document.getElementsByTagName('li');
```

Like getElementsByClassName this will retrieve multiple elements that can be accessed via an index number starting from zero

```
elements[0].addEventListener('click', clickHead);
elements[1].addEventListener('click', clickHead);
```

```
function myLoadFunction() {

   var element = document.getElementById('head0');
   element.addEventListener('click', clickHead);
   var element = document.getElementById('head1');

   element.addEventListener('click', clickHead);

   var element = document.getElementById('head2');
   element.addEventListener('click', clickHead);

   var element = document.getElementById('head3');
   element.addEventListener('click', clickHead);

   var element = document.getElementById('head4');
   element.addEventListener('click', clickHead);
}
```

```
function myLoadFunction() {

   var elements = document.getElementsByTagName('li');
   elements[0].addEventListener('click', clickHead);
   elements[1].addEventListener('click', clickHead);
   elements[2].addEventListener('click', clickHead);
   elements[3].addEventListener('click', clickHead);
   elements[4].addEventListener('click', clickHead);


}
```

Repetition

- This solution is a lot better, however there is still repetition

```
function myLoadFunction() {

   var elements = document.getElementsByTagName('head0');
   elements[0].addEventListener('click', clickHead);
   elements[1].addEventListener('click', clickHead);
   elements[2].addEventListener('click', clickHead);
   elements[3].addEventListener('click', clickHead);
   elements[4].addEventListener('click', clickHead);


}
```

Loops

- All programming languages provide a way to create a counter
- This is done using a *loop*
- In Javascript you can create a loop like this:

```
for (var i = 0; i < 10; i++) {
            console.log(i);
}
```

Loops

- A loop has 3 parts:
  - A starting value
  - A condition
  - A counter

```
for (var i = 0; i < 10; i++) {
            console.log(i);
}
```

```
for (var i = 0; i < 10; i++) {
            console.log(i);
}
```

```
0
1
2
3
4
5
6
7
8
```

- Changing the starting value affects the first value in the loop:

```
for (var i = 0; i < 10; i++) {
            console.log(i);
}
```

```
Output (in console)

3
4
5
7
8
9
```

```
for (var i = 0; i < 10; i++) {
            console.log(i);
}
```

```
0
1
2
3
4
5
6
7
8
```

- Changing the condition affects when the loop stops

```
for (var i = 0; i < 7; i++) {
            console.log(i);
}
```

```
Output (in console)
0
1
2
3
4
5
6
```

```
for (var i = 0; i < 10; i++) {
            console.log(i);
}
```

```
0
1
2
3
4
5
6
7
8
```

- Changing the counter changes how much is incremented each time (++ means add 1)

```
for (var i = 0; i < 10; i=i+2) {
            console.log(i);
}
```

```
Output (in console)
0
2
4
6
8
```

Loops and elements

- It's possible to use a variable in place of the number when selecting an element by an index:

```
function myLoadFunction() {

   var elements = document.getElementsByTagName('li');
   var num = 3;
   elements[num].addEventListener('click', clickHead);

}
```

Loops and elements

- By combining a loop with getElementsByTag name it's possible to apply the same thing to each element that was matched:

```
var elements = document.getElementsByTagName('li');
for (var i = 0; i < 10; i++) {
  elements[i].style.backgroundColor = 'blue';
}
```

Exercise 3

- 5 minutes
- Use a loop to assign the event listener to all the heads when they are clicked.

Exercise 3 - solution

```
function clickHead() {
        var elements = document.getElementsByClassName('head');
        elements[0].style.backgroundImage = 'url(images/' + this.id +  '.png)';
}

function myLoadFunction() {

        var elements = document.getElementsByTagName('li');

        for (var i = 0; i < 10; i++) {
                elements[i].addEventListener('click', clickHead);
        }
}
```

Loops and elements

- This is a lot shorter than assigning each click event manually

```
function clickHead() {
        var elements = document.getElementsByClassName('head');
        elements[0].style.backgroundImage = 'url(images/' + this.id +  '.png)';
}

function myLoadFunction() {

        var elements = document.getElementsByTagName('li');

        for (var i = 0; i < 10; i++) {
                elements[i].addEventListener('click', clickHead);
        }
}
```

```
function myLoadFunction() {
   var elements = document.getElementsByTagName('li');
   elements[0].addEventListener('click', clickHead);
   elements[1].addEventListener('click', clickHead);
   elements[2].addEventListener('click', clickHead);
   elements[3].addEventListener('click', clickHead);
   elements[4].addEventListener('click', clickHead);
 }
```

Loops and elements

- Running this code could cause an error

```
function myLoadFunction() {

        var elements = document.getElementsByTagName('li');

        for (var i = 0; i < 10; i++) {
                elements[i].addEventListener('click', clickHead);
        }
}
```

- The number of <li> elements may not be 10
- If there were fewer than 10 <li> elements this would cause an error because the element does not exist

```
                elements[10].addEventListener('click', clickHead);
```

Loops and elements

- It's possible to enquire how many elements were retrieved from getElementsByClassName and getElementsByTagName using the code

```
var elements = document.getElementsByTagName('li');
console.log(elements.length);
```

- Elements.length stores the number of elements that were retrieved from the page

Loops and elements

-
- The code that selects the elements

```
var elements = document.getElementsByTagName('li');
```

- Will select all the <li> elements on the page
- This will include both the <li> elements that contain the heads and the <li> elements that contain the bodies
- Clicking on a body will change the head!

getElementsbyTagName

- This is because we are calling:
  - document.getElementsByTagName()
- This will find all the elements inside the document
- Instead, it's possible to call getElementsByTagName on a particular element

getElementsbyTagName

```
<div class="div1">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>

<div class="div2">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>
```

```
var elements = document.getElementsByTagName('p');
```

- This will select all the <p> elements on the page

```
<div class="div1">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>

<div class="div2">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>
```

getElementsbyTagName

It's possible to find an element on the page, and then find elements inside it:

```
<div class="div1">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>

<div class="div2">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>
```

```
        var div1 = document.getElementsByClassName('div1');
        var elements = div1[0].getElementsByTagName('p');
```

```
<div class="div1">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>

<div class="div2">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
</div>
```

Exercise 4

- Extend exercise 3 to assign an event listener to both heads and bodies
- You will need to use elements.length to alter the loop condition

Exercise 4 - Solution

```
function clickBody() {
        var elements = document.getElementsByClassName('body');
        elements[0].style.backgroundImage = 'url(images/' + this.id +  '.png)';
}

function clickHead() {
        var elements = document.getElementsByClassName('head');
        elements[0].style.backgroundImage = 'url(images/' + this.id +  '.png)';
}

function myLoadFunction() {

        var heads = document.getElementsByClassName('heads');
        var elements = heads.getElementsByTagName('li');

        for (var i = 0; i < elements.length; i++) {
                elements[i].addEventListener('click', clickHead);
        }


        var heads = document.getElementsByClassName('heads');
        var elements = heads.getElementsByTagName('li');

        for (var i = 0; i < elements.length; i++) {
                elements[i].addEventListener('click', clickHead);
        }
}
```

Exercise 5

- Amend the code to assign both the head and body click listeners using loops, use getElementsByClassName to get the <ul> elements then elements[0].getElementsByTagName() to get the <li> elements