CSY1018
Web Development

Tom Butler
thomas.butler@northampton.ac.uk

Topic 6

- Quick recap of last week
- Debugging and common errors
- DOM (Document Object Model) introduction
- Selecting any element from the document

Random Numbers

- In Javascript random numbers can be generated using the code:

```
var randomNumber = Math.random();
```

- This will store a random number between 0 and 1 inside the variable randomNumber
- e.g. 0.23445466 or 0.93445477

Random Numbers

- This will always generate a random number between 0 and 1
- In most cases this value will not be useful on its own
- However, it's possible to change the maximum value by multiplying the random number by your chosen maximum:

Random Numbers

- This will create a random number between 0 and 10

```
var randomNumber = Math.random() * 10;
```

- e.g.
  - 8.23344564
  - 2.04933531
  - 0.12467897
  - 9.46589342

Random Numbers

- The chance of a whole number e.g. 5.00000 being generated is tiny
- Most of the time you will want a whole number e.g. 1,2,3 etc
- It's possible to round the generated number by using the Math.round() function:

```
var randomNumber = Math.round(Math.random() * 10);
```

Random Numbers

- Instead of Math.round() there are two other rounding methods:
  - Math.ceil(); which rounds up
  - Math.floor(); which rounds down
- To get a random number between 1 and 10 (inclusive) you can use

```
var randomNumber = Math.ceil(Math.random() * 10);
```

- Or for 0-9 you can use:

```
var randomNumber = Math.floor(Math.random() * 10);
```

Arrays

- So far, most variables you have assigned a value to has contained a *single value* e.g.

```
        var number = 4;
        var string = 'hello world';
        var element = document.getElementById('mydiv');
        var diceRoll = Math.ceil(Math.random() * 6);
```

- A different type of variable known as an *array* lets you store more than one value in a single variable

Arrays

- You have already used an array when using getElementsByTagName() and getElementsByClassName:

```
<div class="circle">

  </div>

  <div class="circle">

  </div>
```

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

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

Arrays

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

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

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

- In this example, the elements variable is an *array*
- The variable stores more than one element

Arrays

- You can also create your own arrays
- Arrays can store any type of variable
- Arrays are used to map one value to another

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

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

- Here, 0 and 1 are mapped to *elements* on the page

Creating an array

- To create an array you use two square brackets:

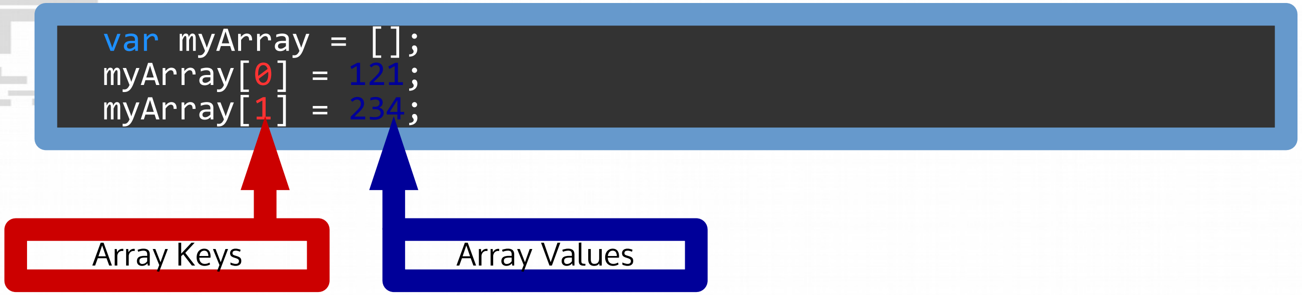
```
var myArray = [];
```

- This will create an empty array, you can then write values to individual indexes. e.g.

```
var myArray = [];
myArray[0] = 121;
myArray[1] = 234;
```

Creating an array

- An array consists of two parts:
  - Keys – The value you will use to access the value
  - Values – The value stored under the named key



```
var myArray = [];
myArray[0] = 121;
myArray[1] = 234;
```

Creating an array

- You can store any type of variable in the array e.g. strings

```
  var myArray = [];
  myArray[0] = 'green';
  myArray[1] = 'blue';
```

- Once the array is created, you can reference the value like any other variable:

```
alert(myArray[1]); // prints “blue” in an alert box
```

Creating an array

```
var myArray = [];
myArray[0] = 'green';
myArray[1] = 'blue';
```

- You can also use a variable in place of the numerical key

```
var num = 1;
alert(myArray[num]); // prints “blue” in an alert box
```

```
var num = 0;
alert(myArray[num]); // prints “green” in an alert box
```

Exercise 1

- 15-20 Minutes
- 1) Create a HTML page with two elements: A <div> and a <a>. Add relevant CSS to make the div a 100x100px red square. The link should say "Click me"
- 2) When the link is clicked on, generate a random number than selects a random colour from an array and sets the div's background colour accordingly
- 3) **Optional** Also select a random border and border size/shape e.g. randomly generate
  - border: 2px solid red;
  - border: 8px dotted green;
  - border: 1px dashed yellow;
- 4) **Optional** also add randomized size, width, height as well as border-radius and opacity

Exercise 1 - Solution

```html
<!doctype html>
<html>
  <head>
    <title>Exercise 1</title>
    <script src="ex1.js"></script>
    <link rel="stylesheet"
            href="style.css" />
  </head>
  <body>
  <div>
  </div>

  <a>Click me</a>

  </body>

</html>
```

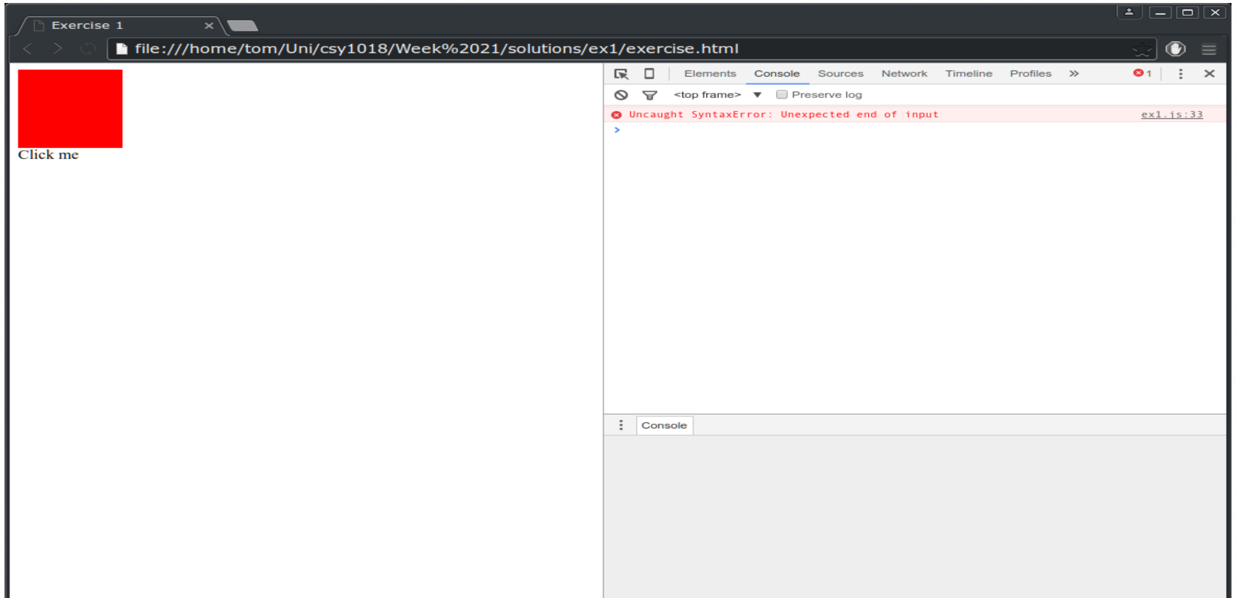
```javascript
function linkClick() {
    var colourArray = [];
    colourArray[0] = 'red';
    colourArray[1] = 'blue';
    colourArray[2] = 'green';
    colourArray[3] = 'yellow';
    colourArray[4] = 'white';
    colourArray[5] = 'black';

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

    var randomNumber = Math.floor(Math.random() * 6);
    elements[0].style.backgroundColor = colourArray[randomNumber];

    var borderArray = [];
    borderArray[0] = 'solid';
    borderArray[1] = 'dashed';
    borderArray[2] = 'dotted';

    var randomBorder = Math.floor(Math.random() * 3);
    var randomSize = Math.floor(Math.random() * 15);
    var randomColour = Math.floor(Math.random() * 6);

    elements[0].style.border = randomSize + 'px ' +
      borderArray[randomBorder] + ' ' + colourArray[randomColour];
}

function myLoadEvent() {
    var elements = document.getElementsByTagName('a');
    elements[0].addEventListener('click', linkClick);
}

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

Debugging

- If your code isn't working as you expected (e.g. nothing is happening when you click on an element and you're expecting it to do something) firstly open the developer tools (Usually F12 on the keyboard or open the menu and select developer tools)

Debugging

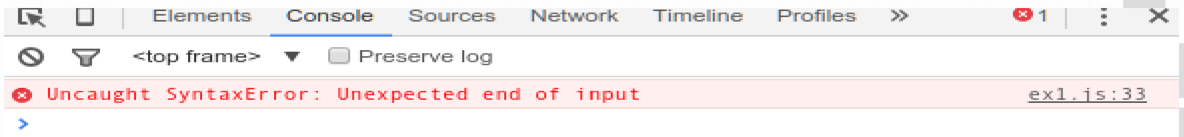
- Select "Console" and you'll see the Javascript console
- Any errors in your code will be highlighted here



```
Uncaught SyntaxError: Unexpected end of input                ex1.js:33
```

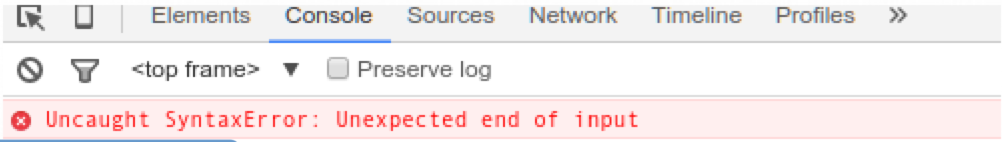
Common error #1

- Unexpected end of input
- Every opening brace { in javascript requires a closing brace
- This error means that you are missing a closing brace somewhere



```
Uncaught SyntaxError: Unexpected end of input                     ex1.js:33
>
```

Common error #1

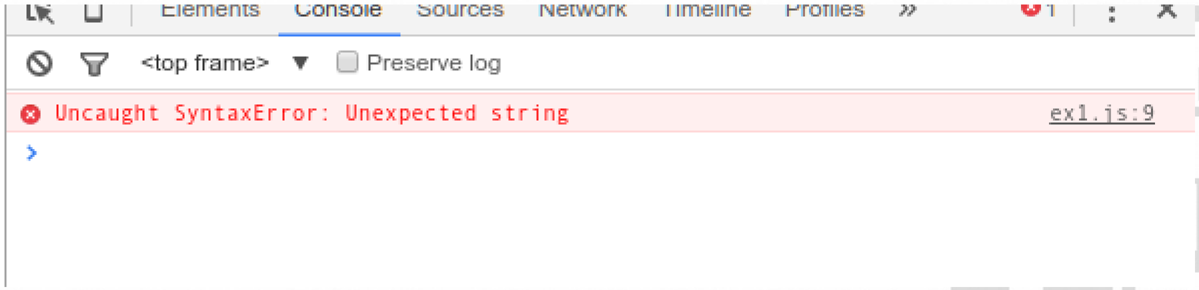


```
Uncaught SyntaxError: Unexpected end of input
```

```
function linkClick() {
    var colourArray = [];
    colourArray[0] = 'red';
    colourArray[1] = 'blue';
    colourArray[2] = 'green';
    colourArray[3] = 'yellow';
    colourArray[4] = 'white';
    colourArray[5] = 'black';

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

    var randomNumber = Math.floor(Math.random() * 6);
    elements[0].style.backgroundColor = colourArray[randomNumber];

    var borderArray = [];
    borderArray[0] = 'solid';
    borderArray[1] = 'dashed';
    borderArray[2] = 'dotted';

    var randomBorder = Math.floor(Math.random() * 3);
    var randomSize = Math.floor(Math.random() * 15);
    var randomColour = Math.floor(Math.random() * 6);

    elements[0].style.border = randomSize + 'px ' +
      borderArray[randomBorder] + ' ' + colourArray[randomColour];


function myLoadEvent() {
    var elements = document.getElementsByTagName('a');
    elements[0].addEventListener('click', linkClick);
}

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

Common error #2

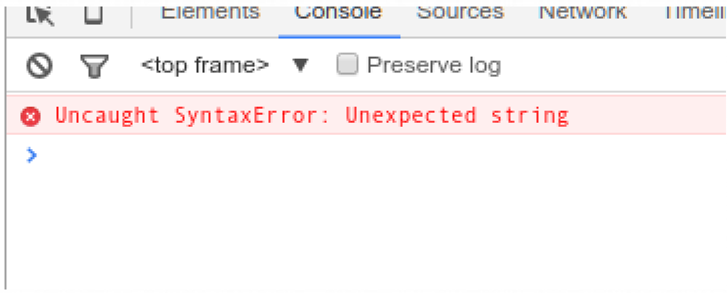
- Unexpected string/Unexpected variable/unexpected number
- This means there is a syntax error, you are missing part of the code or have the code in the wrong order



```
Uncaught SyntaxError: Unexpected string                                ex1.js:9
```

Common error #2

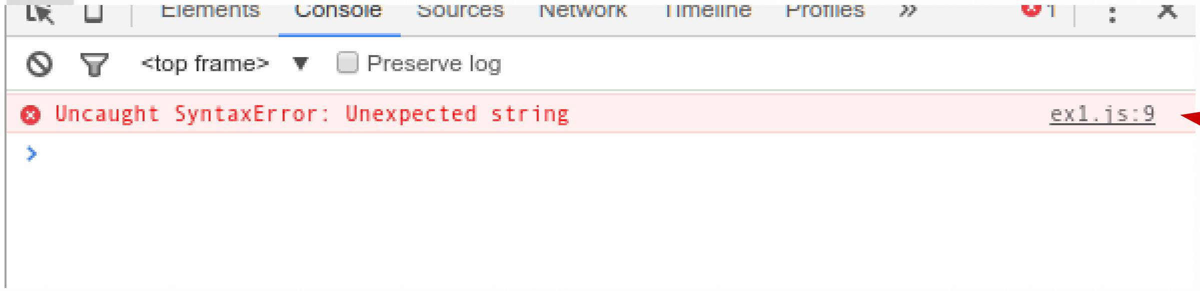
```
function linkClick() {
    var colourArray = [];
    colourArray[0] = 'red';
    colourArray[1] = 'blue';
    colourArray[2] 'green';
    colourArray[3] = 'yellow';
    colourArray[4] = 'white';
    colourArray[5] = 'black';

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

    var randomNumber = Math.floor(Math.random() * 6);
    elements[0].style.backgroundColor = colourArray[randomNumber];

    var borderArray = [];
    borderArray[0] = 'solid';
    borderArray[1] = 'dashed';
    borderArray[2] = 'dotted';

    var randomBorder = Math.floor(Math.random() * 3);
    var randomSize = Math.floor(Math.random() * 15);
    var randomColour = Math.floor(Math.random() * 6);

    elements[0].style.border = randomSize + 'px ' +
      borderArray[randomBorder] + ' ' + colourArray[randomColour];


function myLoadEvent() {
    var elements = document.getElementsByTagName('a');
    elements[0].addEventListener('click', linkClick);
}

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



```
Uncaught SyntaxError: Unexpected string
>
```

Common error #2

- The browser will give you the line number:



```
Uncaught SyntaxError: Unexpected string                ex1.js:9
```

Line number

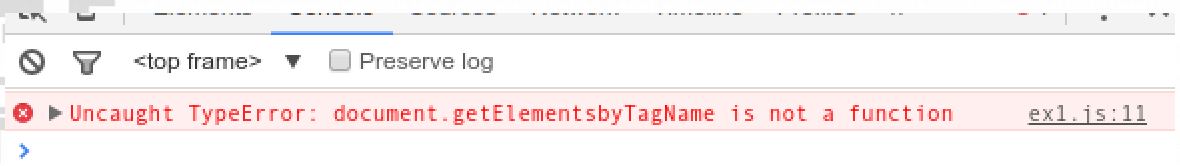
- Take a look at the line and carefully examine it:

```
colourArray[0] = 'red';
colourArray[1] = 'blue';
colourArray[2] 'green';
colourArray[3] = 'yellow';
colourArray[4] = 'white';
colourArray[5] = 'black';
```

In this case the error is caused by a missing = sign

Common error #3

- Typos
- Javascript is case sensitive
- It's very easy to get an error like this:



```
Uncaught TypeError: document.getElementsbyTagName is not a function    ex1.js:11
```

- This is because of the line:

```
    var elements = document.getElementsbyTagName('div');
```

- getElementsByTagName should have an uppercase B!

Common error #4

- = and ==. This won't produce an error in the console
- When using an if statement and comparing two values, you must used the == operator instead of =
- This will assign the randomNumber variable the value 1

```
if (randomNumber = 1) {
}
```

- Instead you should use == for comparison

```
if (randomNumber == 1) {
}
```

Document Object Model (DOM)

- HTML files are processed with javascript using the DOM (Document Object Model)
- This is a standardised way of accessing information from XML and HTML documents
- Each element can contain other elements or have properties (attributes)

DOM

- There are two main types of *node* in a HTML document:
  - Element: <p> tags, <div> tags, <h2> tags, etc that represent HTML elements
  - Text nodes: These store the text inside the element
- Elements can contain other elements
- Text nodes only contain *text*

DOM

- Most HTML documents contain a mix of text nodes and elements

```
<!doctype html>
<html>
  <head>
    <title>Exercise 1</title>
    <script src="ex1.js"></script>
    <link rel="stylesheet"
                href="style.css" />
  </head>
  <body>
  <div>
  </div>

  <a>Click me</a>

  </body>

</html>
```

"click me" is a text node inside an <a> element

Text nodes

- Text nodes have a *nodeValue* attribute which stores the text that the node contains
- You have already used this when updating the content of a node:

```
var elements = document.getElementsByTagName('p');
elements[0].firstChild.nodeValue = 'paragraph text to set';
```

- To write content to a text node, you first have to select it.
- This is done using

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

```
<p>Paragraph text</p>
```

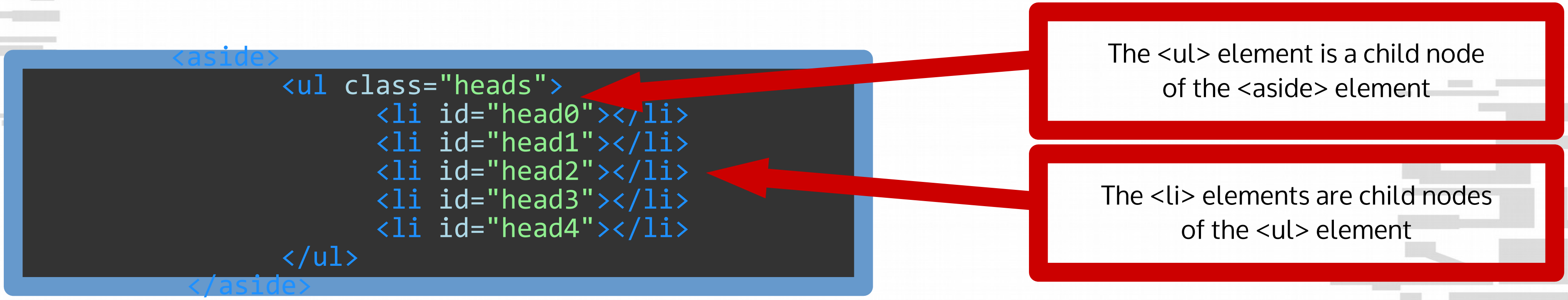
- element.firstChild selects the first child node inside the <p> element. In this case, the first child node is a text node
- The text node can then have its nodeValue set to update the content

```
var elements = document.getElementsByTagName('p');
elements[0].firstChild.nodeValue = 'paragraph text to set';
```

```
<p>paragraph text to set</p>
```

Child Nodes

- Every element has zero or more *child nodes*
- These are text nodes and elements that exist inside the selected element



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

The <ul> element is a child node of the <aside> element

The <li> elements are child nodes of the <ul> element

Child Nodes

- You can read an element's child nodes as an array using

```
    var elements = document.getElementsByTagName('ul');
    var childNodes =  elements[0].childNodes;
```

- Once you have the childNodes you can select a specific node using the array syntax:

```
childNodes[0].style.backgroundColor = 'green';
```

Child Nodes

- However, childNodes also contains the text nodes (even whitespace!)

```
var elements = document.getElementsByTagName('ul');
var childNodes =  elements[0].childNodes;
```

```
<aside>
     <ul class="heads">[ childNodes[0] - text]
       <li id="head0"></li>[ childNodes[2] - text ]
       <li id="head1"></li>[ childNodes[4] - text ]
       <li id="head2"></li>[ childNodes[6] - text ]
       <li id="head3"></li>[ childNodes[2] - text ]
       <li id="head4"></li>[ childNodes[8] - text ]
    </ul>
</aside>
```

ChildNodes

- Because of this, it's usually better to use getElementsByTagName instead of childNodes
- getElementsByTagName does not retrieve the text nodes, only the elements

```
var elements = document.getElementsByTagName('ul');
var childNodes =  elements[0].getElementsByTagName('li');
```

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

Other properties

- There are other properties available to find elements in the document
- Last week we looked at nextSibling
- This selects the next node (text or element) relative to the element it is called on:

```
var element = document.getElementById('head2');
var nextSibling =  element.nextSibling;
```

```
      <ul class="heads">
        <li id="head0"></li>
        <li id="head1"></li>
        <li id="head2"></li>----
--------<li id="head3"></li>
        <li id="head4"></li>
     </ul>
  </aside>
```

nextSibling

- You can chain nextSibling calls to select elements further down the document:

```
var element = document.getElementById('head2');
var nextSibling =  element.nextSibling.nextSibling;
```

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

previousSibling

- There is also previousSibling which looks up the document:

```
var element = document.getElementById('head2');
var previousSibling =  element.previousSibling.previousSibling;
```

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

parentNode

- There is also a parentNode property which selects the parent element of selected element
- This will always select an **element** and not a text node

```
var element = document.getElementById('head2');
var parentNode =  element.parentNode;
```

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

parentNode

- Like nextSibling, you can chain parentNode selectors to climb up the document:

```
var element = document.getElementById('head2');
var parentNode =  element.parentNode.parentNode;
```

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

DOM

- By combining:
  - getElementsByTagName
  - childNodes
  - parentNode
  - nextSibling/previousSibling
- It's possible to select any element or text node on any HTML page

Exercise 2

- 15 Minutes
- 1) Download Exercise2-Exercise from Github
- 2) **Without giving any elements IDs or class names**, when a <button> is pressed, randomly generate 6 lottery numbers between 1 and 49 for that block and display them in the table cells immediately below the button that was pressed
- 3) **Optional** Can you prevent the same number being drawn twice in a single draw?

Exercise 2 - Solution

```
function drawNumbers() {
        var tableCells = this.parentNode.getElementsByTagName('td');

        for (var i = 0; i < tableCells.length; i++) {
                var randomNumber = Math.ceil(Math.random() * 49);
                tableCells[i].firstChild.nodeValue = randomNumber;
        }
}

function myLoadEvent() {
        var buttons = document.getElementsByTagName('button');

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

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

Creating Elements

- Javascript can also create elements
- This is done using:

```
var element = document.createElement('tagName');
```

- E.g.

```
var element = document.createElement('div');
var element = document.createElement('p');
var element = document.createElement('h2');
```

Creating elements

- Once an element has been created you can treat it like any other element and:
  - Set CSS properties
  - Add event listeners
  - Add CSS Classes/IDs

```
var element = document.createElement('div');
element.style.width = '200px';
element.style.height = '200px';
element.style.backgroundColor = 'red';
```

Creating elements

- Once an element has been created it exists as a variable but is not visible on the page
- Before you can put the element on the page, you need to decide where to put it
- This is done by finding the element you want to add it to and then using the function *appendChild*

Adding elements to the page

- This code will create an element and add it to the <body> tag

```
var element = document.createElement('div');
element.style.width = '200px';
element.style.height = '200px';
element.style.backgroundColor = 'red';

var body = document.getElementsByTagName('body')[0];
body.appendChild(element);
```

Exercise 3

- 15 minutes
- 1) Create a basic HTML file with an empty body and a single button that says “Click me”
- 2) Adjust your code from Exercise 1 so that a random square is **added** to the page each time you click the button

Exercise 3 - Solution

```javascript
function buttonClick() {
        var colourArray = [];
        colourArray[0] = 'red';
        colourArray[1] = 'blue';
        colourArray[2] = 'green';
        colourArray[3] = 'yellow';
        colourArray[4] = 'cyan';
        colourArray[5] = 'black';

        var element = document.createElement('div');

        var randomNumber = Math.floor(Math.random() * 6);
        element.style.backgroundColor = colourArray[randomNumber];

        var borderArray = [];
        borderArray[0] = 'solid';
        borderArray[1] = 'dashed';
        borderArray[2] = 'dotted';

        var randomBorder = Math.floor(Math.random() * 3);
        var randomSize = Math.floor(Math.random() * 15);
        var randomColour = Math.floor(Math.random() * 6);

        element.style.border = randomSize + 'px ' + borderArray[randomBorder] +
                    ' ' + colourArray[randomColour];

        var body = document.getElementsByTagName('body')[0];
        body.appendChild(element);
}

function myLoadEvent() {
        var elements = document.getElementsByTagName('button');
        elements[0].addEventListener('click', buttonClick);
}

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

Creating Text Nodes

- To change the text of an element you use the code:

```
element.firstChild.nodeValue = 'text to display';
```

- This won't work with newly created elements because there is no *text node* to update
- element.firstChild doesn't actually exist
- When an element is created, it does not contain any text nodes

Creating Text Nodes

- To add text to a newly created element you can use the code

```
var textNode = document.createTextNode('text to create');
```

- Once the text node has been created, you can use *appendChild* to add it to the new element:

```
var element = document.createElement('p');
var textNode = document.createTextNode('text to create');
element.appendChild(textNode);
```

- Now when the element is added to the page it will contain the text "text to create"

Exercise 4

- 15 minutes
- 1) Take exercise 2 and remove the <table> elements from the HTML file
- 2) When the page loads no lottery numbers or balls will be shown. When one of the “Draw Lottery Numbers” buttons is pressed, use createElement() and createTextNode() to create the table with all 6 numbers
  - Hint: make sure the table is appended to the relevant <div> instead of the <body> tag

Exercise 3 - Solution

```
function drawNumbers() {

        var table = document.createElement('table');
        var tr = document.createElement('tr');
        table.appendChild(tr);

        for (var i = 0; i < 6; i++) {
                var randomNumber = Math.ceil(Math.random() * 49);
                var td = document.createElement('td');
                var textNode = document.createTextNode(randomNumber);
                td.appendChild(textNode);
                tr.appendChild(td);
        }

        this.parentNode.appendChild(table);
}

function myLoadEvent() {
        var buttons = document.getElementsByTagName('button');

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

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

Removing elements

- Along with adding elements to the page, you can also remove them
- This is done using the code:
-
```
parentElement.removeChild(elementToBeRemoved);
```

Remove Child

- Given the following HTML

```
<!doctype html>
<html>
	<head>
		<title>Example page</title>
	</head>
	<body>

		<h1>Example page</h1>

		<p>Some example text</p>
	</body>
</html>
```

- The <p> element can be removed with the code:

```
var body = document.getElementsByTagName('body')[0];
var paragraph = document.getElementsByTagName('p')[0];
body.removeChild(paragraph);
```

Removing elements

- This involves finding two elements:
  - The element that contains the element you want to remove
  - The element you want to remove
- This can be extra work
- It's possible to do this by utilising parentNode and only finding the element you want to remove:

```
var paragraph = document.getElementsByTagName('p')[0];
paragraph.parentNode.removeChild(paragraph);
```

Exercise 5

- 10 minutes
- 1) Extend exercise 3 to include a button that says "Clear" for each <div>
- 2) When the clear button is pressed, any tables inside the <div> should be removed
  - Hint: getElementsByTagName returns a live snapshot, when elements are removed from the page they are also removed from the array!

Exercise 5 - Solution

```
function drawNumbers() {

        var table = document.createElement('table');
        var tr = document.createElement('tr');
        table.appendChild(tr);

        for (var i = 0; i < 6; i++) {
                var randomNumber = Math.ceil(Math.random() * 49);
                var td = document.createElement('td');
                var textNode = document.createTextNode(randomNumber);
                td.appendChild(textNode);
                tr.appendChild(td);
        }

        this.parentNode.appendChild(table);
}

function clear() {
        var tables = this.parentNode.getElementsByTagName('table');
        for (var i = 0; i < tables.length; i++) {
                tables[i].parentNode.removeChild(tables[i]);
                i--;
        }
}

function myLoadEvent() {
        var divs = document.getElementsByTagName('div');

        for (var i = 0; i < divs.length; i++) {
                var buttons = divs[i].getElementsByTagName('button');
                buttons[0].addEventListener('click', drawNumbers);
                buttons[1].addEventListener('click', clear);
        }
}

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

Exercise 6

- Optional – Until the end of the session
- Using timers, generate the numbers one at a time
- Like the dice 2 weeks ago show various numbers until one is settled on and draw the numbers one after another