CSY1018
Web Development

Tom Butler
thomas.butler@northampton.ac.uk

Topic 8

- Function arguments
- Return values

Exercise 1

- 20 minutes
- 1) Create a HTML page with a single <button> element and an empty
- 2) When the button is clicked, generate a random number from 1-100 and add it to the <ul> element as a
  - Each time the button is pressed a new number should be added to the page
- 3) Use an array to keep track of which numbers have already been picked
  - Hint: use array.push
- 4) When a random number is picked, keep looping using a while loop until a new number is picked
  - The same number should not appear on the page more than once!

```
var pickedNumbers = [];


function buttonClick() {

        var pickAgain = true;

        while (pickAgain == true) {
                var randomNumber = Math.ceil(Math.random() * 100);

                var alreadyPicked = false;

                for (var i = 0; i < pickedNumbers.length; i++) {
                        if (pickedNumbers[i] == randomNumber) {
                                alreadyPicked = true;
                        }
                }

                if (alreadyPicked == true) {
                        pickAgain = true;
                }
                else {
                        pickAgain = false;
                }
        }

        pickedNumbers.push(randomNumber);

        var ul = document.getElementsByTagName('ul')[0];
        var li = document.createElement('li');
        var textNode = document.createTextNode(randomNumber);

        ul.appendChild(li);
        li.appendChild(textNode);
}

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

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

Whether to pick another number

Keep looping until a new number is picked

Loop through the list of numbers that have already been picked

If it's already in the array set alreadyPicked to true

If it's already been picked, pick another number

Add the newly picked number to the list of numbers that have already been picked

Functions

- So far we've used functions to respond to *events*
  - Page load
  - Button clicks
  - Timers
- It's also possible to write a function and call it yourself

Functions

- Every function has a name
- You can explicitly call a function by using its name followed by brackets

```
function createDiv() {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode('test');

        body.appendChild(div);
        div.appendChild(textNode);
}

function buttonClick() {
      createDiv();
}


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

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

Functions

- Both of these pieces of code have an identical result

```
function createDiv() {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode('test');

        body.appendChild(div);
        div.appendChild(textNode);
}

function buttonClick() {
      createDiv();
}


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

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

```
function buttonClick() {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode('test');

        body.appendChild(div);
        div.appendChild(textNode);
}


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

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

Functions

- Functions can be used to reduce repeated code and break the code into smaller chunks
- By putting code inside functions it can be a lot more manageable and easier to read

Arguments

- When you call a function, the code is run
- It's also possible to send values to a function that is unique each time it is called
- These are called *arguments*
- You have already used *arguments* when using the in-built javascript functions

Arguments

- When getElementById is called, you send the function the *id* of the element you want to find
- When you use *alert* you send the function the string you want to print to the screen
- These are *arguments*

```
var element = document.getElementById('myelement');
var divs = document.getElementsByTagName('div');
alert('Hello');
```

Arguments

- You can create a function that takes *arguments* in this way
- This is done by putting a *variable name* inside the brackets when the function is defined:

```
function createDiv(text) {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode('test');

        body.appendChild(div);
        div.appendChild(textNode);
}
```

Function Arguments

- When the function is called, the *argument* must be provided (like with alert/getElementById/etc)

```
function createDiv(text) {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode('test');

        body.appendChild(div);
        div.appendChild(textNode);
}



createDiv('test 1');
```

When createDiv is run,
The variable *text*
will be set to `test 1`

Function Arguments

```
function createDiv(text) {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode('test');

        body.appendChild(div);
        div.appendChild(textNode);
}


createDiv('hello');
```

When createDiv is run,
The variable *text*
will be set to `hello`

```
function createDiv(text) {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode('test');

        body.appendChild(div);
        div.appendChild(textNode);
}



createDiv('1');
createDiv('2');
createDiv('3');
```

Functions can be called multiple times with different argument values

Function Arguments

- This function could be changed so a div was created with the text provided as the argument

```
function createDiv(text) {
        var body = document.getElementsByTagName('body')[0];
        var div = document.createElement('div');
        var textNode = document.createTextNode(text);

        body.appendChild(div);
        div.appendChild(textNode);
}



createDiv('1');
createDiv('2');
createDiv('3');
```

Function arguments

- You can create a function that takes multiple arguments by separating them with a comma

```
function createElementWithText(tag, text) {
        var body = document.getElementsByTagName('body')[0];
        var li = document.createElement(tag);
        var textNode = document.createTextNode(text);

        ul.appendChild(li);
        li.appendChild(textNode);
}



createElementWithText('p', ‘paragraph text’);
createElementWithText('div', ‘some text for the div’);
createElementWithText('h2', ‘heading text’);
```

Exercise 2

- 10 minutes
- 1) Amend Exercise 1 to have a function called *createLi*
  - This function should have an argument which is the text which will be displayed in the

```
var pickedNumbers = [];

function createLi(text) {
        var ul = document.getElementsByTagName('ul')[0];
        var li = document.createElement('li');
        var textNode = document.createTextNode(text);

        ul.appendChild(li);
        li.appendChild(textNode);
}

function buttonClick() {
        var pickAgain = true;

        while (pickAgain == true) {
                var randomNumber = Math.ceil(Math.random() * 100);

                var alreadyPicked = false;

                for (var i = 0; i < pickedNumbers.length; i++) {
                        if (pickedNumbers[i] == randomNumber) {
                                alreadyPicked = true;
                        }
                }

                if (alreadyPicked == true) {
                        pickAgain = true;
                }
                else {
                        pickAgain = false;
                }
        }

        pickedNumbers.push(randomNumber);
        createLi(randomNumber);
}

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

Return values

- As well as *arguments*, functions can *return* a value back to the place they were called
- You have already seen this behaviour with the inbuilt functions:

```
var element = document.getElementById('myelement');
var divs = document.getElementsByTagName('div');
var randomNumber = Math.random();
```

Return values

- Each of these functions performs a task and then sends a value back to where it was called

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

- The random function generates a random number then sends that number back to the place it was called
- The value sent back (the "return value") can then be stored inside a variable
- The same thing happens with other functions

```
var element = document.getElementById('myelement');
var divs = document.getElementsByTagName('div');
```

Return Values

- Your own functions can also return values
- This can be use to reduce repeated code and simplify the calling code
- To return a value use the *return* keyword followed by a value which will be returned

```
function randomNumber1to10() {
        var randomNumber = Math.ceil(Math.random() * 10);
        return randomNumber;
}


var random1 = randomNumber1to10();
var random2 = randomNumber1to10();
var random3 = randomNumber1to10();
```

Return Values

- Return values can be combined with arguments e.g.:

```
function randomNumber(max) {
        var randomNumber = Math.ceil(Math.random() * max);
        return randomNumber;
}


//Generate a random number between 1 and 5
var random1 = randomNumber(5);

//Generate a random number between 1 and 3
var random2 = randomNumber(3);

//Generate a random number between 1 and 100
var random3 = randomNumber(100);
```

Exercise 3

- 15 minutes
- 1) Create a function called **arrayContains** that takes two arguments:
  - An array
  - A value
- The function should return true or false if the value exists in the array
- 2) Test the function works with the code below

```
var array1 = [];
array1[0] = 5;
array1[1] = 7;
array1[1] = 2;
array1[1] = 6;


//should alert “true” because the number 2 exists in the array
var result1 = arrayContains(array1, 2);
alert(result1);

//should alert “false” because the number 1 does not exist in the array
var result2 = arrayContains(array1, 2);
alert(result2);
```

Exercise 3 - Solution

```
function arrayContains(array, value) {
        var found = false;

        for (var i = 0; i < array.length; i++) {
                if (array[i] == value) {
                        found = true;
                }
        }

        return found;
}

var array1 = [];
array1[0] = 5;
array1[1] = 7;
array1[1] = 2;
array1[1] = 6;


//should alert “true” because the number 2 exists in the array
var result1 = arrayContains(array1, 2);
alert(result1);

//should alert “false” because the number 1 does not exist in the array
var result2 = arrayContains(array1, 2);
alert(result2);
```

Exercise 4

- 15 Minutes
- 1) Create a function called newRandom()
  - It should take two arguments:
    - An array and generate a random number that isn't already in the array
    - A maximum value to generate and be usable like this:
  - And return a random number that does not already exist in the given array between 0 and the max argument
    - Hint: Use your existing arrayContians function from exercise 3

```
var array1 = [];
array1[0] = 5;
array1[1] = 7;
array1[1] = 2;
array1[1] = 6;


//should alert a random number between 0 and 9 but excluding 5, 7, 2 and 6
var result1 = newRandom(array1, 10);
alert(result1);
```

Exercise 4 - Solution

```
function newRandom(array, max) {
        var newNumber = false;

        while (newNumber == false) {
           var randomNumber = Math.floor(Math.random() * max);

           var alreadyPicked = arrayContains(array, randomNumber);

           if (alreadyPicked == false) {
              newNumber = true;
           }
        }

        return randomNumber;
}

var array1 = [];
array1[0] = 5;
array1[1] = 7;
array1[1] = 2;
array1[1] = 6;


//should alert a random number between 0 and 9 but excluding 5, 7, 2 and 6
var result1 = newRandom(array1, 10);
alert(result1);
```

Exercise 5

- Update exercise 2 to use the *newRandom* function

```
var pickedNumbers = [];

function createLi(text) {
        var ul = document.getElementsByTagName('ul')[0];
        var li = document.createElement('li');
        var textNode = document.createTextNode(text);

        ul.appendChild(li);
        li.appendChild(textNode);
}

function newRandom(array, max) {
        var newNumber = false;

        while (newNumber == false) {
           var randomNumber = Math.floor(Math.random() * max);

           var alreadyPicked = arrayContains(array, randomNumber);

           if (alreadyPicked == false) {
              newNumber = true;
           }
         }

        return randomNumber;
}

function buttonClick() {
        var randomNumber = newRandom(pickedNumbers, 100);

        pickedNumbers.push(randomNumber);

        createLi(randomNumber);
}

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

```
var pickedNumbers = [];

function createLi(text) {
        var ul = document.getElementsByTagName('ul')[0];
        var li = document.createElement('li');
        var textNode = document.createTextNode(text);

        ul.appendChild(li);
        li.appendChild(textNode);
}

function newRandom(array, value) {
        var newNumber = false;

        while (newNumber == false) {
           var randomNumber = Math.floor(Math.random() * max);

           var alreadyPicked = arrayContains(array, randomNumber

           if (alreadyPicked == false) {
              newNumber = true;
           }
        }

        return randomNumber;
}

function buttonClick() {
        var randomNumber = newRandom(pickedNumbers, 100);

        pickedNumbers.push(randomNumber);

        createLi(randomNumber);
}

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

```
var pickedNumbers = [];

function buttonClick() {
    var pickAgain = true;

     while (pickAgain == true) {
        var randomNumber = Math.ceil(Math.random() * 100);
        var alreadyPicked = false;

        for (var i = 0; i < pickedNumbers.length; i++) {
                if (pickedNumbers[i] == randomNumber) {
                        alreadyPicked = true;
                }
        }

        if (alreadyPicked == true) {
                pickAgain = true;
        }
        else {
                pickAgain = false;
        }
    }

    pickedNumbers.push(randomNumber);

    var ul = document.getElementsByTagName('ul')[0];
    var li = document.createElement('li');
    var textNode = document.createTextNode(randomNumber);

    ul.appendChild(li);
    li.appendChild(textNode);
}

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

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

Exercise 6

- Modify the game from topic 6. Add a function called
  - isColliding(element1, element2)
- That returns true or false depending on whether element1 is overlapping element2
- Use the function to prevent the player walking through the tree or the opponent character