**ES6**

**BASICS**

**Let & Const declarations**

*Let* is set to replace *var*\*\*\*\*

On the global scope; *var* and *let* are pretty much the same. The *let* statement declares a block scope local variable. *Let* allows you to declare variables that are limited to scope to the block, statement or expression on which it is used.

*Let* example:

//using var

Function test() {

Var a = 30;

If(true) {

Var a = 50;

Console.log(a);

}

Console.log(a);

}

Test(); //50, 50

//using let

Function test() {

Var a = 30;

If(true) {

Var a = 50;

Console.log(a);

}

Console.log(a);

}

Test(); //50, 30

*Const*  = Constant. i.e. it cannot be changed once it is initially set.

Const name = “Ali”;

Name = “Lauren”; //error!!

**Classes & inheritance**

Class User{

Constructor(username, email, password){

This.username = username;

This.email = email;

This.password = password;

}

Register() {

Console.log(‘Hello ‘+this.username);

}

}

Constructors are methods that will run when your object is created.

Let ali = new User(‘Rezaa91’, ‘rezaa91@hotmail.co.uk’,123);

Ali.register(); //runs register method in User class

//you can also create static methods within class:

Static countUsers() {

Console.log(‘there are 50 users’);

}

//to call – nameOfClass.nameOfStaticMethod();

User.countUsers();

You can also inherit/extend classes

Class Member extends User{

Constructor(username, email, password, memberPackage) {

Super(username, email, password); //this does same as this.username = username etc…

This.memberPackage = memberPackage;

}

getPackage() {

console.log(this.username + ‘ is on the ‘ + this.memberPackage +’ package.’);

}

}

Let lauren = new Member(‘Lauren’,’example@example.com’, 123, ‘standard’);

//lauren will now have access to methods from the User and Member class.

**Template literals**

In order to put html or text over multiple lines for readability, you can use backticks (`), instead of concatenating each line (+).

Let template = `<h3>Hello World</h3>

<p>This is a paragraph</p>

<p>Nice to meet you ${printName()}</p>`

Document.getElementById(‘template’).innerHTML = template;

**New string and number methods**

startsWith(); //returns true or false, depending on argument

endsWith(); //same as above but at end of string

includes(); //returns true or false if argument is in the string

isFinite(); //number.isFinite(3); = true

isNaN(); //checks if not a number

isInteger(); //checks if integer

**Default parameters and spread operator**

Function greet(greeting = “Hello World”){ //set default param

Console.log(greeting);

}

Greet(); //’Hello World’

Greet(‘Howdy’); //’Howdy’

Spread syntax allows an iterable such as an array expression to be expanded in places where zero or more arguments are expected, or an object expression to be expanded in places where zero or more key-value pairs are expected.

e.g.1

let arr = [1,2,3];

function spreadTest(one, two, three){

console.log(two);

}

spreadTest(…arr); //prints 2 – basically spreads array out

e.g.2

let arrOne = [‘thing’,’otherThing’];

let arrTwo = [‘one’, …arrOne, true]; //’one’, ‘thing’, ‘otherThing’, true.

//if no spread operator e.g. [‘one’,arrOne, true]; => ‘one’, [‘thing’, ‘otherThing’], true;

e.g. 3 \***important in react**\*

let person = {

name: “Ali”,

surname: “Issaee”

}

Let address = {

No: 150,

Street: ‘Grosvenor rd’

}

Let whole = {…person, …address};

Console.log(whole); => {

Name:”Ali”,

Surname:”Issaee”,

No: 150,

Street: “Grosvenor rd”

}

**Arrow functions**

A shorted function syntax

*This* behaves differently

Function Greet(x) {

This.x = x;

}

Greet.prototype.greetName = function(arr) {

Return arr.map((y) => {

Console.log(this.x + y);

});

}

//as opposed to:

Greet.prototype.greetName = function(arr) {

Var that = this;

Return arr.map(function(y) {

Console.log(that.x + y);

});

}

**Test**

1. What is the difference between *let* and *var* – give example
2. What is a *const* – give example
3. Create a class with a basic method
4. Create another class which inherits the class made in Q4
5. Create a static method in a class and run it
6. What does the constructor method do?
7. How can you print html/text over multiple rows without having to concatenate?
8. Give example of above print functions and variables also in the template
9. Give examples of 3 new string and number functions
10. How do you set default parameters in functions?
11. What does the spread operator do?
12. Give examples of spread operator – in functions, arrays and objects
13. Give examples of arrow functions using parameters and the event object.
14. How does *this* differ between normal functions and arrow functions?