**1.**

* How many **argument values** do you see here and what are they?
* How many **parameter variables** do you see here and what are they?

function logMessage(message) { console.log(message); } logMessage('Welcome!');

**2.**

What will be printed in a console and **why**?

function modifyMessage(message) { message = 'Wow!'; } var message = 'Ok'; modifyMessage(message); console.log(message); // ???

* JavaScript is **pass-by-value**. Which means each argument is **copied** into the parameter variable.

**3.**

What will be printed in a console and **why**?

function getExcited(excitement) { excitement.all = 'Wow!'; } var excitement = {}; excitement.all = ['new', 'and', 'fun']; getExcited(excitement); console.log(excitement.all); // ???

**4.**

Find and fix 2 errors:

var car = { started: false, start: function () { started = true; }, stop: function () { started = false; } }; car.start();

**5.**

What's the benefit of using start() and stop() methods from the previous example instead of directly changingstarted value?

var car = { started: false }; car.start = true;

**6.**

What's the result and **why**?

'1' == true;

1 === true;

0 == '';

0/0;

var message; message === undefined;

undefined !== null;

undefined != null;

typeof undefined > typeof null;

-true;

-true-false;

-false - -false + false;

**Falsey values in JavaScript**

* undefined
* null
* 0
* ''
* NaN

Everything else is **truthy**.

NaN is a number that can't be represented in JavaScript.

**7.**

What will be logged and **why**?

var car = 'Tesla' + ' ' + 'Model' + ' ' + 1 + 2; console.log(car);

**8.**

What will be logged and **why**?

var car = 'Tesla' + ' ' + 'Model' + ' ' + (1 + 2); console.log(car);

**9.**

What will be logged and **why**?

function counter() { var value = 0; function increment() { value = value + 1; } function getValue() { return value; } return { increment: increment, getValue: getValue }; } var theCounter = counter(); theCounter.increment(); console.log(theCounter.getValue()); // ??? theCounter.increment(); console.log(theCounter.getValue()); // ??? theCounter.increment(); console.log(theCounter.getValue()); // ???

**Closures** are a function along with a referencing environment.

## 1.

* How many **argument values** do you see here and what are they?
* How many **parameter variables** do you see here and what are they?

function logMessage(message) { console.log(message); } logMessage('Welcome!');

1 argument value – ‘Welcome’

1 parameter value – message.

## 2.

What will be printed in a console and **why**?

function modifyMessage(message)

{ message = 'Wow!'; }

var message = 'Ok';

modifyMessage(message);

console.log(message); // ???

OK! Value of message at Global level is still Ok.

If modifyMessage did not have parameter then it will print Wow!

\*\* try this one with multiple values.

* JavaScript is **pass-by-value**. Which means each argument is **copied** into the parameter variable.

## 3.

What will be printed in a console and **why**?

function getExcited(excitement)

{ excitement.all = 'Wow!'; }

var excitement = {};

excitement.all = ['new', 'and', 'fun'];

getExcited(excitement);

console.log(excitement.all); // ???

Wow! Is returned.

## 4.

Find and fix 2 errors:

var car = { started: false,

start: function () { started = true; },

stop: function () { started = false; } }; car.start();

console.log(car.started);

car is an object so the property needs to be in [‘start’] and also within the variable needs a reference on each key, to say this.start and this.stop.

var motorbike {isStarted:false}

car.start();

can also be written as car.start.call(car);

to start motorbike, you can access function within the variable by saying:

car.start.call(motorbike);

## 5.

What's the benefit of using start() and stop() methods from the previous example instead of directly changing started value?

var car = { started: false };

car.start = true;

Easier maintenance. Use functions wherever you can.

## 6.

What's the result and **why**?

'1' == true; (loose comparison, true gives 1, so expression is true)

1 === true; (strict comparison – so expression false)

0 == ''; (loose comparison so falsey value of ‘ ‘ means true)

0/0; (NaN )

var message; message === undefined; true (undefined vs undefined)

undefined !== null; null is empty object – evaluates to true (as thye don’t match).

undefined != null; loose comparison – values match - false

typeof undefined > typeof null; this evaluates to ‘undefined’>’object’

which then compares the first characters of each string, which evaluates to true.

-true; evaluates to -1

-true-false; -1

-false - -false + false; evaluates to 0.

### Falsey values in JavaScript enables them to be compared.

* undefined
* null
* 0
* ''
* NaN

Everything else is **truthy**.

NaN is a number that can't be represented in JavaScript.

## 7.

What will be logged and **why**?

var car = 'Tesla' + ' ' + 'Model' + ' ' + 1 + 2; console.log(car);

Evaluates from left to right – makes Tesla Model12

## 8.

What will be logged and **why**?

var car = 'Tesla' + ' ' + 'Model' + ' ' + (1 + 2); console.log(car);

Now has brackets so will make it to Tesla Model3

## 9.

What will be logged and **why**?

function counter() {

var value = 0;

function increment() {

value = value + 1; }

function getValue() {

return value; }

return { increment: increment, getValue: getValue }; }

var theCounter = counter(); //returns object with increment&getvalue environments.

theCounter.increment();

console.log(theCounter.getValue());

theCounter.increment();

console.log(theCounter.getValue());

theCounter.increment();

console.log(theCounter.getValue());

counter() - returns an object

**Closures** are a function along with a referencing environment.