**1.**

What will be logged and **why**?

String.prototype.repeatify = function repeat(number) {

var answer = this + this;

if (number > 0) { return String.prototype.repeatify(number - 1); } return answer; }; console.log('welcome'.repeatify(3)); // ???

Loop will stop when number =1 – returns empty string.

**2.**

Write a function that converts each character of an array into uppercase.

['w','e','l','c','o','m','e'].uppercase(); // ['W', 'E', 'L', 'C', 'O', 'M', 'E']

.toUpperCase only works on string.

['w','e','l','c','o','m','e'].toString().toUpperCase.split();

Can chain methods as each returns a string which can then be processed.

**3.**

Write a function that returns reversed string.

'message'.reverse(); // 'egassem'

function reverse(s) { return s.split('').reverse().join(''); }

**4.**

Does JavaScript pass parameter by value or by reference?

Pass by value.

~But objects are passed by (a copy of) the reference to the object, therefore looking like a copy of the object. SO big difference when passing a string vs passing an object.

**5.**

Explain Function.prototype.bind.

Example often used.

Write a function add, where:

Var add5 = add(5);

Console.lg(add5(10); that gives answer 15.

Function add (number1){

Return function inner(number2){

Return number1 + number2}

}

But now you want a solution that uses bind (as function calls are expensive.).

NOTE: Bind does not call a function. Bind will create a new function and return it.

Eg. Function a()

{console.log(“I am A”)}

a.apply = I am A

a.call=I am A

a.bind = function

function add(number1){

//case 1: return sum

//case2: return function

if (arguments.length===2)

return arguments[0]+arguments[1];

} // otherwise

return add.bind(null, arguments[0];

}

var add5 = add(5);

console.log(add5(10));

**6.**

What will be logged and **why**?

console.log(!!(obj1 && obj2));

True –as the operator turns it into a Boolean value.

!!(null ||null) = !!null = !true = false.

**7.**

Write a one-line piece of JavaScript code that concatenates all strings passed into a function.

concatenate('Welcome', ' ', 'to', ' ', 'The', ' ', Iron’', ' ', 'Yard'); // 'Welcome to The Iron Yard'

Array.prototype.join.call(arguments,'');

**8.**

What will be logged and **why**?

if (!('message' in window)) is property message in Window? Message is declared but not assigned a value, therefore returns undefined.

{ var message = 'Hello'; }

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

**9.**

What will be logged and **why**?

function message(text)

{ return text + '!'; }

var message;

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

The var message is undefined (its there but has no value) so it will look for other value of message and returns the whole function.

Function declaration will overwrite variable declaration.

**10.**

What will be logged and **why**?

function log()

{ console.log(this); // ??? }

log.apply(null);

Will go back to default value – therefore return Window