**Frontend Advanced**

**Module II – Challenges**

**Arrow Function Exercises**

Exercise 1: Write an arrow function that returns the string 'Hello World!'.

Exercise 2: Write an arrow function that expects an array of integers, and returns the sum of the elements of the array. Use the built-in method reduce on the array argument.

Exercise 3: Rewrite the following code by using arrow functions wherever it makes sense to use them:

var Entity = function( name, delay ) {

this.name = name;

this.delay = delay;

};

Entity.prototype.greet = function() {

setTimeout( function() {

console.log( 'Hi, I am ' + this.name );

}.bind( this ), this.delay );

};

var java = new Entity('Java', 5000 );

var cpp = new Entity('C++', 30 );

java.greet();

cpp.greet();

Exercise 4. Write a function that executes a callback function after a given delay in milliseconds. The default value of delay is one second.

**Classes Exercise**

Exercise 1. Create a PlayerCharacter and a NonPlayerCharacter with a common ancestor Character. The characters are located in a 10x10 game field. All characters appear at a random location. Create the three classes, and make sure you can query where each character is.

Exercise 2. Each character has a direction (up, down, left, right). Playercharacters initially go right, and their direction can be changed using the faceUp, faceDown, faceLeft, faceRight methods. Non-player characters move randomly. A move is automatically taken every 5 seconds in real time. Right after the synchronized moves, each character console log sits position. The player character can only influence the direction he is facing. When a player meets a non-player character, the non-player character is eliminated from the game, and the player’s score is increased by 1.

Exercise 3. Make sure the Character class cannot be instantiated.

**Destructuring Exercise**

Exercise 1. Swap two variables using one destructuring assignment.

Exercise 2. Create one destructuring expression that declares exactly one variable to retrieve x.A[2]. let x = { A: [ 't', 'e', 's', 't' ] };

Exercise 3. Make a shallow copy of an array of any length in one destructuring assignment! If you don’t know what a shallow copy is, make sure you read about it, as you will need these concepts during your programming career.

**Exercises - Guess the Value of This**

Question 1

var x = function() {

console.log(this);

};

x();

Question 2

var x = {

y: function() {

console.log(this);

}

}

x.y();

Question 3

var x = function() {

console.log(this);

}

var y = {

x: x

}

y.x();

Question 4

var x = {

y: function() {

console.log(this);

}

};

var a = {

b:x.y

};

a.b();

Question 5

(function() {

var x = function() {

console.log(this);

}

x();

})();

Question 6

(function() {

var x = function() {

console.log(this);

}

y = {};

y.x = x;

y.x();

})();

Question 7

(function() {

var x = function() {

console.log(this);

};

x.apply({a:'b'});

})();

Question 8

(function() {

var Cat = function() {

console.log(this);

};

var x = new Cat();

})();

**Exercise - HIJAX**

Overriding the function of hyperlinks is a useful thing to be able to do. The hyperlink will work normally in the absence of JavaScript but will AJAX in the content if it is able. This technique is sometimes rather cleverly called HIJAX.

Add a list of hyperlinks to a page. When a hyperlink is clicked, detect that event, and call a function that retrieves the content via AJAX and inserts it into an element on the page.