**Section 2 - JS Refresher**

**L-12**

* var 🡪 Obsolete, not used anymore.
* let 🡪 a variable name.
* const 🡪 a constant datatype, which can not be changed.

**L-13 Arrow functions**

Different syntax for normal JS functions.

function myFunc(…) { …. ….. …. } vs const myFunc = () => { …. …. …. }

* We can omit the parentheses if there is a single argument in arrow function.
* When there is only one return statement in arrow function we can write it as -  
  const twiceNum = number => number\*2

**L-14 Exports and Imports (Modules)**

* Separation of concerns in different modules.
* If there is a default export, then we do not need to specify what we are exporting in another file.
* But if there is a non-default export of multiple things from a single module, then we make named-exports.  
  **file1.js**   
  const person = { name: Max;} export default person  
    
  vs  
  **file2.js**  
    
  export const a = () => {……}  
  export const b = 10;
* **importingFile.js**import person from ‘/file1.js’   
  import prs from ‘./file1.js’   
  both the above statements work fine.  
    
  import {a} from ‘./file2.js’  
  import {b} from ‘./file2.js’  
  or import \* as bundled from ‘./file2.js’  
    
  we need the name of exact names to make it run the specific import.

**L-15,16 Classes in JS**

* Classes have properties and methods.
* We will use constructor functions.
* We will make use of inheritance, with extends keyword.
* We must use super constructor in a child class to correctly initialise the parent class by calling it’s constructor, without explicitly creating an object of the parent class.
* ES7 syntax / ES6-Babel 🡨 Next generation JS  
  **Properties:** myprop =”value”; *and not this.myprop =”value”;***Functions:** mymethod = () => { …} *and not myMethod() { …. } the new method allows us to avoid confusion due to ‘this’ keyword.*So with the new method we do not need to call the constructor.

**L-17 Spread and Rest Operators 🡪 …**

* The spread operator is used to split up array elements or object properties.  
  **eg:** const newArray = […oldArray, 1, 2];  
   const newObj = {…0ldObj, newProp:5}  
  This allows us to pull out an old object elements and to add to a new object with additional info.
* Rest operator is usd to merge a list of function arguments into an array  
  **eg:** function sortArgs(…many\_arg) { return many\_arg.sort()}   
  This allows us to merge all arguments being received into an array.

**L-18 Destructuring**

* Easily extract array elements or object properties and store them in variables.
* It allows to extract single elements.   
  const numbers = [1,2,3];  
  [num1, num2] = numbers;  
  it will only store number 1 and 2.  
  [a,b] = [‘Hello’,’Max’];

**L-19 Reference and Primitive Types Refresher**

Const number = 1;  
Const num2 = number;  
*This is to copy the real value and separately store num2 from 1.*

* Strings, Boolean and number are primitive data types.
* Objects and arrays are reference type of objects, instead of this, we have pointers to objects instead of copy.
* Const person = {name: “Max”};  
  const secondPerson = person;  
  person.name = “manu”;  
  console.log(secondPerson); 🡪 Manu.  
  Even though we did not manipulate second person, it got changed.
* Immutable copying – spread operator is used for deep copy. Thee spread operator will pull out the properties and store it in new object.  
  const secondPerson = { …person };  
  *this will create a deep copy.*

**L-20 Refreshing array Methods**

1. map() 🡪 goes through each unit of an array.