NAME:IHEANACHO DAVID IZUNNA
MATRIC NO: 23/1512
DEPARTMENT: INFORMATION TECHNOLOGY (GROUP B)
COURSE: COS 209

Why Should I Learn ES6

Learning ES6 is important for several reasons:

1. Modern Features: ES6 introduces many new features that make JavaScript programming easier and more efficient, such as arrow functions, classes, template literals, and destructuring.

2. improved Syntax: The syntax improvements in ES6 help you write cleaner and more readable code, which can make it easier to maintain and understand.

3. Enhanced Functionality: With features like Promises and modules, ES6 allows for better asynchronous programming and modular code organization, which are essential for building modern web applications.

4. Industry Standard: Many frameworks and libraries, like React and Angular, use ES6 features. Knowing ES6 will help you work more effectively with these tools.

5. Future-Proofing: As JavaScript continues to evolve, understanding ES6 will give you a strong foundation for learning future versions of the language.

Overall, learning ES6 will not only improve your coding skills but also make you more competitive in the job market.

Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

- Using var
- Using let
- Using const

VAR
>The var keyword was used in all JavaScript code from 1995 to 2015.
>The var keyword should only be used in code written for older browsers.
LET
>Only use let if you can't use const
>Unlike variables declared with var, which are function-scoped, let allows for declaring block-scoped variables. This means a variable declared with let is limited to the block, statement, or expression where it is used.
CONST
> Always use const if the value should not be changed
> Always use const if the type should not be changed (Arrays and Objects)

Arrow Function

Arrow functions allow us to write shorter function syntax

Arrow function syntax:

```
const greetings = name => {
  console.log(`Hello, ${name}!`);
};
greetings('John'); // Hello, John!
```

JavaScript Classes

>JavaScript Classes are templates for JavaScript Objects.

JavaScript Class Syntax

>Use the keyword class to create a class.

>Always add a method named constructor().

SYNTAX

```
class ClassName {
  constructor() { ... }
}
```

JavaScript Array Methods

1.Array length

>The length property returns the length (size) of an array:

2.Array toString()

>The JavaScript method toString() converts an array to a string of (comma separated) array values.

3. Array join()

The join() method also joins all array elements into a string.

It behaves just like toString(), but in addition you can specify the separator:

4.Array pop()

>The pop() method removes the last element from an array

5.Array push()

>The push() method adds a new element to an array (at the end):

6.Array shift()

>The shift() method removes the first array element and "shifts" all other elements to a lower index.

>The shift() method returns the value that was "shifted out":

7.Array unshift()

>The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:

>The unshift() method returns the new array length:

8.Array delete()

>Using delete() leaves undefined holes in the array.

>Use pop() or shift() instead.

9.Array concat()

>The concat() method creates a new array by merging (concatenating) existing arrays:

>The concat() method does not change the existing arrays. It always returns a new array.

>The concat() method can take any number of array arguments.

10.Array copyWithin()

>The copyWithin() method copies array elements to another position in an array:

>The copyWithin() method overwrites the existing values.

>The copyWithin() method does not add items to the array.

>The copyWithin() method does not change the length of the array.

11.Array flat()

≥The flat() method creates a new array with sub-array elements concatenated to a specified depth.

12.Array flatMap()

>The flatMap() method first maps all elements of an array and then creates a new array by flattening the array.

13.Array splice()

>The splice() method can be used to add new items to an array:

14.Array toSpliced()

>The difference between the new **toSpliced()** method and the old **splice()** method is that the new method creates a new array, keeping the original array unchanged, while the old method altered the original array.

15. Array slice()

>The slice() method slices out a piece of an array into a new array:

>The slice() method creates a new array.

>The slice() method does not remove any elements from the source array.

Destructuring

Destructuring Assignment Syntax

The destructuring assignment syntax unpack object properties into variables:

let {firstName, lastName} = person;

It can also unpack arrays and any other iterables:

let [firstName, lastName] = person;

Note:

Destructuring is not destructive.

Destructuring does not change the original object.

Modules

>JavaScript modules allow you to break up your code into separate files.

This makes it easier to maintain a code-base.

Modules are imported from external files with the import statement.

Modules also rely on type="module" in the <script> tag.

>Modules with **functions** or **variables** can be stored in any external file.

>There are two types of exports: **Named Exports** and **Default Exports**.

>Named Exports

Let us create a file named person.js, and fill it with the things we want to export.

You can create named exports two ways. In-line individually, or all at once at the bottom.

>Default Exports

Let us create another file, named message.js, and use it for demonstrating default export.

You can only have one default export in a file.

Ternary Operator

>The ternary operator is a simplified conditional operator like if / else.

>Syntax: condition ? <expression if true> : <expression if false>

Spread Operator

>The JavaScript spread operator (...) allows us to quickly copy all or part of an existing array or object into another array or object.