**Node JS Assignment**

# **Module – 1: Node - JavaScript Fundamental**

Q1. What is the difference between Java & JavaScript?

🡺 Java is an object-based programming language, JavaScript is an OOP(object-oriented program. JavaScript is more commonly used in web applications like browsers, while Java is more widely used in app development, smart devices and back-end applications.

Q2. What is javascript?

🡺 JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images.

Q3. What are the data types supported by JavaScript?

### 🡺 **JavaScript has 8 Datatypes**

1. String  
2. Number  
3. Bigint  
4. Boolean  
5. Undefined  
6. Null  
7. Symbol  
8. Object

### **The Object Datatype**

The object data type can contain:

1. An object  
2. An array  
3. A date

Q4. What are the scopes of a variable in JavaScript?

🡺 The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

* **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
* **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Q5. What is Callback?

🡺 A callback is a function passed as an argument to another function

This technique allows a function to call another function. A callback function can run after another function has finished.

Q6. What is Closure? Give an example

🡺 In JavaScript, a closure is a function that has access to variables from its outer (enclosing) scope, even after the outer function has finished executing. This means that the inner function "closes over" the variables of the outer function, allowing it to access and manipulate those variables.

Here's an example to illustrate the concept of closure in JavaScript:

function outerFunction() {

let outerVariable = 'I am from the outer function';

function innerFunction() {

console.log(outerVariable);

}

return innerFunction;

}

// Create a closure by calling outerFunction and assigning the result to a variable

let closureExample = outerFunction();

// Call the inner function, which still has access to outerVariable

closureExample(); // Output: I am from the outer function

Q7. What is the difference between the operators ‘==‘ & ‘===‘ ?

🡺 The ‘==’ (Equality Operation) operator checks for equality after performing types coercion if the operands are of different types.

It compares values after converting them to a common type.

Example:

'5' == 5; // true (string '5' is coerced to a number)

1 == true; // true (boolean true is coerced to a number)

🡺 The ‘===’ operator (String Equality Operation), also known as the strict equality operator, checks for equality without performing type coercion.

It requires both the value and the type to be the same for the comparison to be true.

Example:

'5' === 5; // false (string '5' is not equal to number 5 in both value and type)

1 === true; // false (number 1 is not equal to boolean true in type)

Q8. What is the difference between null & undefined?

🡺 **Null** : null is an assignment value that represents the intentional absence of any object value.

**Undefined** : A variable that has been declared but has not been assigned a value is undefined. Function parameters that are not provided with an argument receive the value undefined. Accessing an object property or array element with a non-existing key/index results in undefined.

Q9. What would be the result of 2+5+”3″

🡺 Ans : “73”

JavaScript converts the numeric result 7 into a string before concatenating it with the string "3".

Q10. What is the difference between Call & Apply?

🡺 **call method:**

The call method is used to invoke a function with a specified this value and individual arguments.

**Syntax** : functionName.call(thisValue, arg1, arg2, ...);

🡺 **apply method:**

The apply method is similar to call, but it takes arguments as an array or an array-like object.

**Syntax** : functionName.apply(thisValue, [arg1, arg2, ...]);

**Module - 2: Node - NodeJS – Introduction**

Q1. What is Node.js? Where can you use it?

🡺 Node.js is a server-side JavaScript runtime that enables the execution of JavaScript code outside a web browser.

It utilizes the V8 JavaScript engine and is known for its non-blocking, event-driven architecture.

Node.js is commonly used to build scalable and high-performance network applications, such as web servers and APIs.

It's versatile and finds applications in various domains, including web development, real-time applications, and microservices architecture.

Q2. Explain callback in Node.js.

🡺 In Node.js, a callback is a function passed as an argument to another function, which will be invoked at a later time, often upon completion of an asynchronous operation.

Eaxmple:

const fs = require('fs');

// Asynchronous file read operation with a callback

fs.readFile('example.txt', 'utf8', function (err, data) {

if (err) {

console.error('Error reading file: ', err);

} else {

console.log('File content:', data);

}

});

console.log('Reading file asynchronously...');

Q3. What are the advantages of using promises instead of callbacks?

🡺 Promises provide a cleaner and more readable syntax for handling asynchronous code compared to nested callbacks, making the code structure more linear and easier to understand.

Promises simplify error management through the use of the '.catch()' method, allowing centralized error handling and avoiding the need to scatter error-handling logic throughout the code.

Promises support method chaining with '.then()', facilitating a sequential and organized flow of asynchronous operations.

Q4. What is NPM?

🡺 NPM (Node Package Manager) is the default package manager for Node.js, streamlining the installation and management of JavaScript libraries and tools. It provides a vast repository of open-source packages, making it easy for developers to integrate and share code across projects. NPM simplifies dependency management, allowing users to declare and install project dependencies effortlessly using simple commands like npm install.

Q5. What are the modules in Node.js? Explain

🡺 In Node.js, modules are units of encapsulated code that help organize and structure a program's functionality. The CommonJS module system is used in Node.js, where each file is treated as a module, and code within a module is isolated from the rest of the application. Modules can include variables, functions, and classes, and their contents are scoped to the module by default.

To use code from one module in another, the ‘**require**’ function is employed to import modules, and the ‘**module.exports**’ or ‘**exports’** object is used to expose specific functionalities to other modules. This modular architecture facilitates code reuse, maintainability, and collaboration in Node.js applications by breaking down complex systems into manageable, independent components.