**Node Js Assignment**

**Module-1: Node-Javascript Fundamental**

**Q.1) what is difference between java & javascript?**

**Ans:**  java and Javascript difference between in this detail.

**:- java:** java is a strongly typed language and variables must be declared first to use in the program. In java, the type of a variable is checked at compile-time.

**:- Javascript:** Javascript is loosely typed language and has a more relaxed syntaxe and rules.

**:- java:** java is an object oriented programming language primarily used for developing complex enterprise applications.

**:- Javascript:** Javascript is a scripting language used for creating interactive and dynamic web pages.

**:- java :** java applications can run in any virtual machine (JVM) or browser.

**:- Javascript :** Javascript code used to run only in the browser, but now it can run on the server via Node.js.

**:- java :** object of java are class-based even we can’t make any program in java without creating a class.

**:- Javascript:** Javascript objects are prototype-based.

**:- java :** java is a standalone language.

**:- Javascript:** contained within a web page and integrates with its HTML content.

**Q.2) What is Javascript ?**

**Ans:** Javascript is a computer programming language used to make websites and applications dynamic and interactive.

:- it’s unique because it can run directly in your browser, not just on a server.

:- Along with hypertext markup language (HTML) and cascading style sheets (CSS), Javascript is one of the most commonly used programming languages of the internet.

:- Javascript, CSS, and HTML work together to make up the user facing elements of most websites and online applications.

:- Think of these coding languages as the components of a house:

:- HTML is the foundation of the house. It provides the basic layout, structure and content of a website.

:- CSS is the interior design. It provides design, font, colours, effects, and other visual elements.

:- Javascript is the electrical and plumbing systems. Js brings dynamic and interactivity to the website.

:- without Javascript, web pages would be mostly static and boring.

:- Javascript adds behavior and interactivity. But it can do so much more.

:- Brendan Eich developed javascript in 1995 while working for Netscape.

**Q.3) What are the data types supported by Javascript ?**

**Ans:** Javascript support several data type, Which can be broadly categorized into two groups:

**:- (1)** Primitive Data Types.

**:- (2)** Object Data Types.

**:- (1) Primitive Data Types:-**

**.String:-** Represents textual data.

**.Number:-** Represents numeric values, including integers

and floating-point number.

**.Boolean:-** Represent either ‘true’ or ‘false ’.

**.undefined:-** Represents an uninitialized variable or

Object property.

**.Null:-** Represents the absence of any object value.

**:- (2) Object Data Types:-**

**.Object:-** A collection of key-value pairs, where values can

Be of any data type.

**.Array:-** An order list of values.

**.Function:-** A reusable block of code that can be invoked

By its name.

**.Data:-** Represent dates and times.

**.RegExp:-** Represents regular expression for patterns for

Matching.

:- it’s define primitive data type and object data type in define.

**Q.4) What are the scopes of a variable in Javascript?**

**Ans:** Javascript variable scopes determine where in your code a variable is accessible or visible.

**[1] Global scope:-** A variable declared outside any function

Or block has global scope.

:- Global variables are accessible from any part of your code,

Including functions.

:- They persist as long as your application is running.

**[2] Local scope:-** A variable declared inside a function has

Local scope.

:- Local variables are only accessible within the function where they are declared.

:- They are not visible outside of the function.

**[3] Block scope:-** introduced with the ‘let’ and ‘const’

Keywords in ECAMA Script 6(ES6).

:- Variables declared with ‘let’ and ‘const’ have block scope, which means they are only accessible with in the block where they are defined.

:- Blocks include loops, conditionals, and any code surrounding by curly braces ‘{ }’.

**[4] Function parameters:-** parameters of a function act as

Local variables within that function.

**[5] Lexical scope:-** Javascript has lexical scooping, which

Means a function can access variables from its outer scope.

:- This is particularly relevant when dealing with closures.

**Q.5) What is callback ?**

**Ans:** A callback function is a function passed in to another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

:- The consumer of a callback based API writes a function that is passed in to API. The provider of the API (Called the caller) takes the function and calls back the function at some point insider the caller’s body.

:- The caller is responsible for passing the right parameters in to the call back function. The caller may also expect a particular return value from the call back function, which is used in to instructor further behavior of the caller.

:- There are two ways in which the callback may be called:

Synchronous. Synchronous callbacks are called immediately after the invocation of the outer function, with no intervening asynchronous tasks, while asynchronous callbacks are called at some point is particularly later, after an asynchronous operation has completed.

**Q.6) What is closure? Give an example.**

**Ans:** A closure is a programming concept where a function retains access to variables from its containing scope even after the scope even after the scope has finished executing.

:- In simpler terms, a closure allows a function to “remember ” the environment in which it was created. This behavior is particularly useful in situations where you want to create and return a function with some specific behavior or state.

:- Closure example in details about Javascript.

**Example :-**

function outerFunction(x) {

**//Inner function is defined inside the outer function.**

Function innerFunction(y) {

return x + y;

**// innerFunction has access to the ‘x’ variable from outerFunction**

}

Return innerFunction;

}

**//create a closure by calling outerFunction with a value.**

Const closure = outerFunction(5);

**//use the closure to add a value to the original ‘x’**

Const result = closure(25);

**//Result is 30 (5 + 25)**

**:-** In this example, ‘outerFunction’ takes a parameter ‘x’ and defines an inner function ‘innerFunction’. When ‘outerFunction’ is called with the argument ‘5’, it returns ‘innerFunction’. The returned function still has access to the ‘x’ variable, allowing it to remember the value ‘5’ when later invoked with closure(25)

:- The result is about 30 (5 + 25).

**Q.7) what is the difference between the operators ‘==’ &**

**‘===’?**

**Ans:** The ‘==’ and ‘===’ operators are used for equality comparison in various programming languages, but they differ in their strictnes.

**[1] ‘==’ (Equality operator):-** This operator checks for equity

Of values but preforms type coercion if the operands are of different types.

:- It only checks weather the values are equal after converting them to a common type.

:- Equality operator(‘==’) example in this details

**:- Example :-**

:- 5 == “5”

**//True**

**[2] ‘===’ (Strict Equality Operator):-** This operator checks for

Both equality of valuesand equality of types without performing any type coercion.

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

:- It is often considered safer and is recommended for most equality comparisons to avoid unexpected results due to type correction

:- Strict Equality Operator (‘===’) example in this details.

**:- Example :-**

:-“5” === 5

**//False**

**Q.8) What is the difference between Null & undefined ?**

**Ans:** Javascript is the set of both Undefined and Null represent the absence of a meaningful value but they are used in slightly different contexts.

:- In this article we are going to learn the difference between undefined and null in Javascript.

**[1] Undefined:-** When a variable is declared but not a word

Initialized, or when a function does not return a value, the variable or the function’s result is undefined.

:- Accessing an object property or array element that does not exist also results in undefined.

:- It is a primitive value.

**[2] Null :-** It is a deliberate assignment that represents the

absence of any object value.

:- It is often used to explicitly indicate that a variable or object property should have no value or no reference to any object.

:- It is also a primitive value.

:- It is defined undefined value and null value in briefly explain.

**Q.9) What would be the result of 2+5+”3”?**

**Ans:** It is the most programming languages when you use the ‘+’ operator with a combination of numbers and strings the operation is performed from left to right.

:- In this case, the additional value it’s would be carried out as follows.

:- 2 + 5 equals 7 (Numerica addition)

:- 7 + “3” would result in the concatenation of the numeric result (7) with the string “3”.

:- As a result, the final outcome would be the string “73”.

**Q.10) What is the difference between Call & Apply?**

**Ans:** “Call” and “Apply” are both methods in Javascript that allow you to invoke a function with a specified ‘this’ value and arguments provided in a different way.

**[1] Call() method:-** The call method is basically used to

Invoke the function with different this object. In javascript, this refers to an object. It depends on how we are calling a particular function.

:- In the global scope, this refers to the global object window. Inside the function also this refer to the global object window.

:- In strict mode, when we use any function then this refers to undefined in functions like call, this could refer to a different object. With the help of the call method, we can invoke a particular function with different objects.

:- **Syntax :-**

object.objectMethod.call(objectinstance,arguments)

:- **Objectinstance:** it is an object which we want to use explicitly.

:- **Arguments:-** it is arguments that we want to pass to the calling function.

**[2] Apply() method:-** just like the call method we want to

You can also bind the function to any object. Using apply() method also we can invoke a given function with different objects.

:- **Syntax:-** object.objectMethod.apply(objectInstance, arrayofArguments).

:- **Objectinstance:** it is an object which we want to use of

Explicitly.

:- **Arguments:-** it is arguments that we want to pass to the calling function.

:- it is define about call method and apply method in define the condition.