Assignment _ Javascript Introduction

Dr. Ghulam Shabbir

Q.1: Introduction to Programming Concepts in JavaScript

JavaScript is a popular programming language used for developing web applications, games, and desktop applications. It's an object-oriented language, which means it's based on the concept of objects and their interactions. In JavaScript, everything is an object or a primitive data type.

Here are some basic programming concepts in JavaScript:

- Variables: Variables are used to store data values in JavaScript. They are declared using the var, let, or const keyword followed by the variable name.

Example: Code

```
var nameshabbir = "Bilal";
var message = "Hello, " + nameshabbir + " Would you like to learn some Python today";
console.log(message);
```

Output:

```
PS C:\Users\DELL\OneDrive\Documents\Javascript_Basics\Asst1> node main.js
Hello, Bilal Would you like to learn some Python today
```

- Data types: JavaScript supports various data types, such as numbers, strings, booleans, null, undefined, objects, and arrays.

Example: Code

```
var myNum = 10; // Number
var myString = "Hello"; // String
var myBool = true; // Boolean
var myArray = [1, 2, 3]; // Array
var myObject = {name: "John", age: 30}; // Object

console.log(myNum);
console.log(myString);
console.log(myBool);
console.log(myArray);
console.log(myObject);
```

Output:

```
PS C:\Users\DELL\OneDrive\Documents\Javascript_Basics\Asst1> node main.js
10
Hello
true
[ 1, 2, 3 ]
{ name: 'John', age: 30 }
```

- Operators: JavaScript has arithmetic, comparison, logical, and assignment operators.

Example: Code

```
var num1 = 10;
var num2 = 5;
var sum = num1 + num2; // Addition operator
var greaterThan = num1 > num2; // Comparison operator
var logicalAnd = (num1 > 5) && (num2 < 10); // Logical operator

console.log(sum);
console.log(greaterThan);
console.log(logicalAnd);
```

Output:

```
PS C:\Users\DELL\OneDrive\Documents\Javascript_Basics\Asst1> node main.js
15
true
true
```

- Control flow: JavaScript has conditional statements (if/else, switch) and loops (for, while) for controlling program flow.

Example: Code

```
// var age = 20;
var age = 10
if (age >= 18) {
  console.log("You are a young man");
} else {
  console.log("You are a child");
}

for (var i = 0; i < 5; i++) {
  console.log(i);
}
```

Code:

```
PS C:\Users\DELL\OneDrive\Documents\Javascript_Basics\Asst1> node main.js
0
1
2
3
4
PS C:\Users\DELL\OneDrive\Documents\Javascript_Basics\Asst1> node main.js
You are a child
0
1
2
3
4
```

- Functions: Functions are reusable blocks of code that can take parameters and return values.

Example: Code

```
function addNumbers(num1, num2) {
    return num1 + num2;
}

var result = addNumbers(5, 10);
console.log(result);
```

Output:

```
PS C:\Users\DELL\OneDrive\Documents\Javascript_Basics\Asst1> node main.js
15
```

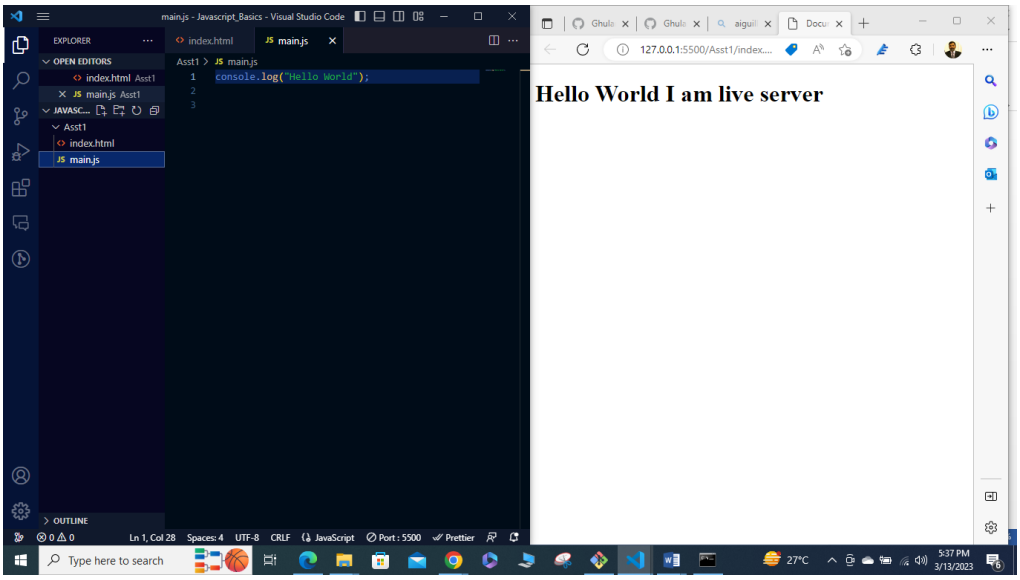
Q.2: Research and write a paragraph defining a source code editor and an Integrated Development Environment (IDE).

A source code editor is a software application that allows developers to write, edit, and manage source code for programming languages like JavaScript and TypeScript. These editors typically provide features like syntax highlighting, code completion, code folding, and multiple language support. Some popular source code editors for JavaScript and TypeScript include Visual Studio Code, Atom, Sublime Text, and Notepad++.

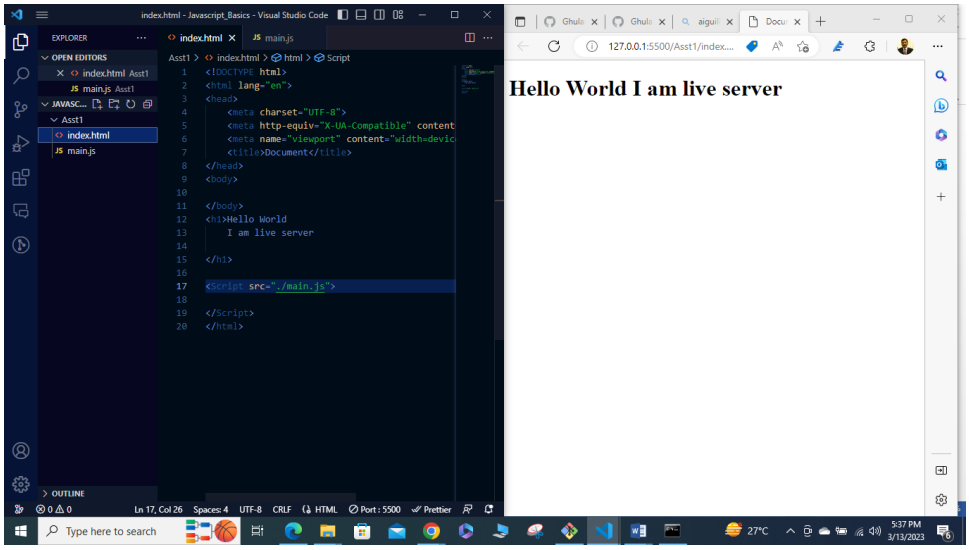
An Integrated Development Environment (IDE) is a software application that provides a complete development environment for coding, debugging, and deploying applications. In addition to the features provided by a source code editor, an IDE may also include integrated debugging tools, project management features, and built-in version control. Some popular IDEs for JavaScript and TypeScript include Visual Studio Code (which can function as both a source code editor and an IDE), WebStorm, and IntelliJ IDEA.

Following IDEs on Visual Studio Code were established for Javascript programming:

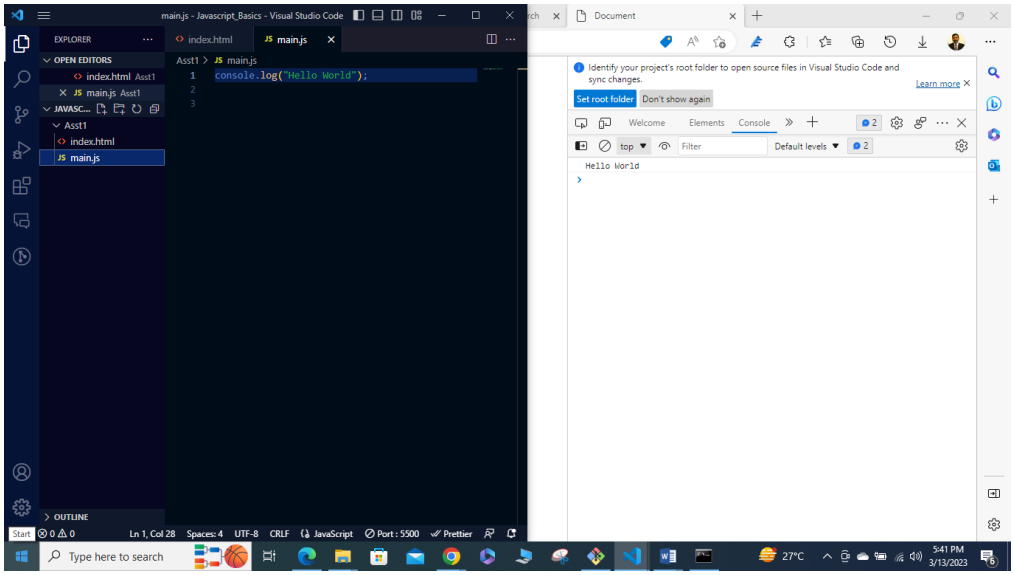
(1). Installation of Live Server Extension for run-time saving on html page with console.



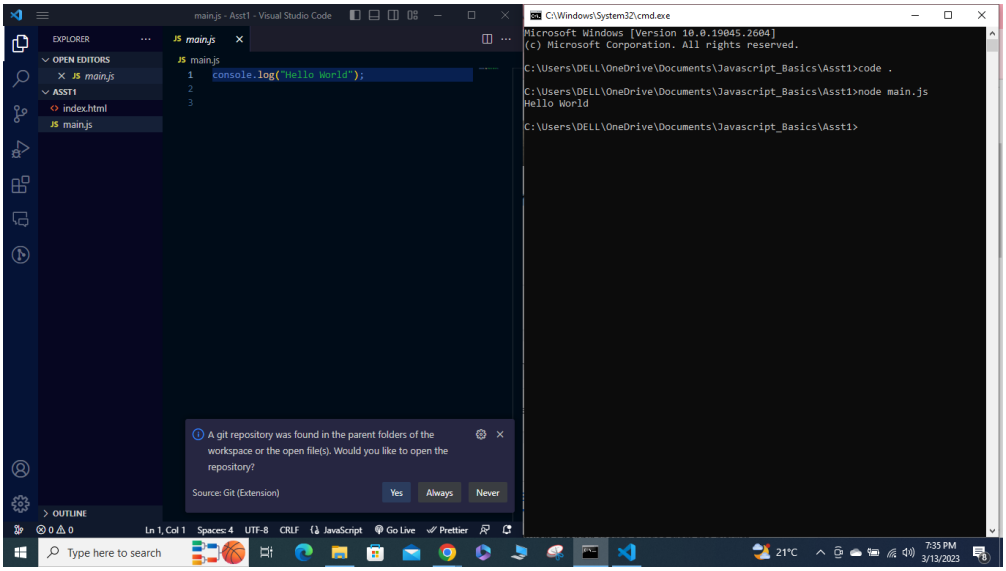
(2). Binding html file with Javascript file on Visual Studio Code:



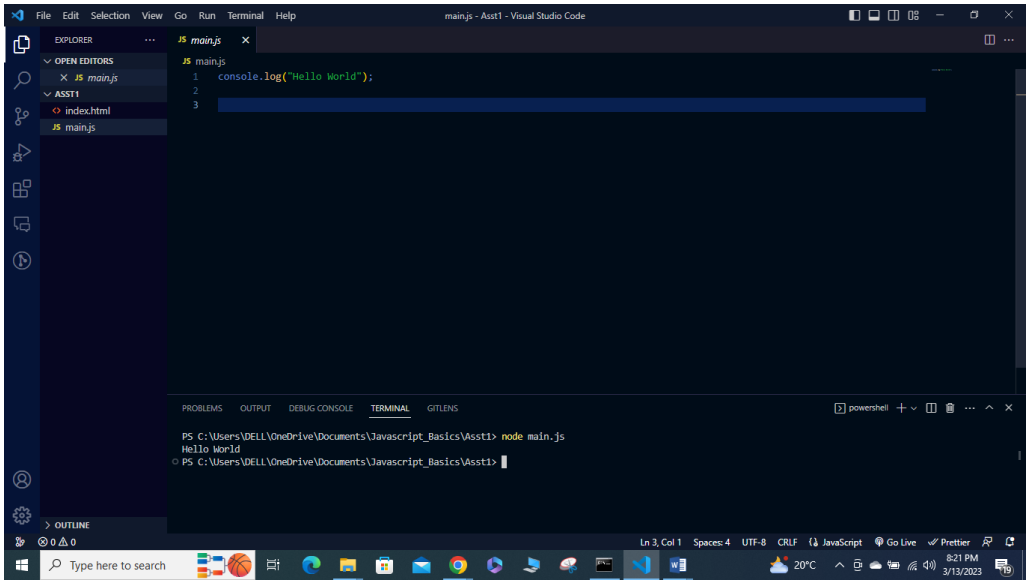
(3). Installation Integrated Development Environment (IDE) for Javascript on Visual Studio Code



(4). IDE on VS Code with command-line interpreter (cmd) of Windows operating systems



(5). IDE on VS code with inbuilt default terminal



Q.3: Create a table listing the differences between IDE and source code/text editor.

Feature	IDE (Visual Studio Code)	Source Code/Text Editor (Console, PowerShell, Command Line (cmd), Bash)
Code Editing	Full-featured code editing capabilities, such as syntax highlighting, autocompletion, and refactoring tools.	Basic code editing capabilities, such as syntax highlighting and basic text manipulation features.
Debugging	Built-in debugging tools, including breakpoints, step-by-step execution, and inspection of variables and memory.	Limited debugging capabilities, such as console output or third-party debugging extensions.

Project Management	Integrated project management features, such as version control integration, build tools, and project templates.	Basic file management features, such as opening and saving files, and file navigation.
Collaboration	Built-in collaboration tools, such as shared editing, code review, and real-time collaboration.	Limited collaboration capabilities, such as version control systems or third-party collaboration tools.
Language Support	Full support for the JavaScript language, including syntax checking, error highlighting, and type checking.	Basic support for the JavaScript language, including syntax highlighting and some error checking.
User Interface	A full-featured user interface, including customizable layouts, themes, and menus.	A minimal user interface, with basic customization options.
Learning Resources	Built-in tutorials, documentation, and community support to help users learn and troubleshoot.	Limited learning resources, such as online documentation or third-party tutorials.
Cost	Some IDEs are free, but many require a license or subscription fee.	Many source code/text editors are free and open-source, while some offer paid versions with additional features.

Q.4: Write a paragraph explaining the environment setup process and its importance in programming.

Environment setup is a crucial step in the programming process, as it involves configuring the necessary software and tools that enable developers to write and run their code. The setup process typically involves installing and configuring software such as an IDE or text editor, a compiler or interpreter, and any necessary libraries or frameworks. Additionally, developers may need to configure their development

environment to work with version control systems, testing frameworks, and deployment tools. The importance of environment setup lies in the fact that without a properly configured development environment, developers cannot effectively write, test, or deploy their code. By taking the time to properly configure their environment, developers can ensure that they have the tools they need to be productive and efficient, and that their code runs smoothly across various platforms and environments.

Q.5: Research and write a paragraph on what Visual Studio Code (VS Code) is and why it is considered a hybrid of a text/source editor and an IDE.

- Visual Studio Code (VS Code) is a free and open-source text editor developed by Microsoft. It has gained immense popularity among developers due to its versatility and flexibility.
- VS Code is widely regarded as a hybrid of a text/source editor and an Integrated Development Environment (IDE) because it offers many of the advanced features of an IDE while retaining the simplicity and flexibility of a text editor.
- VS Code offers a comprehensive set of features that include syntax highlighting, auto-completion, debugging, version control integration, and many more.
- Additionally, it supports a vast array of programming languages and offers a vast library of extensions that enable users to customize their development environment according to their specific needs.
- This unique combination of features has made VS Code the go-to choice for many developers who require the power of an IDE but prefer a lightweight and flexible development environment.

Q.6: Write a paragraph introducing data types and provide a list of data types available in JavaScript.

In programming, data types are a fundamental concept that refers to the classification of different types of data that can be used in a program. These data types help define the values that a program can store, process, and manipulate. In JavaScript, there are several built-in data types that programmers can use to define their variables and values. These data types include primitive types such as numbers, strings, booleans, null, and undefined, as well as complex data types such as objects and arrays. Each data type has its own unique characteristics, which can impact how a program stores and manipulates data. Understanding data types is critical in programming, as it can help ensure that a program operates as intended and can prevent errors and unexpected results.

Here's a list of the data types available in JavaScript:

- Number
- String
- Boolean
- Null
- Undefined
- Object
- Symbol (added in ES6)
- BigInt (added in ES2020)

Q.7: Explain the concepts of strongly typed and loosely/weakly typed languages with examples.

Strongly Typed Language

A strongly typed language is one in which the types of variables are checked at compile-time or at run-time, ensuring that variables are only used in ways that make sense. The compiler or interpreter will flag any attempt to assign a value of one type to a variable of another type, or any attempt to use a variable in an incompatible way.

Examples

C++ and Java are strongly typed languages. In C++, the following code would cause a compile-time error:

```
int x = 5;
char c = 'a';
x = c; // type mismatch: char cannot be converted to int
```

In Java, the following code would also cause a compile-time error:

```
int x = 5;
String s = "hello";
x = s; // type mismatch: String cannot be converted to int
```

Loosely/Weakly Typed Language

A loosely/weakly typed language is one in which type checking is not enforced, or is enforced to a lesser degree. In these languages, variables can be used in a variety of ways, regardless of their type.

For example, JavaScript and PHP are loosely/weakly typed languages. In JavaScript, the following code would not cause an error:

```
var x = 5;
var s = "hello";
x = s; // no error, x now contains the string "hello"
```

Similarly, in PHP, the following code would not cause an error:

```
$x = 5;
$s = "hello";
$x = $s; // no error, $x now contains the string "hello"
```

The lack of strict type checking in these languages can make them easier to use and more flexible, but it can also lead to unexpected behavior and errors if not used carefully.

Q.8: Differentiate between static and dynamic types with examples.

Static types and dynamic types are two different approaches to type systems in programming languages.

Static types are types that are determined at compile-time, and the type of a variable cannot change during runtime. In a statically typed language, the data type of a variable must be declared explicitly and checked by the compiler before the program is run. Any mismatch between the declared type and the actual value assigned to the variable will be caught by the compiler and reported as an error.

Examples of statically typed languages include Java, C++, and C#. Here's an example in Java:

```
int x = 5; // x is statically typed as an integer
x = "hello"; // compile-time error: string cannot be assigned to integer
```

Dynamic types, on the other hand, are types that are determined at runtime. In a dynamically typed language, the data type of a variable can change during runtime, based on the value assigned to it. Dynamic typing allows for greater flexibility and ease of use, as the programmer doesn't need to worry about declaring types explicitly. However, it can also make the code more error-prone, as type errors may only be detected at runtime.

Examples of dynamically typed languages include JavaScript, Python, and Ruby. Here's an example in Python:

```
x = 5  # x is dynamically typed as an integer
x = "hello"  # no error, x is now dynamically typed as a string
```

In summary, the main difference between static and dynamic types is when the type of a variable is determined: at compile-time (static) or at runtime (dynamic).

Q.9: Explain the concepts of implicit and explicit in JavaScript.

In JavaScript, "implicit" and "explicit" refer to the way in which the language handles type conversions.

Implicit conversions occur when JavaScript automatically converts a value from one data type to another, without the programmer explicitly specifying the conversion. This can happen in a variety of situations, such as when performing arithmetic operations on different data types or when using comparison operators.

For example, in JavaScript, the following code would result in an implicit conversion:

```
var x = 5;
var y = "10";
var z = x + y; // z will be "510" (a string)
```

Here, the string "10" is implicitly converted to a number in order to perform the addition operation with the integer 5.

Explicit conversions, occur when the programmer explicitly specifies the conversion using built-in functions or operators. This is also known as "**type casting**."

For example, in JavaScript, the **Number()** and **String()** functions can be used to explicitly convert a value to a number or a string, respectively:

```
var x = "5";
var y = Number(x); // y will be 5 (a number)

var z = 10;
var w = String(z); // w will be "10" (a string)
```

Here, the Number() function is used to explicitly convert the string "5" to a number, and the String() function is used to explicitly convert the number 10 to a string.

"implicit" and "explicit" in JavaScript refer to whether the language automatically handles type conversions or whether the programmer must explicitly specify the conversion.

Q.10: Describe variables in JavaScript and provide a list of variable types available.

In JavaScript, a variable is a container that holds a value, such as a number, a string, or an object. The value stored in a variable can be changed or accessed later in the program.

To create a variable in JavaScript, you can use the var, let, or const keyword, followed by the variable name:

```
var x = 5;  // creates a variable called x and assigns the value 5 to it
let y = "hello";  // creates a variable called y and assigns the string "hello" to it
const z = true;  // creates a constant variable called z and assigns the value true to it
```

In this example, var, let, and const are used to create variables of different types. Here's a list of the different variable types available in JavaScript:

Number: used to store numeric values, such as 3, 3.14, or 2.5e6

String: used to store text values, such as "hello", "world", or "JavaScript"

Boolean: used to store true or false values

Null: used to represent a null value, indicating that a variable has no value

Undefined: used to represent a variable that has been declared but not yet assigned a value

Object: used to store a collection of key-value pairs, such as { name: "John", age: 30 }

Symbol: used to create unique identifiers for object properties

Additionally, JavaScript also supports dynamic typing, which means that a variable can change its type at runtime. This allows for greater flexibility but can also make code more error-prone if types are not managed carefully.

Q.11: Explain why objects are considered the king in JavaScript.

Objects are considered the "king" in JavaScript because they are the primary data type in the language and form the foundation of many JavaScript features.

- In JavaScript, an object is a collection of key-value pairs, where each key represents a property name and each value represents the property value. Objects can be created using object literals, as shown below:

```
var person = { name: "Imran", age: 30 };
```

- Objects can also be created using constructor functions, which allow for more complex object creation and inheritance.

- Objects in JavaScript are dynamic and can be modified at runtime by adding, deleting, or updating properties. This makes them very flexible and allows for powerful programming techniques such as object-oriented programming.
- In addition to being the primary data type in JavaScript, objects are used extensively in the language's built-in APIs, such as the Document Object Model (DOM) for manipulating HTML documents, and the XMLHttpRequest object for making HTTP requests.
- Furthermore, many other JavaScript data types are actually objects under the hood, such as arrays, functions, and regular expressions. This allows for additional functionality and methods to be added to these data types.

Q.12: Explain the differences between null and undefined.

In JavaScript, null and undefined are both used to represent the absence of a value, but they are used in slightly different ways.

undefined is a primitive data type in JavaScript that is automatically assigned to a variable that has been declared but not yet initialized with a value. It can also be explicitly assigned to a variable to indicate that it has no value.

undefined is used to represent the absence of a value when a variable has not been initialized or does not exist

Example:

```
var x; // x is undefined
var y = undefined; // y is explicitly assigned undefined
```

null is an object that represents the intentional absence of any object value. It is often used to explicitly indicate that a variable has no value or that an object property should be empty.

null is used to represent the intentional absence of a value or an empty object property.

Example:

```
var z = null; // z is explicitly assigned null
var person = { name: "Ali", age: null }; // age property is explicitly set to null
```

Q.13: Describe the rules for naming variables in JavaScript.

JavaScript lets to give names to values using variables. We can think of a variable as a box that you can fit one thing in. If we put something else in it, the first thing goes away. To create a new variable, use the keyword var, followed by the name of the variable. A keyword is a word that has special meaning in JavaScript. In this case, when we type var, JavaScript knows that we are about to enter the name of a new variable.

In JavaScript, variables are used to store data values that can be used throughout the program. The rules for naming variables in JavaScript are as follows:

1. Variable names must begin with a letter, an underscore (_), or a dollar sign ($). They cannot begin with a number.
2. Variable names can contain letters, numbers, underscores, and dollar signs.
3. Variable names are case sensitive. For example, "myVar" and "myvar" are two different variables.
4. Variable names should be descriptive and meaningful. Avoid using generic names such as "temp" or "data" and use descriptive names like "userName" or "userAge".
5. JavaScript reserved words such as "if", "else", "function", "return", "var", etc. cannot be used as variable names.
6. Variable names should not contain spaces or special characters except for underscores and dollar signs.

Q.14: Examples of valid variable names in JavaScript:

In JavaScript, a variable name can consist of letters, numbers, underscores, and dollar signs. However, it must begin with a letter, underscore, or dollar sign (not a number), and it cannot be a reserved keyword.

Some examples of valid variable names in JavaScript:

```
var myVariable;
var _underscoredVariable;
var $dollarVariable;
var camelCaseVariable;
var PascalCaseVariable;
var long_variable_name;
```

Some examples of invalid variable names in JavaScript:

```
var 1stVariable; // cannot begin with a number
var variable-name; // cannot contain hyphens
var function; // cannot be a reserved keyword
```

It's also worth noting that variable names are case sensitive in JavaScript, so myVariable and myvariable would be considered two different variables.

Q.15: Explain the difference between case sensitive and case insensitive in JavaScript.

In JavaScript, case sensitive refers to the property of treating characters as distinct based on their case, while case insensitive refers to the property of treating characters as the same regardless of their case.

For example, in JavaScript, myVariable and myvariable are considered two different variable names because the language is case sensitive. This means that JavaScript distinguishes between uppercase and lowercase letters and treats them as distinct.

```
var myname = "Jeet Kumar";
console.log(myname);
console.log(myName); // This is wrong. In javascript it is not valid javascript is case sensitive language
```

On the other hand, many JavaScript operations, such as string comparisons, are case insensitive by default. This means that they treat characters as the same regardless of their case. For example, the following comparison is case insensitive and returns true:

```
"hello" === "HELLO"      // returns true

//Case Insenstive like php
// php code example
// $myvar = "jeet kumar"
// print($myvar);
// print($MyVar)
// print($MYVAR)
```

However, in some cases, it is possible to make operations case sensitive by using the appropriate flags or options. For example, the indexOf method can be made case sensitive by using the indexOf method with the sensitive flag:

```
"Hello World".indexOf("WORLD", 0, true)      // returns -1 (case sensitive)
"Hello World".indexOf("WORLD", 0, false)     // returns 6 (case insensitive)
```

In summary, case sensitive in JavaScript refers to treating characters as distinct based on their case, while case insensitive refers to treating characters as the same regardless of their case. Many JavaScript operations are case insensitive by default, but it is possible to make them case sensitive in some cases.

References

(1). Class notes
(2). GPT Chat
(3). Internet
(4). Some books provided by PIAIC