AIMERZ.ai
Aim.Act.Achieve

NOTES

Introduction to Javascript

Introduction To Javascript

JavaScript is a popular programming language that is widely used to build web applications. It is a client-side scripting language, which means that it is run by your web browser rather than on a server. This makes it a good choice for building web applications that need to be fast and responsive, as the code is run locally on the user's device rather than having to be sent back and forth between a server and a client. JavaScript is also used to build **mobile apps,** create **interactive documents**, and build **server-side applications** with the help of runtime environments such as **Node.js.**

History Of Javascript

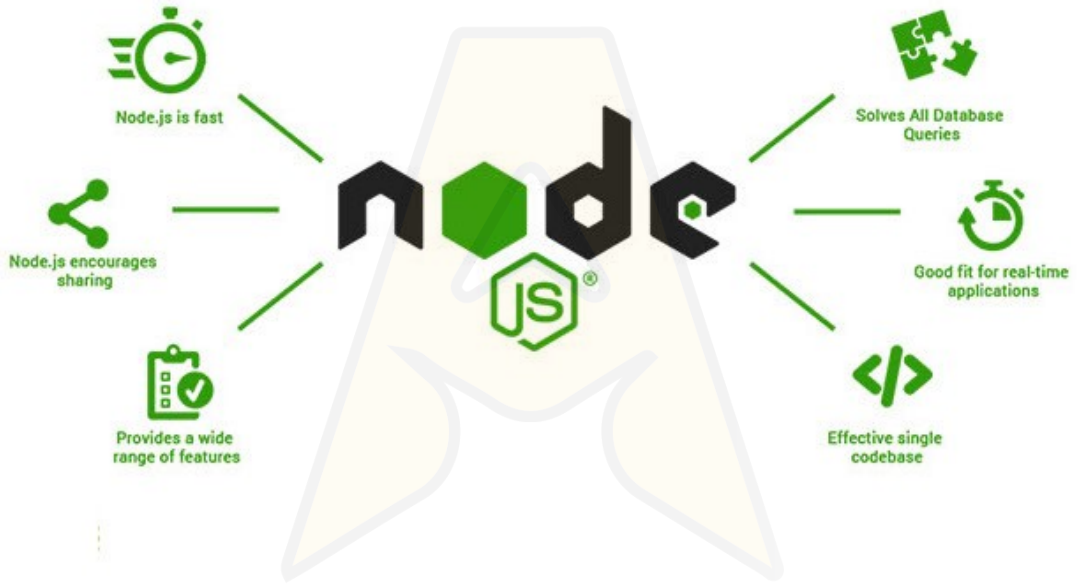
JavaScript was created in **1995** by **Brendan Eich**, a programmer at Netscape Communications Corporation. It was originally called **Mocha**, then changed to **LiveScript**, and finally, it was given the name JavaScript to leverage the popularity of Java, which was a popular programming language at the time. JavaScript was first introduced in Netscape Navigator 2.0, a popular web browser of the time.

In 1996, JavaScript was submitted to the European Computer Manufacturers Association (ECMA) and it was standardized as ECMAScript. This standardized version of JavaScript is still used today and is supported by all modern web browsers like Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, etc.

In the early days of JavaScript, it was primarily used for simple things like form validation and simple mouse interactions, but as browsers became more powerful and web standards evolved, JavaScript became more widely used for building more complex

web applications. With the introduction of popular libraries and frameworks like jQuery, AngularJS, React, and Vue.js, it has become easier to build complex and powerful web applications using JavaScript.

The emergence of Node.js in 2009 made it possible to run JavaScript on the server side and it became more popular as a full-stack language, allowing for code reuse and sharing between the client side and the server side.



Features Of Javascript

1. **Ease of use:** JavaScript is a high-level language, which means that it is easy to learn and use. It is also a dynamically-typed language, which means that you don't have to specify the type of a variable when you declare it, making it easy to write code quickly.

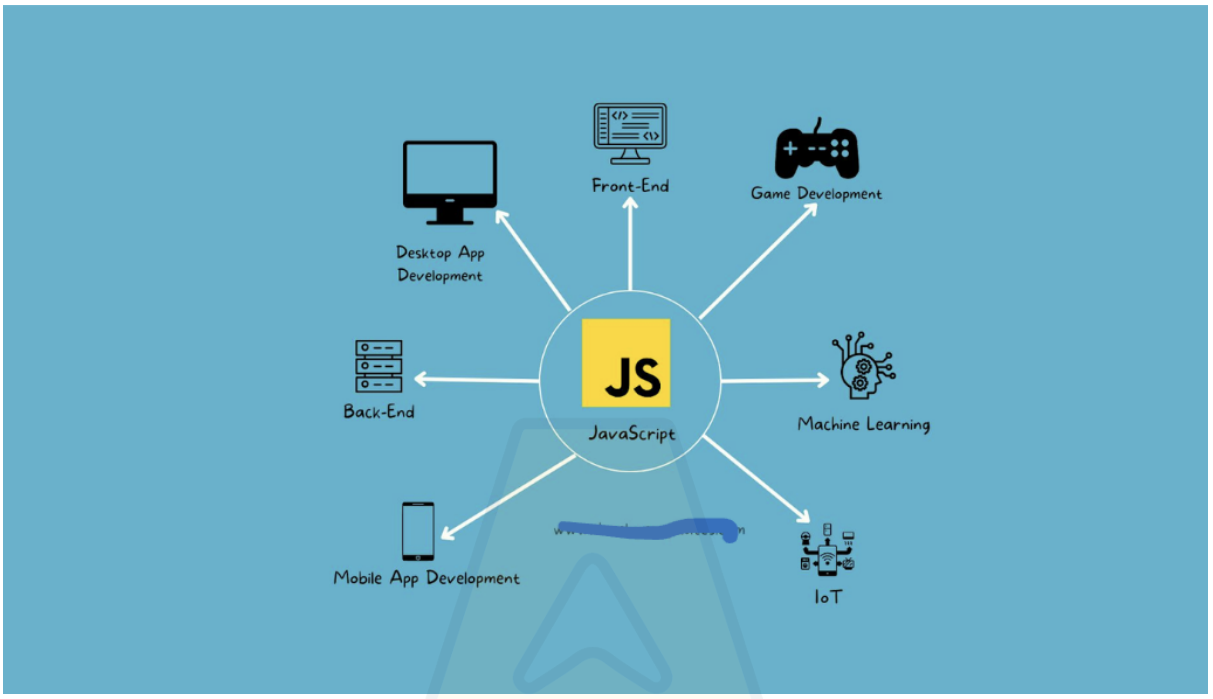
2. **Cross-platform compatibility:** JavaScript is supported by all modern web browsers, so you can use it to build web applications that will run on any device with a web browser.

3. **A large developer community:** There is a large and active community of developers who use and contribute to JavaScript, which means that there are many resources available for learning the language and getting help when you need it.

4. **Powerful capabilities:** JavaScript has a lot of powerful features that allow you to build complex and interactive applications. For example, you can use JavaScript to manipulate the HTML and CSS of a web page, make asynchronous network requests, and work with multimedia and other types of data.

5. **Growing demand:** The demand for JavaScript developers is high and continues to grow, making it a good language to learn if you want to pursue a career in software development.

Application Of Javascript



JavaScript is a very flexible and powerful programming language. It is used in many different areas because it can work both in **web browsers** (the part of the internet you see) and on **servers** (computers that provide information to web browsers).

Some of the Common Applications of Js:

1. **Web Development:** JavaScript is primarily used to add interactivity and dynamic elements to websites. It allows developers to create responsive user interfaces, handle user interactions, and update web page content without requiring a full page reload.
2. **Web Applications:** With the help of modern JavaScript frameworks like React, Angular, and Vue.js, developers can build powerful and complex web applications that provide a smooth user experience similar to traditional desktop applications.

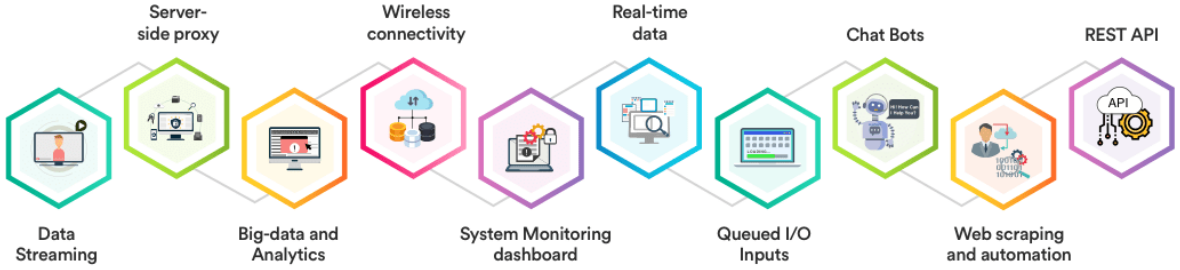
3. **Server-Side Development:** JavaScript is used with Node.js to build server-side applications. Node.js allows developers to create high-performance, scalable, and event-driven server applications that can handle a large number of simultaneous connections.
4. **Mobile App Development:** JavaScript is used in combination with frameworks like React Native and Ionic to develop cross-platform mobile applications that run on both Android and iOS devices. This approach allows developers to write code once and deploy it on multiple platforms.
5. **Game Development:**With the arrival of HTML5 and WebGL, JavaScript has become a practical option for making games that can be played directly in web browsers. Game engines like Phaser and Three.js make it easy for developers to create fun and visually attractive games using JavaScript.
6. **Browser Extensions:** JavaScript is commonly used to develop browser extensions and add-ons that enhance the functionality of web browsers, adding new features or customizing the browser's behavior.
7. **Data Visualization:** JavaScript, along with libraries like D3.js and Chart.js, is employed to create interactive and informative data visualizations on web pages, making complex data more accessible to users.
8. **Real-time Applications:** JavaScript's support for asynchronous programming makes it ideal for building real-time applications like chat applications, collaborative tools, and live data dashboards.
9. **Automation:** JavaScript can be used in automation to build applications.

Limitation Of Javascript

JavaScript, despite being a versatile and widely used programming language, has some limitations:

1. **Not supported by Old Browsers:** Although JavaScript is supported by almost all modern web browsers,some older web browsers may not fully support JavaScript, causing issues for users who use those browsers.
2. **Depends on the User's Browser:** JavaScript relies on the user's web browser, so if someone disables it, certain website features may not work as expected.
3. **Security Concerns:** JavaScript can be misused by hackers to attack websites and steal information if developers don't take proper precautions.
4. **Performance:** JavaScript performance may vary depending on the browser and device, and overly complex code can cause web applications to run slowly, negatively impacting user experience.
5. **Lack of Strong Typing:** JavaScript's dynamic typing can lead to errors that may only be caught during runtime, making it more challenging to detect certain types of coding mistakes during development.
6. **SEO Challenges:** Some search engine crawlers may not execute JavaScript, leading to potential SEO issues if critical content and links are hidden behind JavaScript-based interactions.

Setting up node js and vs code for running javascript



Before starting working on Javascript, We have to set up NodeJs and Vscode . Lets understand how to set up step by step process:

1. Install NodeJS:
    - Go to the Official website Of nodeJs.
    - Download the latest stable version(LTS) of Node Js for your operating system(Windows, macOS or Linux).
    - Run the installer and follow the installation instructions.

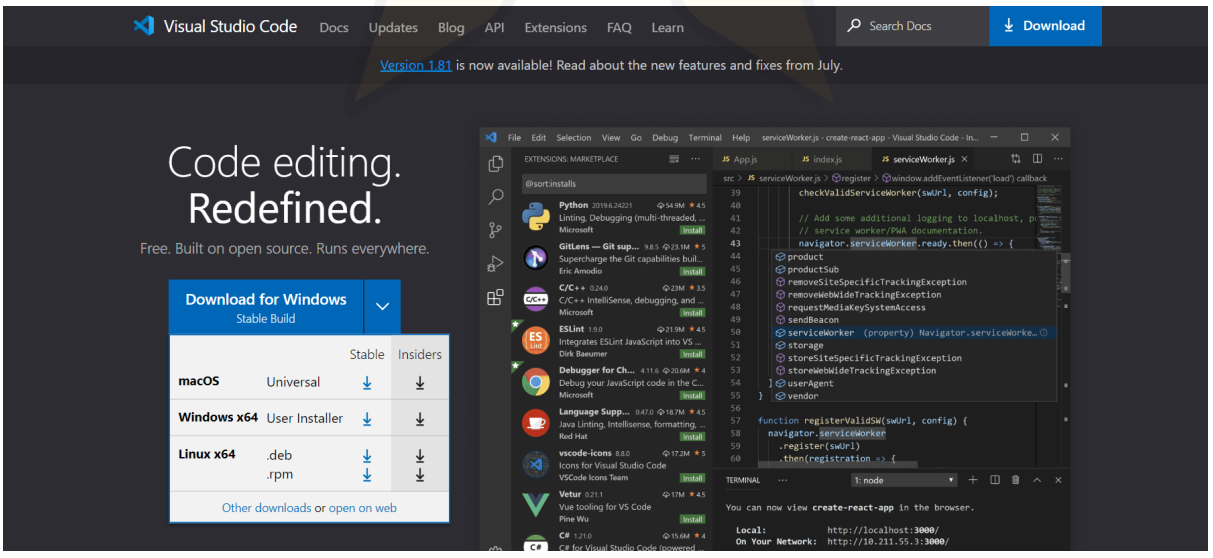
2. **Verify Node.js Installation:**
   - Open a terminal or command prompt.
   - Type node -v and press Enter. This will display the installed version of Node.js.

```
prabir~$node -v
v21.5.0
prabir~$
```

   - Type npm -v and press Enter. This will display the version of Node Package Manager (npm), which comes bundled with Node.js.

```
prabir~$npm -v
10.4.0
prabir~$
```

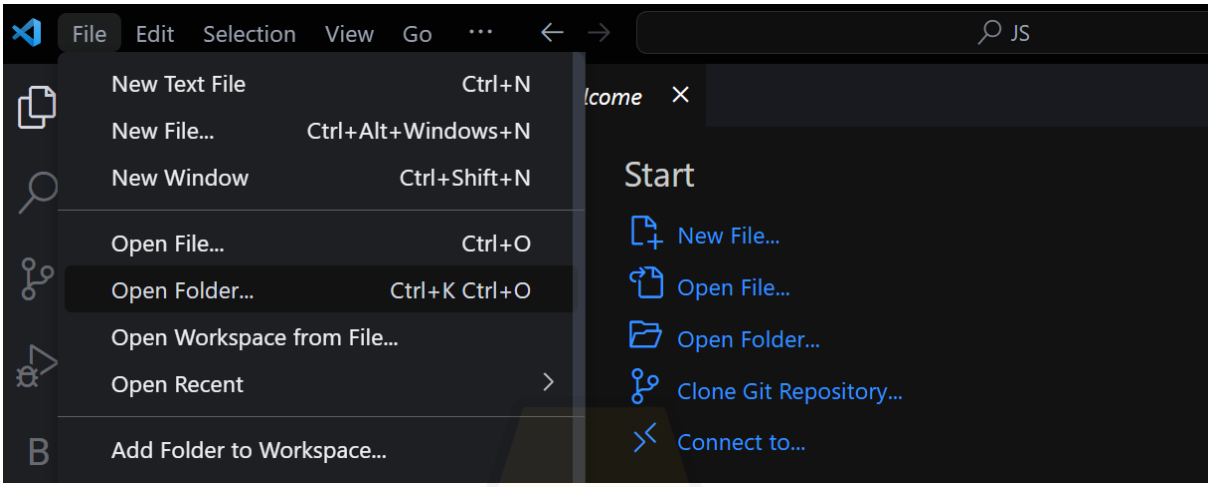
3. **Install Visual Studio Code:**
   - Go to the official VS Code website
   - Download the installer for your operating system.



   - Run the installer and follow the installation instructions.

4. Open a JavaScript Project in VS Code:

- Launch Visual Studio Code.
- Click on "File" in the top menu and select "Open Folder."



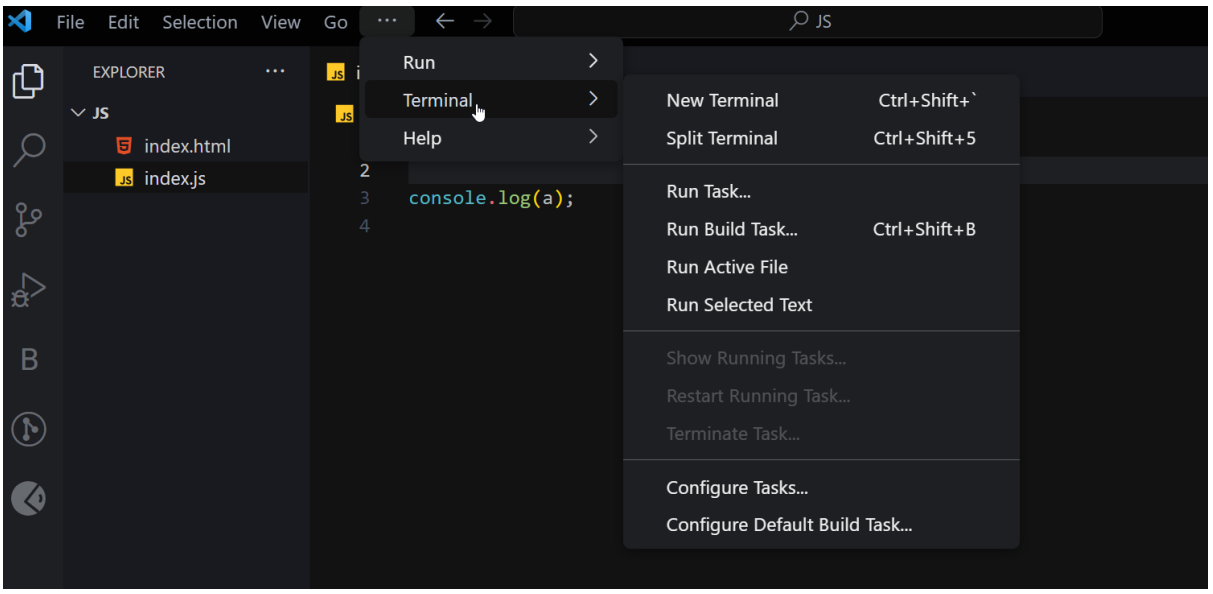
- Navigate to your JavaScript project folder and click "Select Folder."

5. Write and Run JavaScript Code:

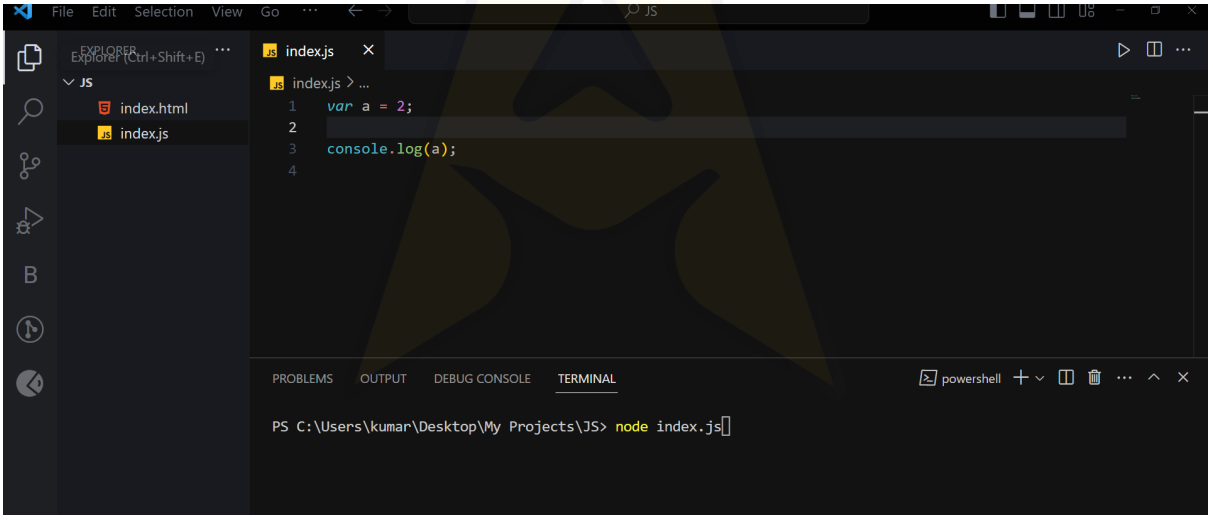
- Create a new JavaScript file (e.g., index.js) in your project folder or open an existing one.
- Write your JavaScript code in the file.

To run the JavaScript code:

- Open the integrated terminal in VS Code by clicking "View" in the top menu and selecting "Terminal."



- In the terminal, type **node index.js** (replace index.js with the name of your JavaScript file) and press Enter.



- The output of your JavaScript code will be displayed in the terminal.

JavaScript virtual machine

JavaScript can now run not just in browsers, but also on servers and any device with a JavaScript engine. A browser has a built-in engine, sometimes called a **JavaScript virtual machine.**

Different browsers use different engines, like:

- V8 in Chrome, Opera, and Edge
- SpiderMonkey in Firefox
- Chakra in Internet Explorer
- JavaScriptCore, Nitro, and SquirrelFish in Safari

How do JavaScript engines work?

- The engine reads (or **parses**) the JavaScript code.
- It then converts (**compiles**) the code into machine code that the computer can execute.
- Finally, the machine code runs very quickly.

Throughout this process, the engine constantly optimizes the code to make it run as efficiently as possible.

Values and Data type

Introduction To Data Type

- Data types are used to define the way the data is stored in memory. Storing data is an essential part of programming as it enables the manipulation, processing, and sharing of information within a program.

- The data type is a classification of data according to the type of value that we want to operate on.

- JavaScript is a dynamically typed language, which means the data type is identified during execution. The programmer need not to explicitly declare the data type in code

There are `**primitive types**` and `**non-primitive types**`. The big difference under the hood is the way that they're **stored** and **accessed from memory**.

A. **Primitive types:** Stored directly in the location that the variable is accessed.
B. **Non-primitive types:** The data types that are derived from primitive data types. It is also known as derived data types or reference data types.

Primitive Data types

Following are the 7 primitive types in javascript:

1. **String**

   Combination of alphanumeric characters wrapped in either single or double quotes. Strings can include any number, letter or symbol.

```javascript
"Prabir" or 'Prabir'
```

2. **Number**

   Any number in JavaScript is the Number type, including floats and decimals. Some languages have separate types for floats and integers. JavaScript does not. Numbers are not wrapped in quotes.

```javascript
30
98.9
```

Other possible number values are **infinity** and **NaN.**

Infinity is a special value that is greater than any number.

NaN stands for **"Not a Number"** and is a special value that represents the result of an undefined or unrepresentable mathematical operation.

We will learn more in detail in the upcoming lectures.

3. BigInt:

In JavaScript, there is a maximum safe value, which is approximately 2^53 - 1. Similarly, there is also a minimum safe value, which is approximate -(2^53 - 1).

This means that integers less than min safe value or greater than max safe value, may lose precision when represented as a JavaScript number. So, for such numbers, we use BigInt data type.

The BigInt data type number can also be treated as a regular number by adding n to it at the end.

```
JavaScript
9007199243740991n
```

4. Boolean:

A boolean is a true or false value.

```
JavaScript
true or false
```

5. **Null:**
   Null means nothing or empty value. It is often used to indicate that a variable or property has no value. We will learn more about variables in the upcoming lecture.

```javascript
null
```

6. **undefined**
   undefined is a special value that indicates that a variable or property has been declared but has not been assigned a value.

```javascript
undefined
```

7. **Symbol:**

   A Javascript Symbol is a relatively new javascript feature . It was introduced in ES6. This is a new Primitive data type called Symbol. Symbols are immutable i.e means can not be changed and are unique.

   We will learn more about symbol in the upcoming lectures.

```javascript
Symbol("name")
```

Non-primitive types:

Reference types are a non-primitive value and when assigned to a variable, the variable is given a reference to that value. The reference points to the object's location in memory. Unlike primitives, where the variable contains the actual value.

Arrays:

I know we haven't gone over arrays yet so don't worry if you've never worked with them. They're essentially a data structure that can hold multiple values. We will learn in depth in the upcoming lectures.

```javascript
[1,2,3,4] ;
["Prabir', 43, 'Kumar']
```

Objects

Objects are comma separated lists of name-value pairs. Objects are usually created by curly brackets { }.We'll get into them later, but just to give you an example here.

```javascript
{
  name: 'Prabir',
  age: 20
}
```

Variable & Identifier

Introduction to Variable

Variables are containers for pieces of data,they are used to store the information. Variables store data of any data type that can be used throughout a program. we are just going to look at the syntax for creating and re-assigning variables as well as the differences between how we declare them.

Declaring a variable

In JavaScript, we need to first declare a variable with one of three keywords:

- Var
- let
- const

JavaScript

```
var name = "Prabir"
let name = 'Prabir'
const name = "Prabir"
```

What is identifier

An identifier is a name that is given to entities like variables, functions, class, etc.

JavaScript Keywords

Keywords are reserved words that are part of the syntax in the programming language. For example,

```javascript
const a = "Prabir"
```

Here, **const** is a **keyword** that denotes that **a** is a constant.

Keywords cannot be used to name identifiers.

Example: Some of the Javascript keywords are awaited, break, continue, const, let, var, catch etc. We will learn more in details in the upcoming lecture.

Variable Naming Conventions

When naming the variables, we must consider making the names descriptive and easily understandable. This will make our program easy to read and understand in the future when we have to refactor it.

Different languages have different rules and conventions when it comes to naming things.

There are some rules that you have to follow when it comes to the formatting of your variable names.

- Variable names should begin with either a letter or an underscore or a dollar sign.
- Variable names should not begin with numbers or special characters except the underscore and dollar signs.
- Keywords are reserved words that have a specific meaning and cannot be used as variables. Keywords like if, else, and for should not be used as variable names.
- Variable names are case-sensitive. That means name and Name are different variable names. Let's take an example to understand better:

```javascript
const a = "Prabir";
const A = 25;
console.log(a); // Prabir
console.log(A); // 25

//Javascript are case-sensitive, so A and a are different variable
```

```javascript
//valid variable name

var name = "Prabir";
var Name = "Prabir";
```

```
var _name = "Prabir";
var $name = "Prabir";
```

```
JavaScript

//invalid
const 1a = 'Prabir'; // this gives an error
```

Variable names can't start with numbers. So it gives errors .

```
JavaScript
//invalid
const new = 5; // Error! new is a keyword.

// new is a keyword
```

Multi-Word Variables

To ensure consistency in naming variables adopt one of the following naming conventions in naming variables. Developers mostly prefer **camel case** naming.

This is where we start with a lowercase letter but every word after that starts with an uppercase letter.

```
JavaScript
let firstName = 'Prabir';
```

We may also see underscore like below:

```javascript
let first_name = 'Prabir';
```

There's also a **pascal case**, where the first word is also capitalised.

```javascript
let FirstName = 'Prabir';
```

Reassigning Values

If we want to change a value, we can do that here because we're using the keyword **"var"**. When it comes to changing the value directly, we can't use "const". **"const"** means the value is constant and cannot be changed. So, we have to use **"var" or "let"** if we want to change the value of a variable.

```javascript
var FirstName = 'Prabir';
FirstName = 'Alex';  // Re-assigned to Alex
```

```javascript
let x = 100;
x = 300 // Re-assigned to 300
```

Now, in some cases, you may want to simply declare a variable and not assign a value to it.

```javascript
let score;
```

In this case the score is equal to "**undefined"**. If we want to assign a value to it, we can

```javascript
let score = 10;
```

Constants:
Let's talk about const, which works in a slightly different way compared to let or var. So, if I create a name like this.

```javascript
const x = 400 ;
```

And then , we reassign that value

```javascript
x = 300 ; // results in error
```

We get an error, because we can't directly re-assign a value to a **constant** . We also can't initialize a constant as undefined. Let's take an example to understand better.

```javascript
const score1; // results in error
```

Declaring multiple values

We don't need to declare variables one by one. We can declare multiple values at the same time. With **let**, we can give them an initial value right away. However, with **const,** we have to assign a value immediately and cannot change it later.

```javascript
let a, b, c;
const d = 1, e = 2, f = 3;
```

Let or Const - Which to use ?

So how do we decide whether to use **let, const, or even var**? Well, it ultimately comes down to personal preference. What I like to do is use **const** whenever possible, unless I have a primitive value that I might need to change later on.

Let's take the example of the "score" in a game. The score will be updated throughout the game, so I would use **let** for that. In most cases, you won't need to explicitly change the value of a variable. We usually work with objects where we manipulate them without reassigning them which will be learned in the upcoming lectures.

Basic difference between let, var and const

- When we declare variables using the **var** keyword, it can be accessed within the function or globally, hence called **function scoped** (scope defines boundaries in the program where any variable can be accessed).
- When we declare a variable using the **let** keyword, it will be accessed only within the block in which it is declared, hence called **block scoped** (scope defines boundaries in the program where any variable can be accessed).
- Using const declaration, we define constants (which does not change later), and they are like block scoped.

var	**let**	**const**
Function Scoped: Variables accessible within the function block in which they are defined.	**Block Scoped:** Variables accessible within the block in which they are defined.	**Block Scoped:** Variables accessible within the block in which they are defined.
Initialized with value **undefined**	UnInitialized	UnInitialized
We can reassign values to variables.	We can reassign values to variables.	We cannot reassign values to variables.

Notes: We will discuss scope in detail in the 'scope' lesson.

Comment in JavaScript

Comments

- Programming languages have a commenting feature that allows you to document your code and provide explanations in a way that humans can understand. When multiple developers collaborate on a codebase, comments are helpful for clarifying the purpose of specific code sections.

- Additionally, comments can be used to deactivate code temporarily. This is beneficial when you want to test something without completely removing the code.

- Comments can also serve as a to-do list, helping you keep track of tasks that need to be completed in the future.

Single Line Comment

In JavaScript, single line comments are created by using two forward slashes `//`. Anything after the two forward slashes will be ignored by the JavaScript interpreter. This is useful when you want to add a comment to a single line of code.

Notes:

A JS **interpreter** is a program that executes JavaScript code line by line, producing the desired output. It is essential for running JavaScript in different environments like web browsers or servers.

```JavaScript
// This is a single line comment

console.log('Hello World');  // This is a single line comment
```

Multi Line Comments

In JavaScript, multi line comments are created by using **`/*` and `*/`**. Anything between the two symbols will be ignored by the JavaScript interpreter. This is useful when you want to add a comment to multiple lines of code.

```Unset
/*
This is a multi line comment
*/
```

THANK YOU

AIMERZ.ai
Aim.Act.Achieve

Stay updated with us!

Vishwa Mohan

Vishwa Mohan

vishwa.mohan.singh

Vishwa Mohan