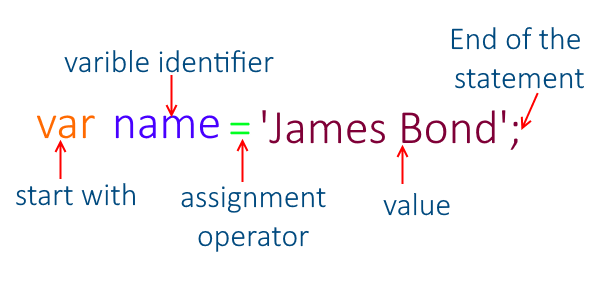
  Variables



**variables are like containers which are used to store the data and manipulate data**

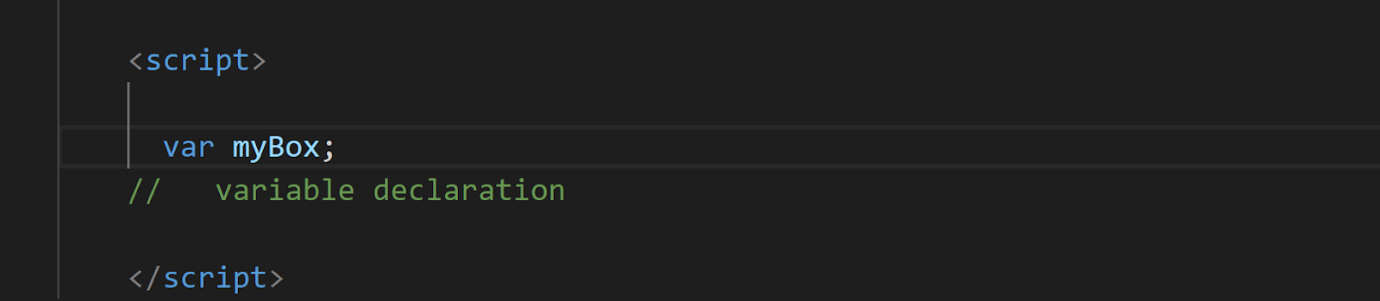
That's a perfect way to describe variables. Just like containers, variables hold values that you can use and manipulate in your code. Whether it's numbers, text, lists, or more complex data, variables provide a way to store and access information in your programs. And just like how you can label containers to know what's inside, you give variables names so you can refer to them later in your code. So, think of variables as the containers of information in your program!

Imagine you have a box. This box is like a container where you can keep different things. You can give this box a name, like "my-Box". Inside this box, you can put anything you want, like toys, books, or snacks.

**- Naming the Box:** You give your box a name, like "myBox". This name helps you remember what's inside. (Variable Declaration)

 Declaration:

   - Definition: Declaration is the process of introducing a variable to the programming environment. It tells the compiler or interpreter the name and data type of the variable.

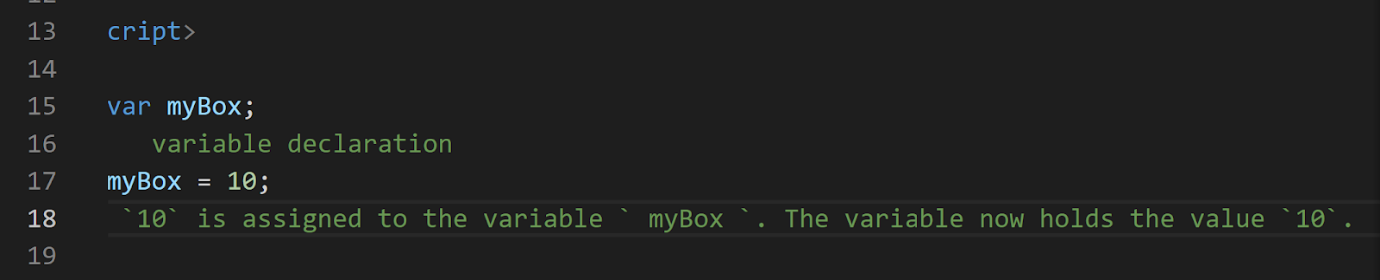


var is used to declare a variable in JavaScript. It tells the JavaScript engine to allocate memory for the variable and associate it with a particular name.

**- Adding Items**: You can put things into your box, like toys, books, or snacks. (Variable assignment)

 Assignment:

   - Definition: Assignment is the process of giving a value to a variable that has already been declared.



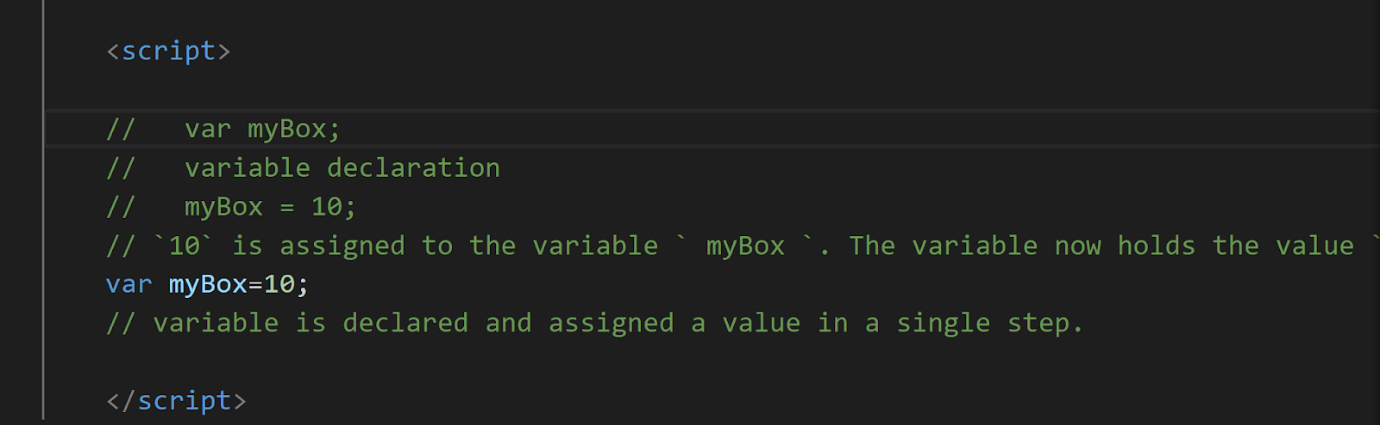
**- Changing Items:** If you want, you can take something out of the box and put something else in its place. For example, you can take out a toy and put in a book instead. (Variable Reassignment)

**- Reusing the Box**: You can use the same box to hold different things at different times. Today it might hold toys, tomorrow it might hold snacks. (Variable Reuse)

So, a variable in programming is like this box. It's a named container where you can store different types of information, change what's inside, and reuse it as needed in your code.

Initialization:

   - Definition: Initialization is a specific type of assignment that occurs when a variable is declared and assigned a value in a single step.



To summarize:

- Declaration introduces a variable to the environment.

- Assignment gives a value to a variable that has already been declared.

- Initialization combines declaration and assignment in a single step, assigning a value to a variable at the time of declaration.

Let’s dive into the concept:

Declaration:

JavaScript Variables can be declared in 4 ways:

* Automatically
* Using var
* Using let
* Using const

Here we will learn about

 Automatically , var

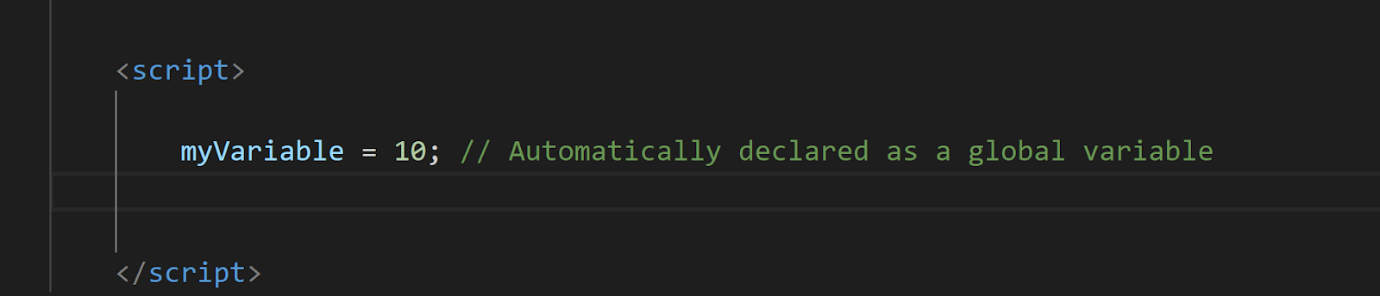
In the further classes we will learn about let const and var in deeper, because these need some basic understanding of blocks , functions  etc…

Automatically:

Variables in JavaScript can be automatically declared when they are first assigned a value without explicitly using a declaration keyword (var, let, const).

When you assign a value to an identifier that hasn't been declared previously, JavaScript automatically creates a global variable (if not in strict mode) or throws a ReferenceError (in strict mode).

This behavior is generally discouraged as it can lead to accidental creation of global variables and make code harder to understand and maintain



Using var:

The var keyword is traditionally used to declare variables in JavaScript.

Variables declared with var have function scope, meaning they are accessible within the function in which they are declared or globally if declared outside of any function.

Variables declared with var can be redeclared and reassigned..



JavaScript Identifiers

All JavaScript variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

**Names can contain letters, digits, underscores, and dollar signs.**

**Names must begin with a letter.**

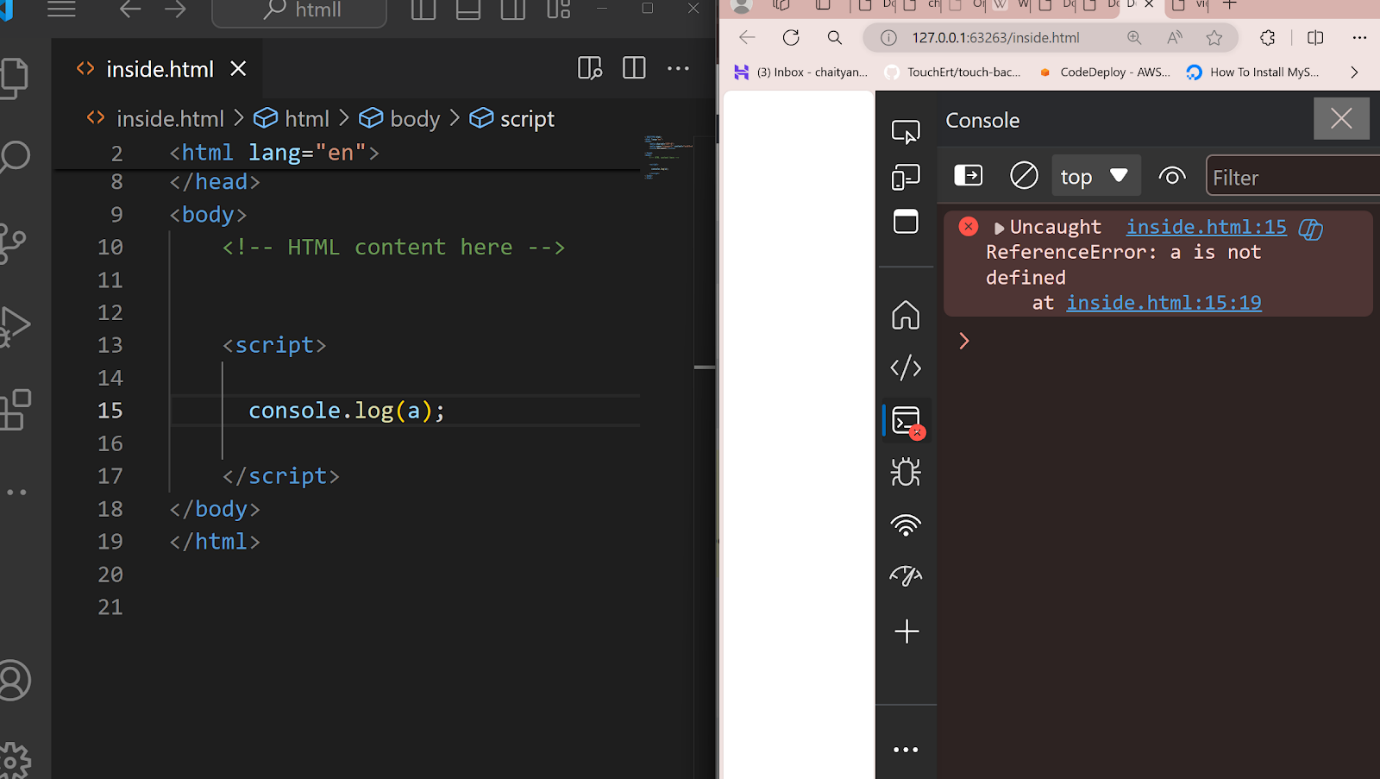
**Names can also begin with $ and \_ (but we will not use it in this tutorial).**

**Names are case sensitive (y and Y are different variables).**

**Reserved words (like JavaScript keywords) cannot be used as names.**

Without declaring a variable

In JavaScript, if you attempt to use a variable without declaring it first, you'll encounter a "not defined" error. This is known as a ReferenceError.



In this example, `a ` is being used without being declared first. JavaScript doesn't automatically create variables if they're referenced before declaration, unlike some other programming languages. This behavior helps catch potential errors in your code and encourages good programming practices like declaring variables before using them.

In JavaScript, if you declare a variable but don't assign it a value, its default value will be **undefined**



You will learn more about undefined in datatypes chapter.

Console.log ---let’s try to understand this:

In JavaScript, `console.log()` is a function used to output information to the console. The console is a built-in tool available in most web browsers' developer tools, which allows developers to debug their JavaScript code by inspecting values, logging messages, and viewing errors.

Debugging with `console.log()` is a common practice in JavaScript development. Here's how you can use it effectively for debugging purposes:

1. Inspecting Variable Values:

   - Use `console.log()` to print the current value of variables at different points in your code.

   - This helps you verify whether variables hold the expected values and track how they change during program execution.

2. Logging Control Flow:

   - Insert `console.log()` statements within loops, conditionals, or function calls to trace the flow of execution through your code.

   - This allows you to understand which parts of your code are being executed and in what order.

3. Identifying Errors:

   - Use `console.log()` to print error messages or diagnostic information when unexpected behaviour occurs.

   - This helps you identify the source of errors and understand what went wrong in your code.

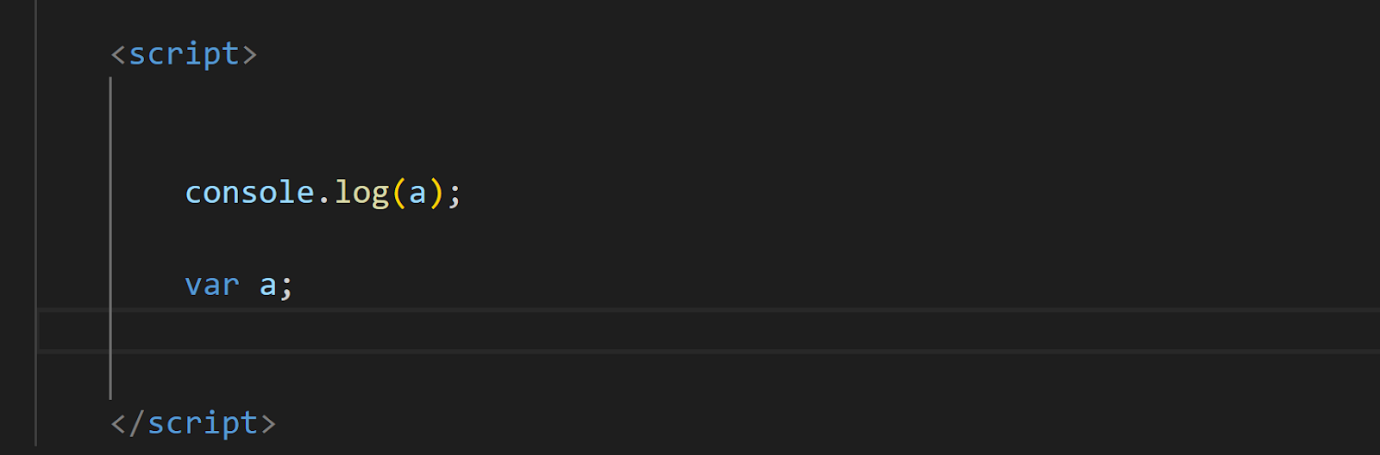
5. Performance Profiling:

   - Use `console.time()` and `console.timeEnd()` to measure the execution time of specific code blocks.

   - This helps you identify performance bottlenecks and optimize critical parts of your code.

By strategically placing `console.log()` statements throughout your code, you can gain insights into its behaviour, diagnose issues, and improve the overall quality of your JavaScript applications.

Let’ s try the snippet :



We will think that the answer for the above snippet is

Error – a is not defined

If we think like this then its good , we are in the right path of the learning journey

But javascript working has some other concept like

when you reference a variable before it's declared, you won't get a "not defined" error. Instead, due to a behaviour called "hoisting," the variable declaration is moved to the top of its containing scope during the compilation phase. This means that the variable is effectively declared before any code is executed, even if its initialization happens later in the code.

In this case:

1. `var a;` declares the variable `a`, but it doesn't assign any value to it.

2. When `console.log(a)` is encountered, `a` is declared due to hoisting, but it's not yet initialized, so its value is `undefined`.

3. Therefore, when `console.log(a)` is executed, it outputs `undefined` to the console.

So we will get the answer as – undefined

**JavaScript hoisting** is a mechanism where variable and function declarations are moved to the top of their containing scope during the compilation phase. JavaScript hoisting is a important concept for understanding how variables and functions are processed during code execution. This guide will delve into the features, nuances, and implications of hoisting in JavaScript, covering key topics such as variable declaration, function scoping, and the differences between var, let, and const declarations