

1.Var vs let vs const

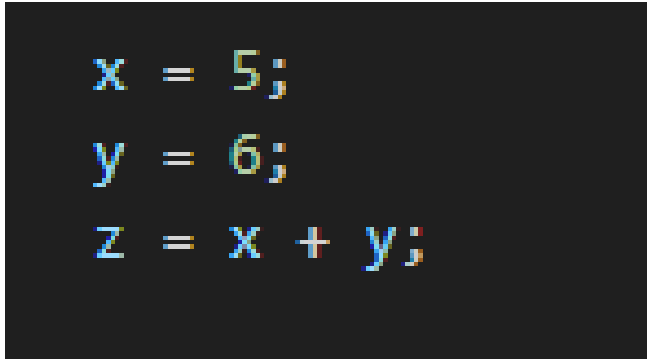
Var

Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

They are automatically declared when first used:



```
x = 5;  
y = 6;  
z = x + y;
```

The var keyword was used in all JavaScript code from 1995 to 2015.

The var keyword should only be used in code written for older browsers.

```
var x = 5;  
var y = 6;  
var z = x + y;
```

- Variables declared by var are available throughout the function in which they're declared.

A var variable will be available throughout the function body in which it is defined, no matter how deeply nested its definition.

The behavior of var can be useful in some cases, but is quite different from other programming languages, and can cause difficult-to-resolve bugs.

In JavaScript, variables are used to store data values. The var keyword is used to declare a variable and assign a value to it.

JavaScript 'var' Definition

The var keyword in JavaScript is used to declare a variable. It's one of the ways to store data values, and its usage has several characteristics that distinguish it from other variable declaration keywords like let and const.

Syntax of JavaScript 'var'

The var keyword in JavaScript is used for declaring a variable, potentially initializing it to a value. The syntax is as follows:

1. Declaration

```
var variableName;
```

2. Initialization

```
var variableName = value;
```

Where,

- variableName: Name of the variable.
- value: Initial value assigned to the variable.

Let

The let keyword was introduced in ES6(2015)

Variables declared with let have Block Scope

Variables declared with let must be Declared before use

Variables declared with let cannot be Redeclared in the same scope

Cannot be Redeclared

Variables defined with let can not be redeclared.

You can not accidentally redeclare a variable declared with let.

```
let x = 8;  
let x = 9;  
// this is wrong
```

Redeclaring a variable inside a block will not redeclare the variable outside the block:

```
let x = 10;  
// here x is 10  
{  
  let x = 5;  
  //here x is 5  
}  
// here x is 10
```

Const

The keyword `const` is a signal that the identifier won't be reassigned. A constant can be declared with the `const` keyword. Once a constant is declared, it cannot be reassigned a different value. Constants are block-scoped, much like variables defined using the `let` keyword.

```
const PI = 3.14;  
console.log(PI);
```

Why `const` is better

But why `const` is better? In my opinion it makes the code easier to reason about, basically because since it cannot be reassigned you have less scenarios, side effects or edge cases to reason about.

ECMAScript 2015 (ES6)

In 2015, JavaScript introduced an important new keyword: `const`.

Cannot be Reassigned

An array declared with `const` cannot be reassigned:

```
const cars = ["Saab", "Volvo", "BMW"];  
cars = ["Toyota", "Volvo", "Audi"];  
// ERROR
```

Arrays are Not Constants

The keyword `const` is a little misleading.

It does NOT define a constant array. It defines a constant reference to an array.

Because of this, we can still change the elements of a constant array.

Elements Can be Reassigned

You can change the elements of a constant array

```
// You can create a constant array:  
const cars = ["Saab", "Volvo", "BMW"];  
  
// You can change an element:  
cars[0] = "Toyota";  
  
// You can add an element:  
cars.push("Audi");
```


2.Alert vs Prompt vs Confirm

The alert() method displays an alert box with a message and an OK button.

The alert() method is used when you want information to come through to the user.

Do not overuse this method. It prevents the user from accessing other parts of the page until the alert box is closed.

```
alert("Hello \n How are you?");
```

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Prompt

displays a message box to the user and has only two options “**OK**” or “**Cancel**”. The dialogue box would wait for the user to enter any input and loads the webpage afterwards. If the user does not wish to respond to the pop-up/dialogue box, press the Cancel option to cancel it. Generally, it takes a string as input.

```
prompt("Hello Fedia");
```

- **message** is a string of text to display to the user. It can be omitted if there is nothing to show in the prompt window i.e. it is optional.
- **default** is a string containing the default value displayed in the text input field. It is also optional.

Return value:

The `prompt ()` method returns the text entered by the user in the input field as a string. If the user cancels the dialog box, `null` is returned.

Confirm

represents the browser window or tab. The primary purpose of confirm in javascript is to prompt the user with a confirmation dialog having two options to choose from: OK or Cancel.

Work Flow:

- Invoking confirm() displays a modal dialog box on top of the current page.
- The dialog presents two buttons: OK and Cancel, along with a developer-specified message.
- The user interacts by selecting an option.
- Choosing OK results in the method returning true; selecting Cancel or closing the dialog without a choice returns false.

```
window.confirm('Do you want to exit?');
```

The `window.confirm()` method can be written without the `window` prefix.

Conclusion

Alert

It shows a message and waits for the user to press "OK".

Prompt

The function prompt accepts two arguments.

It shows a modal window with a text message, an input field for the visitor, and the buttons OK/Cancel.

confirm

shows a message and waits for the user to press "OK" or "Cancel". It returns true for OK and false for Cancel