* **JS Variables**
* **Primitives vs Reference types**
  + Primitive types = Number, String, Boolean, Undefined, Null, Symbol, BigInt.
  + Reference types = Object literal, arrays, functions, … .
  + Graphical user interface

    Description automatically generated
  + To learn how the different types work take the following code as an example

A screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated

* The values at an address are immutable so when the value of age get changed (line no 179) , it creates a new memory location with a new value.
* But in case of reference types, the value is the address of Object in heap storage, and when friend.age changes the value, it gets changes in the heap.
* Graphical user interface, text, application

  Description automatically generated
* **General**
  + You can declare many variables in one statement. Start the statement with var and separate the variables by comma.

Ex var person = "John Doe", carName = "Volvo", price = 200;

OR

var person = "John Doe",

carName = "Volvo",

price = 200;

* + If you **re-declare a JavaScript variable, it will not lose its value**.

Ex. The variable carName will still have the value "Volvo" after the execution of these statements.

var carName = "Volvo";

var carName;

* + Always remember var x = "5" + 2 + 3; will give x=523, execution from **left to right**. OR var x = 2 + 3 + "5"; will give x=55.
* **Let, const and var**
  + **Let and const**
    - Let is used as a variable and const to hold constant value.
    - On changing the value of const it will give an error
    - Let and const are block scoped and var is function scoped
    - Example:

For var => for let and const=>

function(abc) function(abc)

{ {

If() if()

{var a= 5} { let a=10,const b=20}

console.log(a) //5 console.log(a,b) //error

} }

* + - Let and const declaration are hoisted but are not declared with anything.
    - Previously JS only have global scope and function scope, but ES6 introduced block scope too which are reflected up in let and const.
    - As we know redeclaration of var anywhere is allowed, but redeclaring a let variable with let, in the same scope, or in the same block, is not allowed. Redeclaring a variable with let, in another scope, or in another block, is allowed.
    - Let and const variables are not attached to window object.
  + **var**
    - var is hoisted and is declared with undefined.
    - When a variable is created using var , a property of same name is created on global object.
    - var variables are only function scoped and global scoped.
  + **Let and Const vs var (**[**https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/**](https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/) **)**
    - var declarations are globally scoped or function scoped while let and const are block scoped.
    - var variables can be updated and re-declared within its scope; let variables can be updated but not re-declared; const variables can neither be updated nor re-declared.
    - They are all hoisted to the top of their scope. But while var variables are initialized with undefined, let and const variables are not initialized.
    - While var and let can be declared without being initialized, const must be initialized during declaration.
* **Undefined**
  + You declare a JavaScript variable with the var keyword and if no value assigned it will be set to **undefined.**
  + **undefined** can be explained as a placeholder which is assigned to a variable which is declared but not assigned any value , i.e., the variable is allocated memory, but no value is assigned .
  + **not defined** comes into place when variable is not assigned any memory.
  + To check for undefined variable use var === undefined or typeof var === ‘undefined’.
  + Assigning a variable to undefined is not a good practice.
* **JS Hoisting**
* In hoisting only variable declaration are hoisted to the top not the initialization.
* Example

var x = 5; // Initialize x  
  
elem = document.getElementById("demo"); // Find an element  
elem.innerHTML = x + " " + y;           // x = 5, y = undefined  
  
var y = 7; // Initialize y

* Graphical user interface, text, application

  Description automatically generated
* **Temporal Dead Zone**
* var is added to the global object whereas let and const are added to a private script object which cannot be accessed by any program before initialization which causes temporal deadzone.
* Graphical user interface, text, application

  Description automatically generated
* But once the value is assigned to a let/const variable it is then accessible.Graphical user interface, text, application

  Description automatically generated
* So temporal deadzone it the time between a let /const variable is hoisted and initialized.
* When we try to access a variable in temporal deadzone it gives reference error.
* Graphical user interface, text, website

  Description automatically generated
* **Type Conversion**
  + **Number(), parseInt(), parseFloat()** converts to a Number.
  + **String(), toString()** converts to a String.
  + **Boolean()** converts to a Boolean.
  + Getting type of a token –
    - **typeof**
      * This **operator** is used to find the data type of a JavaScript variable.
      * The data type of NaN is number
      * The data type of an array is object
      * The data type of a date is object
      * The data type of null is object
      * The data type of an undefined variable is undefined \*
      * The data type of a variable that has not been assigned a value is also undefined \*.
    - **Constructor property**
      * The constructor property returns the constructor function for all JavaScript variables.
      * You can check the constructor property to find out if an object is an Array (contains the word "Array"), Date (contains the word “Date”).
        + Method 1

myArray.constructor.toString().indexOf("Array") > -1;

* + - * + Method 2

myArray.constructor === Array

* Graphical user interface, text, application

  Description automatically generated