Javascript

Javascript is a programming language used in world wide web, It lets add interactiveness in our applications. e.g sliders, alerts, click interactions, and popups etc.

Variables:

Most of the time a javascript application needs to work with information e.g A chat application - to store information of users, messges etc.

Let and var can be used for declearance

The data of a variable can be given to another variable

The data of the variable can be overwritten but the overwritten line will not contain **let**.

Naming Variables:  
There are two limitations on variable names in JavaScript:

1.The name must contain only letters, digits, or the symbols $ and \_.

2.The first character must not be a digit.

**Camel Case** is used for large names of variables e.g testNumber1, YouTube, iPhone, eBay.

apple and APPLE are two **different** variables.  
  
Different **languages** are allowed but not recommended.  
  
Reserved names: Variables cannot be named **var, let, return and change etc.**We dont need to first declear a variable, we can just assign it the data and the variable wil be created if it does not exist. But its not considered **Good Practice.**Constants:

The difference between a constant and variable is that a constant cannot be **reasigned.**

If a programmer is sure that some data will not change then he assigns it to be a constant to declear it to everyone.

Example:

const COLOR\_RED = "#F00";

const COLOR\_GREEN = "#0F0";

const COLOR\_BLUE = "#00F";

const COLOR\_ORANGE = "#FF7F00";

// ...when we need to pick a color

let color = COLOR\_ORANGE;

alert(color); // #FF7F00

**Benifits:**

1.COLOR\_ORANGE is much easier to remember than "#FF7F00".

2.It is much easier to mistype "#FF7F00" than COLOR\_ORANGE.

3.When reading the code, COLOR\_ORANGE is much more meaningful than #FF7F00.

**Name things right:(very important)**

Talking about variables, there’s one more extremely important thing.

A variable name should have a clean, obvious meaning, describing the data that it stores.

Variable naming is one of the most important and complex skills in programming. A glance at variable names can reveal which code was written by a beginner versus an experienced developer.

In a real project, most of the time is spent modifying and extending an existing code base rather than writing something completely separate from scratch. When we return to some code after doing something else for a while, it’s much easier to find information that is well-labelled. Or, in other words, when the variables have good names.

Please spend time thinking about the right name for a variable before declaring it. Doing so will repay you handsomely.

Some good-to-follow rules are:

Use human-readable names like userName or shoppingCart.

Stay away from abbreviations or short names like a, b, and c, unless you know what you’re doing.

Make names maximally descriptive and concise. Examples of bad names are data and value. Such names say nothing. It’s only okay to use them if the context of the code makes it exceptionally obvious which data or value the variable is referencing.

Agree on terms within your team and in your mind. If a site visitor is called a “user” then we should name related variables currentUser or newUser instead of currentVisitor or newManInTown.

**Hoisting:**

Hoisitng in javascript is when it moves the function to the top of the file before it used, javascript only moves func and var to the top of the file and **not const.**

**Scopes**

**Global Scope:**

Global scope values of inputs are those inputs which can be used by all functions and any children of them wrapped in parenthesis or anything  
  
**Local Scope:**

Local scope are the values of inputs which can only be used by functions using them inside the paresnthesis in which they already are. Local scope variables can not be used globally.

**Function scope and Block scope ({})**  
  
**If there are is a value given to a variable in global scope and in local scope than the function will use the local scope value rather that global scope value.**

**var is function scoped while const and let are block scoped (to further understand open funtions.js in Js learning and read revision.)**

**Tips:**1.Avoid using var, stick to let and const.  
2.Use const if you know you are not going to reassign a value.

3.Use let if you know you are going to reassign a value.

<https://youtu.be/_E96W6ivHng?si=eDC040VYH5MmA6f6>

Data Types

var x = "Hello World"; ===> **String**var a= 25; ===> **Number**  
var b= true; ===> **boolean**

var c= ["HTML, CSS, JAVASCRIPT."]; ===> **Array**

var d= {first="Jane", last="Doe"}; ====> **Object**var e= null; ===> **Null**  
var f; ====> **Undefined**

**Objects:**

Object is a collection of related properties and/or methods

Can represent real world objects (people, products, places) object = {key:value, function()}

**Coversions(implicit- done by the developer) and Coercions(explicit - done by javascript)**

**String Conversions:**

A number can be converted to a string by the following method:

let value = true;

value = String(value);

**Number Conversions:**Two strings can be converted to a number by the following method:

let x = ("6"/ "2");

6 and 2 are strings in this case but after the function has been performed the answer will obviously be a number.  
  
**Boolean Conversions:**The number 0 and 1 can be used to represent true and false values in a boolean, **0 represents False while 1 represents True**. However if 0 is used as a string then it will also represent true:

alert( Boolean(1));

alert( Boolean("0"));

alert( Boolean(0));

**Coercions:  
Addition:**

Coercion is when a number in a process is converted to string forcefully to give some output when one value is string and other is a number:  
  
let a = 2;

let b = '23';

c = a + b;

alert(`Answer: ${c}`)

In the above a is a string and the answer c will not be **25 but 223** because it is not a number but string and **a** is forcefully converted to a string. But if we add parseInt to b (which converts a string to a number) the answer will be the expected one:

let a = 2;

let b = '23';

b = parseInt(b, 10)

c = a + b;

alert(`Answer: ${c}`)

**Important- Javascript knows that addition is function which can be done in strings but if we performed a function like multiplication/division than the b variable would have been converted to a number and the answer would also been a number.**