Eating, wiring

(javascript)

Bricks, paing

(CSS)

House Plan

(HTML)

house

Use of Firefox inspect tools

To clear the console clear() and enter

**Javascript code to change background at every click**

const htmlBody = document.querySelector('body');

const randomClickFunction = function(){

const colors = ["green", "orange", "grey", "red", "blue" , "green"];

const randomIndex = Math.floor(Math.random() \* colors.length);

const randomColor = colors[randomIndex];

htmlBody.style.backgroundColor = randomColor;

console.log('The user clicked and the color is set to ' + randomColor);

}

htmlBody.onclick = randomClickFunction

Simple button modifications and application of simple javascript

<!DOCTYPE html>

<html lang="en">

<head>

 <meta charset="UTF-8">

 <style>

  button{

   background-color: transparent;

   border: 1px solid navy;

   padding: 20px;

   font-size: 1.4rem;

   border-radius: 10px;

  }

  button:hover{

   background-color: navy;

   border: none;

   color: white;

  }

 </style>

 <meta http-equiv="X-UA-Compatible" content="IE=edge">

 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <title>Example</title>

</head>

<body>

 <button>CLICK ME</button>

 <div class="container"></div>

 <script>

  function onClickEvent(){

   const el = document.createElement('p');

   el.innerText = 'Clicked the button';

   document.querySelector('.container').appendChild(el);

  }

  document.querySelector('button').onclick = onClickEvent;

 </script>

</body>

</html>

**JavaScript Variables and Data Types**

const myArray = ["10", "20", "30", "40", "50"];

myArray

Array(5) [ "10", "20", "30", "40", "50" ]

myArray.push("70")

6

myArray

Array(6) [ "10", "20", "30", "40", "50", "70" ]

myArray.reverse()

Array(6) [ "70", "50", "40", "30", "20", "10" ]

Use console.log() to print something in console

One line comment //

Multiline comment /\* \*/

**Variables and datatypes**

Variable declaration and assignment

const firstVar = 'Amitha Shehan';

const secondVar = 50;

const thirdVar = {firstProperty: 'hello world'};

console.log(firstVar)

console.log(secondVar)

console.log(thirdVar)

declaring variable first and then assigning value to it

let newVar;

newVar = 50;

console.log(newVar)

**Accepted declaration and assigning**

Const var1 = 50; //cannot be reassigned/redeclared

Let var2 = 90; //used for reassignment

Var var3 = 50; // this is a legacy method. Do not recommend using this. Var can be redeclared

Use const when we have values not to change as the program runs

Use let when we assign new values to variables in the long run

Use capslock when defining const variables

const TAX\_RATE = 0.08;

**Use of const and function**

const variable1 = 10;

//const variable2 = variable1;

const variable3 = "Amitha";

const variable4 = {variableType:"object",variableValue:"somevalue"};

const variable5 = (function(){

return "Hello , my name is ";})();

const variable6 = variable5 + variable3;

**Boolean variables**

Const variable7 = false

**To get to know the datatype of a declared variable**

typeof variable1;

javascript is a dynamically typed language which means we do not need to define the variable type (number, string, Boolean) when we declare a variable.

Converting a string number and add it to a number

Both single quotes and double quotes can be used in string declaration. It is recommended to use single quotes

When assigning values to a number do not use commas

**Arrays**

Varieties of data can be included into arrays’

const array2 = [10, 'string 1', {prop:'sdkfjd'}, [10,20,330]];

console.log(array2[0]);

console.log(array2[1]);

console.log(array2[2]);

console.log(array2[3]);

console.log(array2[3][2]);

**Objects**

const objectVariable = {prop1: 20, prop2:50};

to access a certain property

objectVariable.prop2;

or

objectVariable[‘prop1’]

Nested object declaration and access to them using dot notation

const nestedVariable = {

layer1: {

layer2:{

layer3:{

targetValue: 20}}}}

nestedVariable.layer1.layer2.layer3.targetValue

**Functions**

Defining a function

const functionContainerVariable = function(){

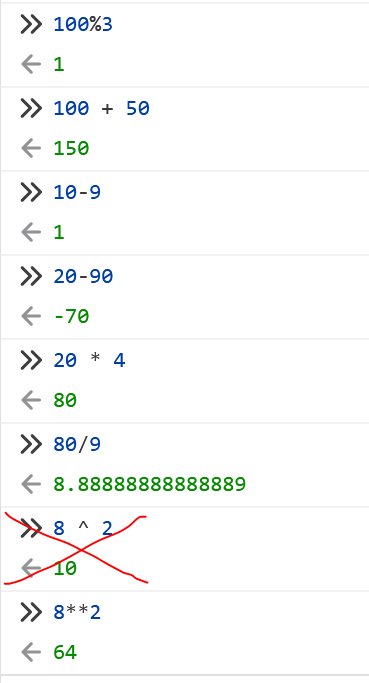
return 20;

}

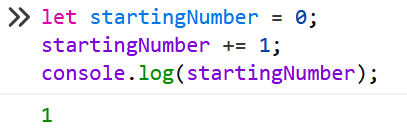
Calling a function

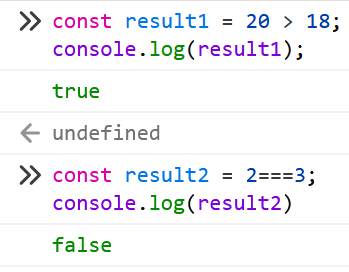
functionContainerVariable()

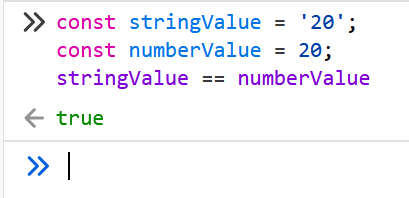
**Javascript operators**

Arithmetic operators

Increment and decrement operator does not work on const variables, but works on let.

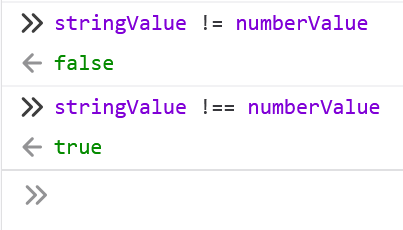


**Comparison operators**



JS converts both variables in to one type and then compares them, we get false if we try

stringValue === numberValue;

double equal does not compare data type, but triple equal compares both data type and value

If we compare two arrays, objects with different names but content is same, when we == or === them answer is false

We cannot use equal operators == or === on arrays and objects.

To do so we can use library called lodash.

**Ternary operator**

const result = 20 === 20 ? 'values match': 'values does not match';

Ternary operator is same as if condition

let resultVariable;

if(20 == 20){

resultVariable = 'values match';

}

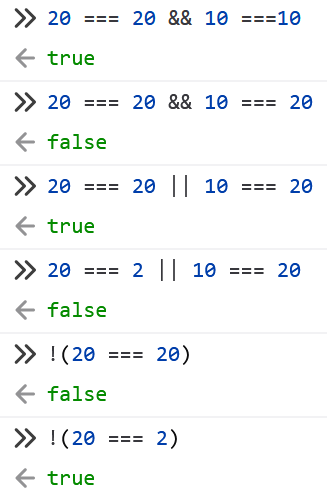
else{

resultVariable = 'values do not match';

}

console.log(resultVariable);

**Logical Operators**



**Conditions**

**If condition if(){} else{}**

if('some string' === 'another string'){

console.log('Strings are equal');

}else{

console.log('strings are not equal');

}

**Switch-case**

const colors = ["red", "green", "blue", "orange"];

const randomIndex = Math.floor(Math.random() \* colors.length);

const randomColor = colors[randomIndex];

console.log(randomColor);

switch (randomColor) {

case "orange":

console.log("color is orange");

break;

case "blue":

console.log("color is blue");

break;

case "green":

console.log("color is green");

break;

case "red":

console.log("color is red");

break;

default:

console.log('no color found');

}

**Looops**

for (let i = 0; i < 10; i++){

console.log(i);

}

To print only numbers in a array

for (let i = 0; i < array.length; i++){

if (typeof array[i] == 'number') {

console.log(array[i]);

}

}

Arrays, Objects and loops in one code

const myArray1 = [

{

name: "amitha",

age: 29,

},

{

name: "kasun",

age: 19,

},

{

name: "John",

age: 24,

},

{

name: "kane",

age: 89,

},

];

for (let i = 0; i < myArray1.length; i++){

const theName = myArray1[i].name;

const theAge = myArray1[i].age;

console.log(theName + ' is ' + theAge + ' years old')

}

**Functions**

Function myFunction(){content}

After defining a function we need to execute, call or invoke the function

function myFunction() {

console.log('hello world this is my first javascript function');

}

myFunction();

The way to call the function just after defining the function (immediately invoked function) (declaring and execution at the same time)

**(function myFunction(){content})()**

(function myOtherFunction() {

console.log('declared and executed in same line ');

})();

**Parameters and Arguments**

function myFunction1(param1, param2) {

console.log(param1);

console.log(param2);

}

myFunction1('argument 1', 'argument 2');

**Another way to declare function in JS**

const anotherFunction = function () {

console.log('another function');

}

anotherFunction();

This is an anonymous function. The function does not have a name, but it is assigned to a variable.

**Arrow Function**

const arrowFunction = () => {

console.log('I am arrow function');

}

arrowFunction();

**Return statement in functions**

const myFunction = () => {

const a = 15;

return a;

}

console.log(myFunction());

we can return Boolean values and functions as well.

We should not put any line of code after return statement

It’s not going to be read after return statement

const myNumber = 50; // defining a global variable

let myFunction = () => {

if (myNumber < 25) { //accessing global variable

return 'the number is less than 25';

}

return 'the number is greater than 25';

}

console.log(myFunction());

**Assigning a variable to a function**

function myFunction() {

return 50;

}

console.log(myFunction());

const aliasVariable = myFunction;

console.log(aliasVariable() + 20);

**Assigning a variable to an object**

function myFunction() {

return 50;

const myObj = {

prop1: 20,

prop2: myFunction

}

console.log(myObj.prop2());

When a function is bound to an object it is called a method.

const myName = 'Amiths';

console.log(myName.replace('s', 'a'));

console.log(myName);

const myCorrectName = myName.replace('s', 'a');

console.log(myCorrectName);

We cannot change myName variable, but instead we can use replace method to assign a new variable.

We can call methods one upon another

const str = 'Hello World';

console.log(str.toUpperCase().split(" ").indexOf('WORLD'));