ECMA Script (Modern JS)



Features of ECMA Script 2015 also known as ES6.  


**var Vs let Vs const:-**

**var:**

* Using var keyword for defining variable name and storing data/value in it. We can further Update or Change it’s value.  
  **Ex:** Output: Chandan Kumar  
  var name = 'Chandan ';

document.write(name);

name = 'Kumar';

document.write(name);

* var variable has ‘Function scope’ means if we define/declare var variable inside a function scope as well as block scope, then we can access its value anywhere throughout the function but not outside of the function.

Ex:  
function biodata(){ //Function Scope Start

var fName = 'Chandan';

document.write('<br>'+fName+'<br>');

if(true){ //Block Scope Start

var lName = 'Kumar';

document.write('<br>'+fName);

document.write('<br>'+lName+'<br>');

} //Block Scope End

document.write('<br>'+fName);

document.write('<br>'+lName);

} //Function Scope End

biodata();

**let:**

* Using ‘let’ keyword for defining variable name and storing data/value in it. We can further Update or Change it’s value.  
  **Ex:** Output: Chandan Kumar  
  let name = 'Chandan ';

document.write(name);

name = 'Kumar';

document.write(name);

* let variable has ‘**Block scope**’ means if we define/declare let variable inside the function scope, then we can access its value anywhere throughout the function.   
  But If it’s declare inside the block scope of that funciton then we can access its value only inside of that block scope.  
  Ex:  
  function biodata(){ //Function Scope Start

let fName = 'Chandan';

document.write('<br>'+fName+'<br>');

if(true){ //Block Scope Start

let lName = 'Kumar';

document.write('<br>'+fName);

document.write('<br>'+lName+'<br>');

} //Block Scope End

document.write('<br>'+fName);

document.write('<br>'+lName);

} //Function Scope End

biodata();

**const:**

* Using ‘const’ keyword for defining variable name and storing data/value in it. We cannot further Update or Change it’s value.  
  **Ex:**   
  const name = 'Chandan ';

document.write(name);

name = 'Kumar';

document.write(name);  
  
Output:- Error (Assignment to constant variable.)

* const variable has ‘**Block scope**’ means if we define/declare const variable inside the function scope, then we can access its value anywhere throughout the function.   
  But If it’s declare inside the block scope of that function then we can access its value only inside of that block scope.  
  Ex:  
  function biodata(){ //Function Scope Start

const fName = 'Chandan';

document.write('<br>'+fName+'<br>');

if(true){ //Block Scope Start

const lName = 'Kumar';

document.write('<br>'+fName);

document.write('<br>'+lName+'<br>');

} //Block Scope End

document.write('<br>'+fName);

document.write('<br>'+lName);

} //Function Scope End

biodata();

**Template Literals (Template Strings):**

JavaScript Program to print table for given number (8)?

**Solution:-**

*//* ***Old JavaScript*** *(Without using Template Literals)*

function multiply(table){

for(var i=1;i<=10;i++){

show ='<br>'+ table + " x " + i + " = " + table\*i;

document.write(show);

}

}

multiply(8);

*// Modern JavaScript (Using* ***Template Literals****)*

function multiply(table){

for(var i=1;i<=10;i++){

show =`<br> ${table} x ${i} = ${table \* i}`;

document.write(show);

}

}

multiply(8);

**Default Parameters:**

Default function parameters allow named parameters to be initialized with default values if no value o undefined is passed.  
Ex:  
var total = function(a, b = 50 ){ //Default parameter ‘b’ assigned default argument

return (a+b);

}

add = total(50);

add2 = total(100);

document.write(add ,'<br>', add2);

**Fat Arrow Function:**

const total = ()=> `sum of two nos. is ${(a=50) + (b=50)}`;

document.write(total());