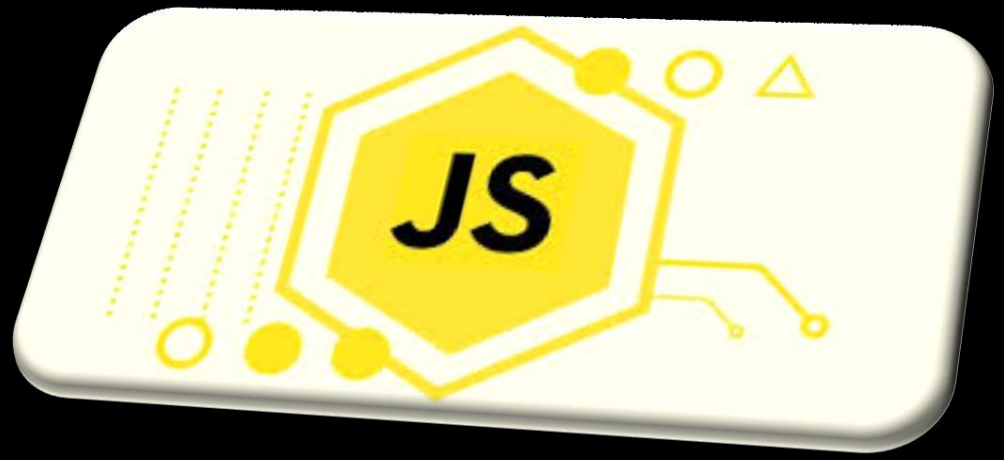
JS Tutorial



- ✓ JS Introduction
- ✓ JS Implementation
- ✓ HTML Tags in Javascript
- ✓ JS Comments
- ✓ JS Variables
- ✓ JS Variables (Let & Const)
- ✓ JS Data Types
- ✓ JS Arithmetic Operators
- ✓ JS Assignment Operators
- ✓ JS with Google Chrome Console
- ✓ JS Comparison Operators
- ✓ JS If Statement
- ✓ JS Logical Operators
- ✓ JS If Else Statement
- ✓ JS If Else If Statement
- ✓ JS Conditional Ternary Operator
- ✓ JS Switch Case
- ✓ JS Alert Box
- ✓ JS Confirm Box
- ✓ JS Prompt Box
- ✓ JS Functions
- ✓ JS Functions with Parameters
- ✓ JS Functions with Return Value
- ✓ JS Global & Local Variable
- ✓ JS Events
- ✓ JS While Loop
- ✓ JS Do While Loop
- ✓ JS For Loop
- ✓ JS Break & Continue Statement
- ✓ JS Even & Odd with Loops
- ✓ JS Nested Loop
- ✓ JavaScript Nested Loop - II

- ✓ JS Arrays
- ✓ JS Create Arrays Method - II
- ✓ JS Multidimensional Arrays
- ✓ JS Modify & Delete Array
- ✓ JS Array Sort & Reverse
- ✓ JS Array Pop & Push
- ✓ JS Array Shift & Unshift
- ✓ JS Array Concat & Join
- ✓ JS Array Slice & Splice
- ✓ JS isArray
- ✓ JS Array index
- ✓ JS Array Includes
- ✓ JS Array Some & Every
- ✓ JS Array find & findIndex
- ✓ JS Array Filter
- ✓ JS Array Methods
- ✓ JS forEach Loop
- ✓ JS Objects
- ✓ JS Objects - II
- ✓ JS Array of Objects
- ✓ JS Const Variable with Array & Objects
- ✓ JS For in Loop
- ✓ JS Map Method
- ✓ JS String Methods
- ✓ JS String Methods - II
- ✓ JS Number Methods
- ✓ JS Math Methods
- ✓ JS Date Methods

JS HTML DOM

- ✓ JS DOM Introduction
- ✓ JS DOM Targeting Methods
- ✓ JS DOM Get & Set Value Methods
- ✓ JS DOM querySelectors
- ✓ JS DOM CSS Styling Methods
- ✓ JS addEventListener Method
- ✓ JS classList Methods
- ✓ JS parent Method
- ✓ JS Children Methods
- ✓ JS firstChild & lastChild Method
- ✓ JS nextSibling & prevSibling Method
- ✓ JS create & TextNode
- ✓ JS appendChild & insertBefore
- ✓ JS insert
- ✓ JS replaceChild & removeChild
- ✓ JS cloneNode
- ✓ JS Contains
- ✓ JS has
- ✓ JS isEqualNode

JS FORM EVENTS

- ✓ JS Form Events - I
- ✓ JS Form Events - II

JS EFFECTS

- ✓ JS Interval
- ✓ JS Timeout

JS BROWSER BOM

- ✓ JS BOM Introduction
- ✓ JS Window Height & Width
- ✓ JS Window Open & Close
- ✓ JS Window move
- ✓ JS resize
- ✓ JS scroll
- ✓ JS Location Object
- ✓ JS History O

Js implementation

Js implementation have a 02 types :

1. In <head> tag :

```
<!DOCTYPE html>
<html lang="en">
<head>
<script>
 document.write("Hello from Yahoo Baba");
</script>
Or.
<script src="js/test.js"></script>
</head>
<body>
   <h1>Hello</h1>
</body>
</html>
```

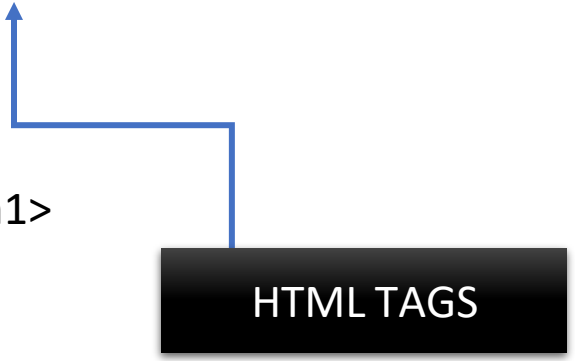
2. At end of the <body> tag (most preferable)

```
<!DOCTYPE html>
<html lang="en">
<head>
</head>
<body>
   <h1>Hello</h1>
   <h1>Hello</h1>
    <h1>Hello</h1>
<script>
 document.write("Hello from Yahoo Baba");
</script>
Or.
<script src="js/test.js"></script>
</body>
</html>
```

```
// test.js file (external file)

document.write("Hey Hello Again");
```

How to add HTML Tags :

```
<!DOCTYPE html>
<html lang="en">
<head>
<script>
   document.write("<i><b>Hello from Yahoo Baba<b></i>");
</script>
</head>
<body>
   <h1>Yahoo Baba</h1>
</body>
</html
```



Js Comments :

1. Single line comments (//)
2. Multiple line comments (/* */)

```
<!DOCTYPE html>
<html lang="en">
<head>
<script>
// this is the single line comments.

/* this is
   the multiple line
   comments.
*/
</script>
</head>
<body>
   <h1>Yahoo Baba</h1>
</body>
</html>
```

Types of variables of javascript :
1. var 2. let 3. const

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>JavaScript</title>
   <script>
1.
      var x = "Hello";
      var y = "World";
      document.write(x + y);
2.
      var x = "Hello";
      var x = “world"; //(valid)
   </script>
</head>
<body>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>JavaScript</title>
   <script>
1.
      let x = "Hello";
      let y = "World";
      document.write(x + y);
2.
      var x = "Hello";
      x = “world"; //(valid)
3.
      let x = "Hello";
      let x = “world"; //(Not valid)
   </script>
</head>
<body>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>JavaScript</title>
<script>
1.
      const x = "Hello";
      const y = "World";
      document.write(x + y);
2.
      const x = "Hello";
      x = “world"; //(Not valid)
3.
      const x = "Hello";
      const x = “world"; //(Not valid)
4.
/*Not Valid .Always Initialize During Declaration. */
       const x;
       x = "Hello";
</script>
</head>
<body>
</body>
</html>
```

Data types in Java Script :

```
var x = "Hello World";      → String
var x = 25;                 → Number
var x = true;               → Boolean
var x = ["HTML","CSS","JS"]; → Array
var x = {first:"Jane", last:"Doe"}; → Object
var x = null;               → Null
var x;                      → Undefined
```

typeof *variable_name* → *datatype of variable*

```
let X = 'Hello';
let X = "Hello";
```

→ string

```
let X = undefined;
let X;
```

→ undefined

```
let X = ["HTML","CSS","JS"];
let X = {
        first:"Jane",
        last:"Doe"
};
```

comma

Display : In both case datatype of X is Object

Actual : first one is array & second one is object

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>JavaScript</title>
  <script>
     /* String Data Type */
     var a = "Hello World";

     /* Number Data Type */
     var b = 25;

     /* Boolean Data Type */
     var c = true;

     /* Array Data Type */
     var d= ["HTML","CSS","JS"];

     /* Object Data Type */
     var x= {first:"Jane",last:"Doe"};

     /* Null Data Type */
     var y = null;

     /* undefined Data Type */
     var z;
</script>
</head>
<body>
</body>
</html>
```

Arithmetic operators :

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

This all are operators work only, When all operands have datatype is **Number.**

```
let x = 2**5;
```

Output :
X = 32

Assignment operators :

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Google chrome Console:

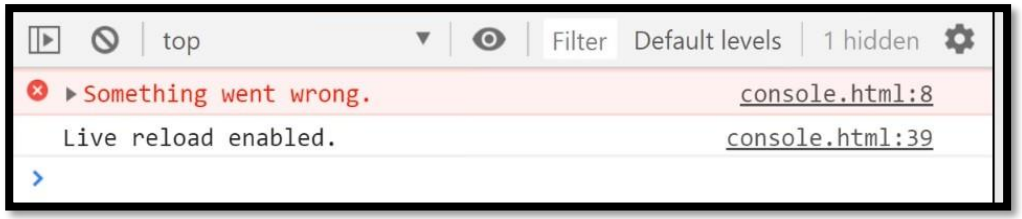
Console generally used for ,

1. To see the errors in Js
2. for testing some operations in Js

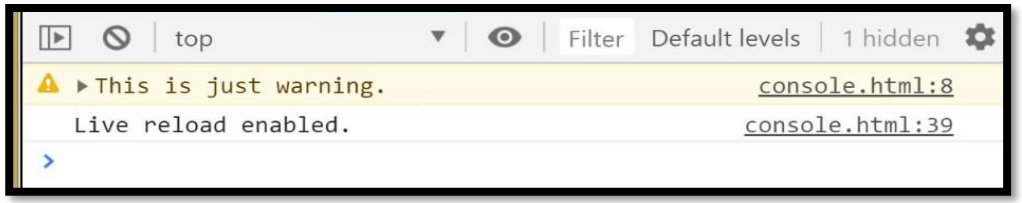
   Ex :

   - check variable value ,
   - check console.log() value ,
   - To see object hierarchy, etc.

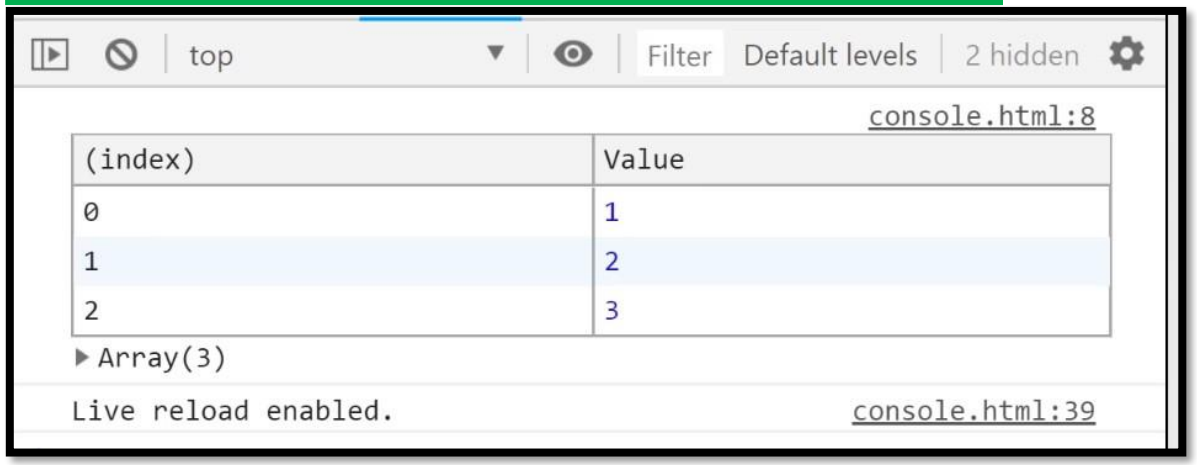
```
console.error("something went wrong");
```




```
console.warn("this is just warning");
```

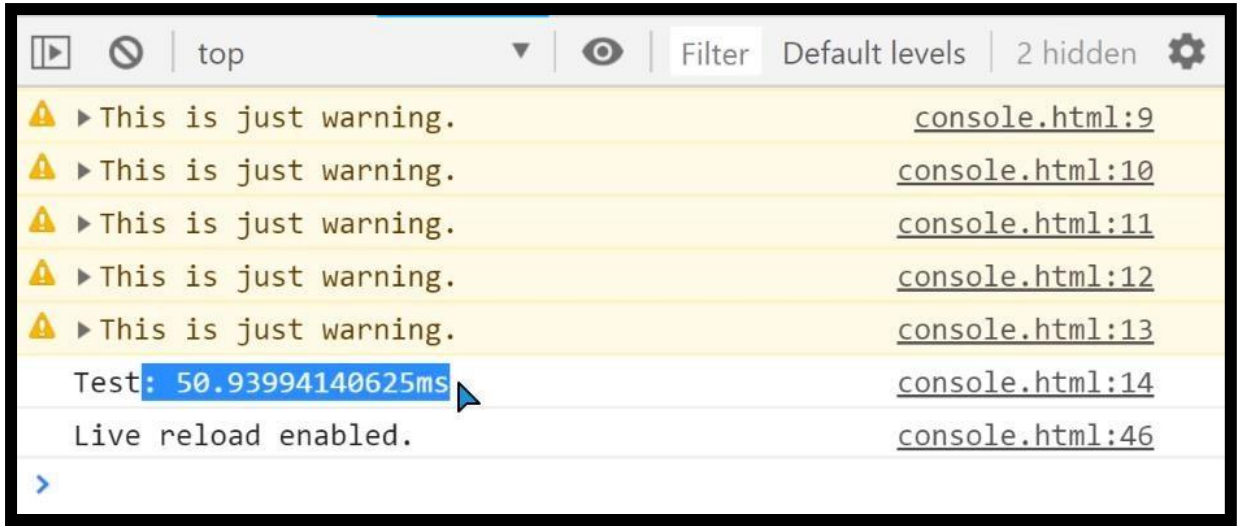


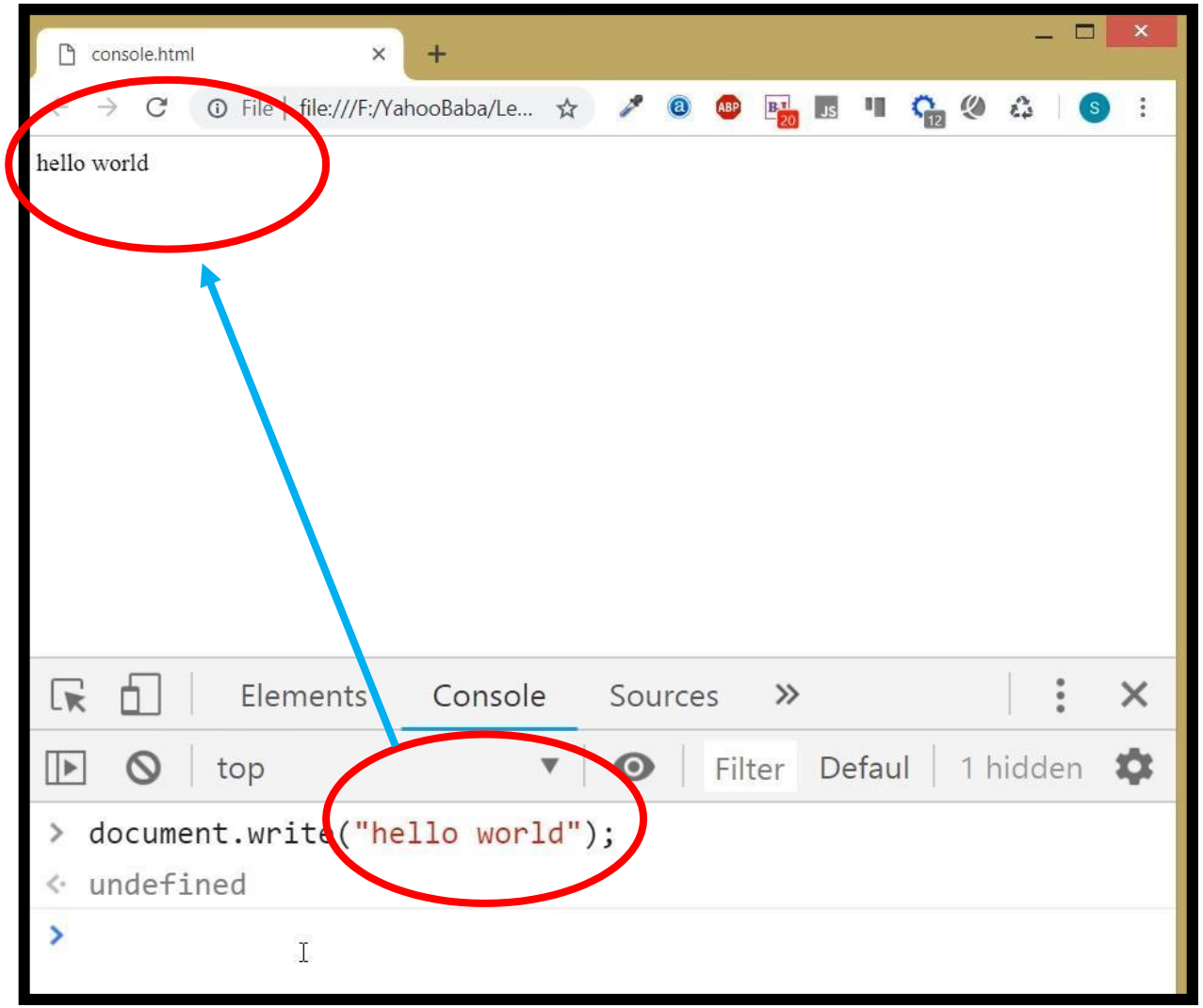

```
let x = [1,2,3];
console.table(x);
```



```
console.time("Test");
console.error("something went wrong");
console.error("something went wrong");
console.error("something went wrong");
console.error("something went wrong");
console.error("something went wrong");
console.timeEnd("Test");
```

This command Is used for knowing that how many time require to execute those commnds ,which are written between ".time & .timeEnd". (ans. return in millisecond)





And many more commands & uses are exists of console ...

Comparison Operator :

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

If Statement :

```
If(condition)
{
   //write some code
}
```

- If condition is true then execute code ,otherwise not.

Comparison Operator :

Operator	Name
&&	Logical AND
||	Logical OR
!	Logical NOT

If Statement with Logical OR :

```
If(Condition 1 || Condition 2){

}
```

Run only when either one condition must be TRUE

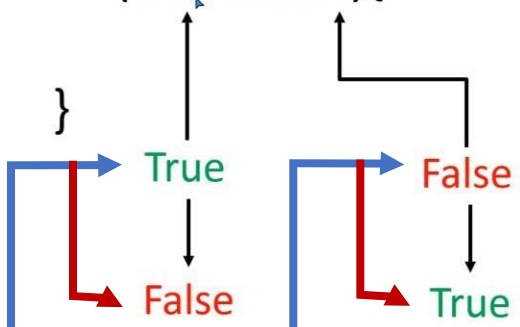
If Statement with Logical AND :

```
If(Condition 1 && Condition 2){

}
```

Run only when both conditions must be TRUE

If Statement with Logical NOT :

```
If !(Condition){

}
```



#Important notes

If else St.

```
If(Condition True){
    Statement for True
} else {
    Statement for False
  }
```

If else if St.

```
If(Condition 1){

} else If(Condition 2){

  } else {

  }
```

Conditional Ternary Operator

```
(Condition) ? True Statement : False Statement
```

var b = (cond.) ? “My Name is John.” : “My Name is Justin.”;

If above cond. Is true then ,
b = “My Name is John.” ;
Otherwise ,
b = “My Name is Justin.”;

case St.

```
<!DOCTYPE html>
    <html>
    <head>
        <title>JavaScript</title>
        <script> var day = 1;
         switch (day) {
                case 0: document.write("Today is Monday"); break;
                case 1: document.write("Today is Tuesday"); break;
                case 2: document.write("Today is Wednesday"); break;
                case 3: document.write("Today is Thursday"); break;
                case 4: document.write("Today is Friday"); break;
                case 5: document.write("Today is Saturday"); break;
                case 6: document.write("Today is Sunday"); break;
                default: document.write("Enter the valid Week Day");
            }
        </script>
    </head>
    <body></body>
    </html>
```

```
Today is Tuesday
```

```
<!DOCTYPE html>
    <html>
    <head>
        <title>JavaScript</title>
        <script> var day = 1;
         switch (day) {
                case 0: document.write("Today is Monday"); break;
                case 1: document.write("Today is Tuesday");
                case 2: document.write("Today is Wednesday");
                case 3: document.write("Today is Thursday");
                case 4: document.write("Today is Friday"); break;
                case 5: document.write("Today is Saturday"); break;
                case 6: document.write("Today is Sunday"); break;
                default: document.write("Enter the valid Week Day");
            }
        </script>
    </head>
    <body></body>
    </html>
```

```
Today is Tuesday
Today is Wednesday
Today is Thursday
Today is Friday
```

```
<!DOCTYPE html>
    <html>
    <head>
        <title>JavaScript</title>
        <script> var day = 1;
         switch (day) {
                case 0: case 1: case 2: document.write("Today is Monday"); break;
                case 3: document.write("Today is Thursday"); break;
                case 4: document.write("Today is Friday"); break;
                case 5: document.write("Today is Saturday"); break;
                case 6: document.write("Today is Sunday"); break;
                default: document.write("Enter the valid Week Day");
            }
        </script>
    </head>
    <body></body>
    </html>
```

```
Today is Monday  -> st. print for option 1, 2 or. 3;
```

```
<!DOCTYPE html>
    <html>
    <head>
        <title>JavaScript</title>
        <script> var day = 1;
         switch (day) {
                case 0: case 1: case 2: document.write("Today is Monday");
                case 3: document.write("Today is Thursday");
                case 4: document.write("Today is Friday");
                case 5: document.write("Today is Saturday");
                case 6: document.write("Today is Sunday");
                default: document.write("Enter the valid Week Day");
            }
        </script>
    </head>
    <body></body>
    </html>
```

All statements are prints from choosen option to "Enter the valid Week Day" . (for any value of *day*)
Because **break;** st. here not written.

Ex.

day = 4;
output :
```
Today is Friday
Today is Saturday
Today is Sunday
Enter the valid Week Day
```

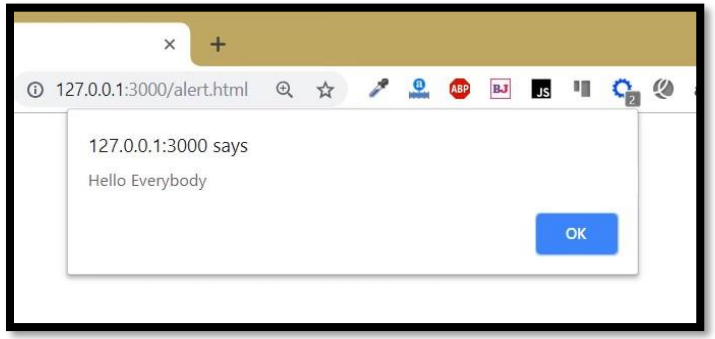
day = 5;
output :
```
Today is Saturday
Today is Sunday
Enter the valid Week Day
```

day = 6;
output :
```
Today is Sunday
Enter the valid Week Day
```

day = 7;
output :
```
Enter the valid Week Day
```

alert box

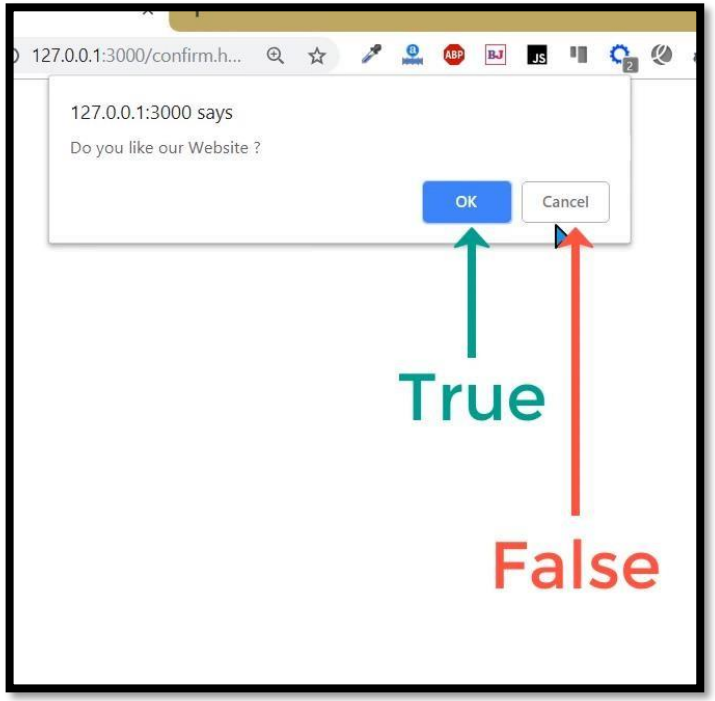
alert (“Hello Everybody”);



confirm box

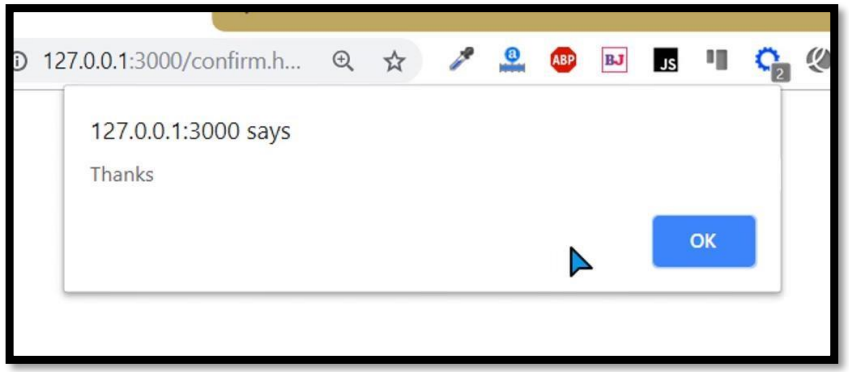
confirm (“Do you like Our website ?”);

confirm box return true or flase.

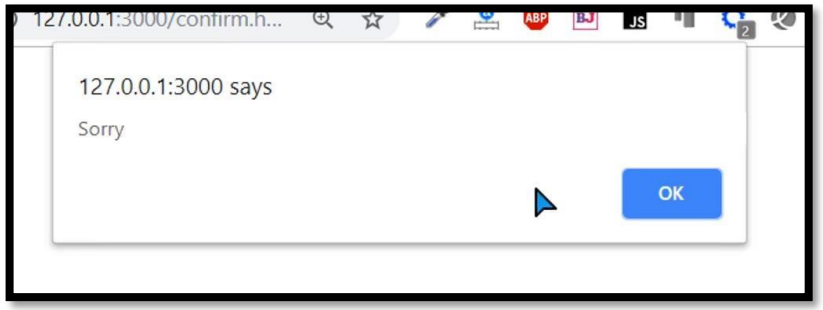



```
<script>
  var a = confirm("Do you like our Website?");
  if (a) alert("Thanks");
  else alert("Sorry");
</script>
```

OK



Cancel



prompt box

```
prompt ("What is your Name ?");
```

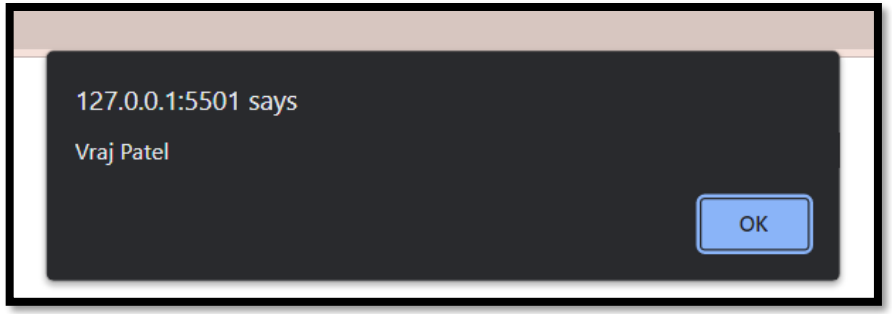
prompt box return *value* with it's *data type*.

Ex.

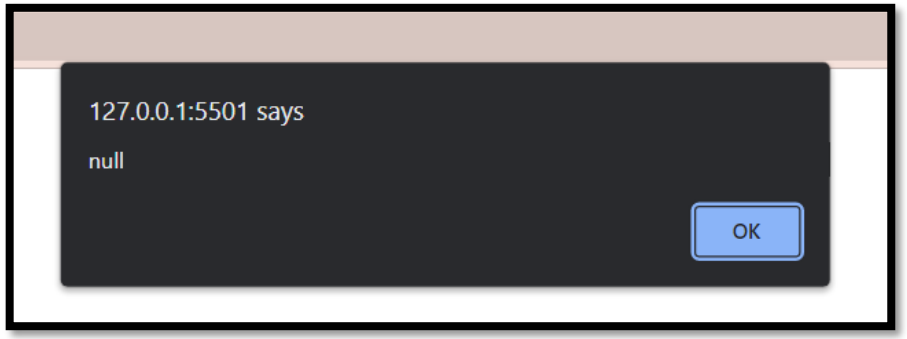
- If you enter number then you will get number.
- If you enter string then you will get string.


```
Ex :
<script>
    var a = prompt("What is your Name ?");
    alert(a);
</script>
```

If press **OK**



If press **Cancel**



Function

```
function functionName(){ ← Function Definition
    Statement
}

functionName(); ← Calling a Function
```

```
<script>
   function hello() {
      document.write("Hello Everybody");
   }
   function yahoo() {
      document.write("Yahoo Baba");
   }

   hello();
   yahoo();
</script>
```

Function calling with arguments

```
function functionName(parameter1, parameter2){
    Statement
}

functionName(argument1, argument2);
```

Called as a parameters

Called as a arguments

```
<script>
      function hello(fname= "Yahoo",lname= "Baba") {
         document.write("Hello" + fname + " " + lname + "<br>");
      }

      hello("Ram","Singh");
      hello("Salman", "Khan");
  </script>
```

Function calling with return value

```
function functionName(parameter1, parameter2){
   Statements
   return value
}

var a = functionName(argument1, argument2);
```

```
<script>
  function sum(math,eng,sc){
   var s = math + eng + sc;
   return s;
  }

  function percentage(tt){
    var per = tt/300 * 100;
    document.write(per);
  }

  var total = sum(80,80,80);

  percentage(total);
</script>
```

Global & Local Variable

```
var a = 10; ← Global Variable

function functionName(){
   var b = 25; ← Local Variable
}
```

```
<script>
      var a = "Yahoo Baba";
      function hello() {
         document.write(a + "<br>");
      }

      hello();
      document.write(a);
   </script>
```

Yahoo Baba
Yahoo Baba

```
<script>
   function hello(){
     var a = "Yahoo Baba";
     document.write(a + "<br>");
   }

   hello();
   document.write(a);
  </script>
```

error :
Variable 'a' globally not defined.

Basic events

Click events :

- Click (onclick)
- Double Click (ondblclick)
- Right Click (oncontextmenu)

keyboard events :

- Key Press (onkeypress)
- Key Up (onkeyup)

Mouse events :

- Mouse Hover (onmouseenter)
- Mouse Out (onmouseout)
- Mouse Down (onmousedown)
- Mouse Up (onmouseup)

window events :

- Load (onload)
- Unload (onunload)
- Resize (onresize)
- Scroll (onscroll)

#important points :

- *click events* & *mouse events* work ***on all HTML Tags*** .

But ,

- *keyboard events* work on only ***<form>*** & ***<body>*** tags.

- *window events* work on only ***<body>*** tag.
- Sometime *onunload* event not work properly. So, don't be hesitate.
- It has special method to use.

- Click (onclick)
- Double Click (ondblclick)
- Right Click (oncontextmenu)
- Mouse Hover (onmouseenter)
- Mouse Out (onmouseout)
- Mouse Down (onmousedown)
- Mouse Up (onmouseup)
- Key Press (onkeypress)
- Key Up (onkeyup)
- Load (onload)
- Unload (onunload)
- Resize (onresize)
- Scroll (onscroll)

#onclick : when we click left button of mouse for one time , event is fired
#ondblclick : when we click left button of mouse for two times , event is fired
#oncontextmenu : this event fired when we click right button of mouse for one time

#onmouseover : when cursor hover on that element, event is fired
#onmouseout : when cursor hover out on that element, event is fired
#onmousedown : when we **press** left button OR. Right button of mouse for one time , event is fired
#onmouseup : when we **release** left button OR. Right button from that element, event is fired

#onkeypress : when we press any key of keyboard , event is fired
#onkeydown : same as #onkeypress
#onkeyup : when we release any key of keyboard , event is fired

#onresize : when page is resized , event is fired
#onscroll : when page is scrolled , event is fired
#onload : when window is load , event is fired
#onunload : when window is unload , event is fired (not sure about this)

While loop :

1. Initialize
2. Condition
3. Increment/decrement

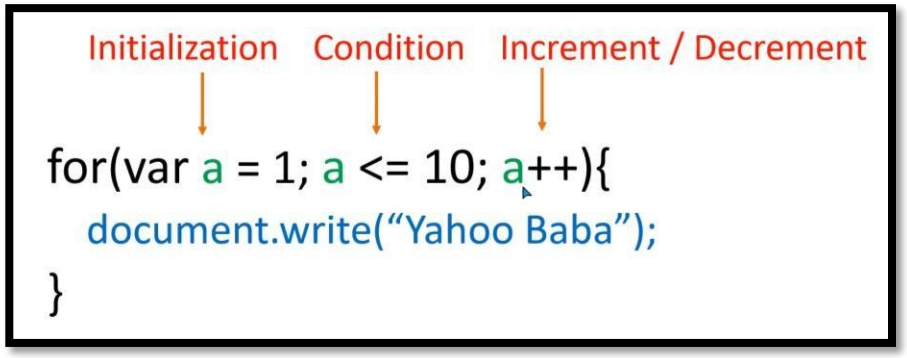
```
<script>
    var x = 10; //initialize
     while (x >= 1) { //condition

        document.write(x + ") Hello Yahoo Baba <br>");

        x = x - 1;  // increment/decrement
    }
  </script>
```

Do while loop :

1. Initialize
2. Increment/decrement
3. Condition

```
<script>
  var a = 1; //initialize
  do{
    document.write(a + " Hello Yahoo Baba<br>");

    a++; // increment/decrement

  }while(a <= 10) //condition
 </script>
```

For loop :




```
<script>
  for(var x = 1; x <= 10; x++){
    document.write("Hello Yahoo Baba <br>");
  }
 </script>
```

continue & break st.

continue st. for While ,Do while & For Loop.

```
<script>
   var a = 0;
    while (a < 5) {
        if (a == 2) {
          continue;
       }
       console.log(a);
       a++;
   }
</script>
```

While loop :
become an infinity loop.

```
<script>
   var a = 0;
   do {
      if (a == 2) {
         continue;
      }
      console.log(a);
      a++;
   } while (a < 5);
</script>
```

Do While loop :
become an infinity loop.

```
<script>
   for (var i = 0; i < 5; i++) {
      if (i == 2)
         continue;
      console.log(i);
   }
</script>
```

For loop :
Worked as a our expectation.

Solution of while & do while loop :

```
<script>
    var a = 0;
    while (a < 5) {
        a++;
        if (a == 2) {
            continue;
        }
        console.log(a);
}
</script>
```

While loop :

1
3
4
5

```
<script>
    var a = 0;
    do {
        a++;
        if (a == 2) {
            continue;
        }
        console.log(a);
} while (a < 5);
</script>
```

Do While loop :

1
3
4
5

break; st. for While ,Do while & For Loop.

```
<script>
    var a = 0;
    while (a < 5) {
        if (a == 2) {
            break;
        }
        console.log(a);
        a++;
    }
</script>
```

While loop :
Termination of Loop

While loop :
0
1

```
<script>
    var a = 0;
    do {
        if (a == 2) {
            break;
        }
        console.log(a);
        a++;
    } while (a < 5);

</script>
```

Do While loop :
Termination of Loop

Do While loop :
0
1

```
<script>
    for (var i = 0; i < 5; i++) {
        if (i == 2)
            break;
        console.log(i);
    }
</script>
```

For loop :
Termination of Loop

For loop :
0
1

Nested loop – part-1 & part-2

Nested for loop :

```
for(var a = 1; a <= 10; a++){
    for(var b = 1; b <= 10; b++){
        Statement
    }
}
```

- As well as for while & do while loop .
- Use of nested loop :
  - To print data in table format
  - To print multidimensional array
  - To print matrix
  - To print diff. – diff. types of patterns. Like …

```
1
1 2
1 2 3
1 2 3 4 5

5 4 3 2 1
5 4 3 2
5 4 3
5 4
5

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
*
**
***
****
*****

*****
****
***
**
*

    *****
     ****
      ***
       **
        *
```

Array – part-1

```
var a = 10;
var a = 10,20,30;   ⟶ Error
var a = "10,20,30"; ⟶ String
var a = [10,20,30]; ⟶ Array
```

```
          0   1   2  ⟵ Index
var a = [10,20,30];
```

- In array we can store multiple values which have different data-types.

Ex :

```
var a = [ "sohan", 12, 34, true, null ];
```

Another way to define an Array – part-2

- var arr = new Array( );
  - ✓ In this case build an empty array.

- var arr = new Array(5);
  - ✓ In this case array have size = 5. But you want to push an element at 7^{th} index then you can also do it. That's not throw the error.

- var arr = new Array("sohan", 12, 34, true, null);
  - ✓ In this case array have 5 elements. But you want to push an element at 9^{th} index then you can also do it. That's not throw the error.

➢ For reference point of you , you can see the "video34.html" file.

Multidimensional array

Name	Age	Gender	Class
Harry	18	Male	B.Com
Sunny	19	Male	BCA
Sarah	18	Female	BCA
Tom	17	Male	B.A.

```
var a = [   0        1        2          3
    0  ["Harry" , 18 , "Male" , "B.Com"],
    1  ["Sunny" , 19, "Male" , "BCA"],
    2  ["Sarah" , 18 , "Female" , "BCA"],
    3  ["Tom" , 17 , "Male" , "B.A."]
    ];
```

Multidimensional array used for to print a data in table manner.

Modify & delete from array

Modification is possible by a replacement

```
var a = [ "sohan", 12, 34, true, null ];

a[0] = 345;
a[2] = "nilesh";
```

After :

```
var a = [ 345, 12, "nilesh", true, null ];
```

Deletion is possible by a delete keyword

```
var a = [ "sohan", 12, 34, true, null ];

delete a[1];
```

After :

"sohan", , 34, true, null

- Here, comma (,) is remaining.
- So, a[1] exist with undefined value.

- document.write (a[1]);
- Output : *undefined*

#important

Array Methods
Or.
Array Predefined Functions

- sort()
- reverse()
- pop()
- push()
- shift()
- unshift()
- concat()
- join()
- slice()
- splice()
- isArray()
- indexOf()
- lastIndexOf()
- entries()
- every()
- filter()
- find()
- findIndex()
- includes()
- some()
- forEach()
- toString()
- valueOf()
- fill()

sort() & reverse()

- sort() method do the sort the array alphabetically or numerically .
- Then store the (new)sorted array in to the main array .

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.sort();
```

After :

```
var a = [“john” , “kailnd” , “kushal” , “nitisha” , “raj” , “sohan”];
```

- reverse() method do the reverse the whole array.
- Then store the (new)reverse array in to the main array.

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.reverse();
```

After :

```
var a = [“raj” , “kailnd” , “john” , “nitisha” , “kushal” , “sohan”];
```

pop() & push()

- By pop() method we can remove last element of the array.
- Then store newArray store in to the mainArray.

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.pop();
```

After :

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” ];
```

- By push() method we can add element at the last position of the array.
- Then store newArray store in to the mainArray.

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.push( “rahul” );
```

After :

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj” , “rahul”];

a.push( 14 );
```

After :

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj” , “rahul” , 14];
```

pop() & push()

- By pop() method we can remove last element of the array.
- Then store newArray store in to the mainArray.

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.pop();
```

After :

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” ];
```

- By push() method we can add element at the last position of the array.
- Then store newArray store in to the mainArray.

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.push( “rahul” );
```

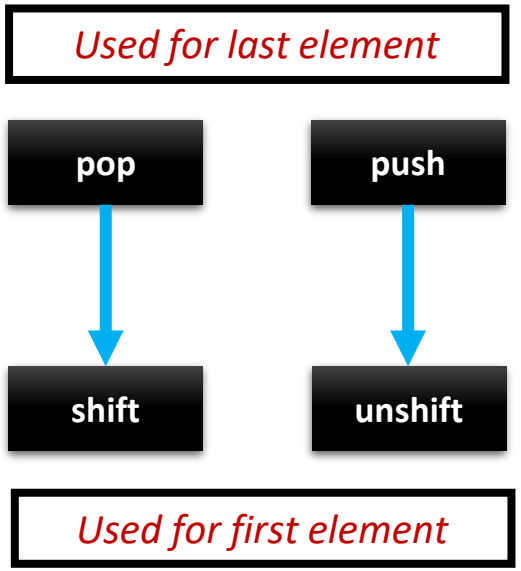
After :

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj” , “rahul”];

a.push( 14 );
```

After :

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj” , “rahul” , 14];
```

shift() & unshift()



- By shift() method we can remove first element of the array.
- Then store newArray store in to the mainArray.

Ex.

var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.shift();

After :

var a = [“kushal” , “nitisha” , “john” , “kailnd” , “raj”];

- By unshift() method we can add element at the first position of the array.
- Then store newArray store in to the mainArray.

Ex.

var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.unshift(“rahul”);

After :

var a = [“rahul” , “sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

a.unshift(14);

After :

var a = [14 , “rahul” , “sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

concat() & join()

- By join() method we can convert whole array in to the one string.
- In this case newString created.
- So , we have to require a create new variable to store new string.

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];
var c = a.join( “-” );
```

After :

```
var c = “sohan-Kushal-nitisha-john-kailnd-raj”
```

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];
var c = a.join( “ - ” );  //with space
```

After :

```
var c = “sohan – Kushal – nitisha – john – kailnd – raj”
```

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];
var c = a.join( “ ** ” );
```

After :

```
var c = “sohan ** Kushal ** nitisha ** john ** kailnd ** raj”
```

- By concat() method we can concatenate more than one arrays at the end of the main array.
- In this case newArray doesn't store in to the mainArray.
- So , we have to require a create new array by own self.

Ex.

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];
var b = [“suraj” , “meet” , “jay”];
var c = a.concat( b );
```

After :

```
var c = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj” , “suraj” , “meet” , “jay”];
```

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

var c = a.concat( “suraj” , “meet” , “jay” );
```

After :

```
var c = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “suraj” , “meet” , “jay”];
```

```
var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];
var b = [“suraj” , “meet” , “jay”];
var c = [“abhay” , “dev” , “mihir”];

var d = a.concat( c , b );
```

After :

```
var d = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj” , “abhay” , “dev” , “mihir” , “suraj” , “meet” , “jay” ];
```

slice() & splice()

- By slice() method we can get multiple elements from any position.

❑ Rules :

-6	-5	-4	-3	-2	-1
0	1	2	3	4	5

var a = [“sohan” , “kushal” , “nitisha” , “john” , “kailnd” , “raj”];

Starting point **Ending point**

- Syntax :

var c = arr.slice(x , y); → Gives array from x^{th} to $(y-1)^{th}$ element

var c = arr.slice(1 , 4); // 1 to 3
var c = [“kushal” , “nitisha” , “john”];

var c = arr.slice(-5 , -1); // -5 to -2
var c = [“kushal” , “nitisha” , “john” , “kailnd”];

var c = arr.slice(2); // 2 to upToEnd
var c = [“nitisha” , “john” , “kailnd” , “raj”];

var c = arr.slice(-5); // -5 to upToEnd
var c = [“kushal” , “nitisha” , “john” , “kailnd” , “raj”];

- By splice() method we can insert a element at any position
- By splice() method we can delete multiple elements for any range
- By splice() method we can perform both above operations parallelly

- Syntax :
- var c = a.splice(index , how many delete , elements' values)

- ✓ In **c** the elements which are *deleted*.
- ✓ In **a** element which are *not deleted* & *newly inserted.*

var a = ["sohan" , "kushal" , "nitisha" , "john" , "kailnd" , "raj" , "shivansh" , "rahul" , "aman"];

- **Only Insertion :**

var c = a.splice(2 , 0 , "sahil",14]
c = [];
a = ["sohan" , "kushal" , "sahil" , 14 , "nitisha" , "john" , "kailnd" , "raj" , "shivansh" , "rahul" , "aman"];

- **Only Deletion :**

var c = a.splice(2 , 3);
c = ["nitisha" , "john" , "kailnd"];
a = ["sohan" , "kushal" , "raj" , "shivansh" , "rahul" , "aman"];

- **Insertion & Deletion :**

var c = a.splice(2 , 3 , "sahil",14]
c = ["nitisha" , "john" , "kailnd"];
a = ["sohan" , "kushal" , "sahil" , 14 , "raj" , "shivansh" , "rahul" , "aman"];

isArray()

- isArray() method returns
- true if variable is an array
- false if variable is not an array

```
var a = [“sohan” , “kushal” , “nitisha” , “john”];
var b = 23.234;
var c = “hsiuqw”;
var d = true;

isArray( a ); → true
isArray( b ); → false
isArray( c ); → false
isArray( d ); → false
```

indexOf() & lastIndexOf()

- **indexOf() :**

✓ Syntax : a.indexOf("search item" , starting index)

✓ indexOf() method returns elements' index if element is find out otherwise returns -1

✓ In this method always traversal done in left to right direction

✓ Here case sensitive searching occurs.

✓ Starting index by default is 0.

- **lastIndexOf():**

✓ Syntax : a.lastIndexOf("search item" , starting index)

✓ lastIndexOf() method returns elements' index if element is find out otherwise returns -1

✓ In this method always traversal done in right to left direction

✓ Here case sensitive searching occurs.

✓ Starting index by default is last index.

```
var a = [“sohan” , “kushal” , “shivansh” , “kushal” , “aman” , “nitisha” , “john” , “kailnd” ];
        0        1          2             3          4          5          6          7
```

indexOf()

Searching starting from left to right

```
//searching starts from index 0
var c = a.indexOf(“kushal”);
c = 1;

var c = a.indexOf(“dhiru”);
c = -1;

//searching starts from index 2
var c = a.indexOf(“kushal” , 2);
c = 3;

//searching starts from index 3
var c = a.indexOf(“kushal” , 3);
c = 3;

//searching starts from index 4
var c = a.indexOf(“kushal” , 4);
c = -1;

//case sensitive searching
var c = a.indexOf(“Kushal” );
c = -1;
```

```
//searching starts from index 7
var c = a.lastIndexOf(“kushal”);
c = 3;

var c = a. lastIndexOf(“dhiru”);
c = -1;

//searching starts from index 5
var c = a. lastIndexOf(“kushal” , 5);
c = 3;

//searching starts from index 3
var c = a. lastIndexOf(“kushal” , 3);
c = 3;

//searching starts from index 4
var c = a. lastIndexOf(“john” , 4);
c = -1;

//case sensitive searching
var c = a. lastIndexOf(“Kushal” );
c = -1;
```

lastIndexOf()

Searching starting from right to left

includes()

var a = [“sohan” , “kushal” , “shivansh” , “kushal” , “aman” , “nitisha” , “john” , “kailnd”];
0 1 2 3 4 5 6 7

- includes() returns
- True if element is find
- Flase if element is not find.
- Here case sensitive searching occurs.

Synatx :

var c = a.includes(“search item”);

- var c = a.includes(“john”);
- c = true
- var c = a.includes(“kush”);
- c = false

//case sensitive searching

- var c = a.includes(“John”);
- c = true

some() & every()

- **some (functionName) :**

- If for at least one element , condition is true then give true.
- For all elements , condition is false then give false.

- **every (functionName) :**

- If for all element , condition is true then give true.
- Otherwise give false

2nd approach
Arrow function

```
var a = [12, 18, 2, 14, 17];

var c = a.some((age) => {
   return age >= 18;
});
document.write(c); → true
```

1st approach

```
var a1 = [12, 18, 2, 14, 17];
var a2 = [10, 8, 5, 4, 7];
var a3 = [28, 18, 25, 41, 73];

var c = a1.some(checkAdult);
document.write(c); → true
var c = a2.some(checkAdult);
document.write(c); → false
var c = a3.some(checkAdult);
document.write(c); → true

var d = a1.every(checkAdult);
document.write(d); → false
var d = a2.every(checkAdult);
document.write(d); → true
var d = a3.every(checkAdult);
document.write(d); → false

function checkAdult(age) {
   return age >= 18;
}
```

find() & findIndex()

- **find (functionName) :**

- If for at least one element , condition is true then give that first element for which condition is true.
- For all elements , condition is false then give undefined.

- **findIndex (functionName) :**

- If for all element , condition is true then give index of that first element for which condition is true.
- Otherwise give -1

2nd approach
Arrow function

```
var a = [12, 18, 2, 14, 17];

var c = a.find((age) => {
   return age >= 18;
});
document.write(c); → true
```

1st approach

```
var a1 = [12, 18, 2, 14, 17];
var a2 = [10, 8, 5, 4, 7];
var a3 = [28, 18, 25, 41, 73];

var c = a1.find(checkAdult);
document.write(c); → 18
var c = a2.find(checkAdult);
document.write(c); → undefined
var c = a3.find(checkAdult);
document.write(c); → 28

var d = a1.findIndex(checkAdult);
document.write(d); → 1
var d = a2.findIndex(checkAdult);
document.write(d); → -1
var d = a3.findIndex(checkAdult);
document.write(d); → 0

function checkAdult(age) {
   return age >= 18;
}
```

filter()

- **filter (functionName) :**

✓ This filter() method return an array.
✓ This array consist a elements for , which condition is true.
✓ This array is Empty. When for at least one element condition is not match.

1st approach

```
var a1 = [12, 18, 2, 14, 17];
var a2 = [10, 8, 5, 4, 7];
var a3 = [28, 18, 15, 14, 73];

var c = a1.filter(checkAdult);
document.write(c); →[18]
var c = a2.filter(checkAdult);
document.write(c); → [ ]  //empty array
var c = a3.filter(checkAdult);
document.write(c); → [28 , 18 , 73]

function checkAdult(age) {
   return age >= 18;
}
```

2nd approach
Arrow function

```
var a = [12, 18, 2, 14, 17];

var c = a.filter((age) => {
   return age >= 18;
});
document.write(c); → true
```

valueOf() , toString() & fill()

var a = ["Sanjay","Aman","Rehman"];

Sanjay, Aman, Rehman

toString()

The toString() method converts an array into a String and returns the result.

var a = ["Aman","Rehman“,”Karan”];

Aman, Rehman, Karan

valueOf()

The valueOf() method returns the array.

var a = ["Aman","Rehman“,”Karan”];

Ram

[“Ram","Ram“,”Ram”]

fill()

The fill() method fills all the elements in an array with a static value.

```
var a = [12, "sohan", 2, true, "17"];

var c = a.valueOf();
document.write(c + "<br><br>");

var c = a.toString();
document.write(c + "<br><br>");

var c = a.fill();
document.write(c + "<br><br>");

var c = a.fill("ok");
document.write(c + "<br><br>");
```

Output :

12,sohan,2,true,nilesh → Here all values in real data types

12,sohan,2,true,nilesh → Here all values in string data type

,,,,

ok,ok,ok,ok,ok

forEach Loop

- Syntax :
- This is work only for *array*.

```
arrayName. forEach((element, index) => {  ...  });
```

```
var a = ["sohan", "nilesh", "nitisha", "aksh", "raj", "rahul", "Aman", "ok"];

// 1st approach (in this case syntax is most imporatant)
a.forEach((element, index) => {
   document.write(index + " - " + element + "<br>");
});

// 2nd approach
a.forEach(funName);

function funName(element, index) {
   document.write(index + " - " + element + "<br>");
}
```

```
//in both case same output
0 - sohan
1 - nilesh
2 - nitisha
3 - aksh
4 - raj
5 - rahul
6 - Aman
7 - ok
```

Object & this operator

```
let mainObj = {
            firstname: "kush",
            lastname: "patel",
            std: 10,
            Address: "abcd complex, near xxxx road, ahmdabd",
            email: "abc1234@gmail.com",
            // array
            hobby: ["cricket", "cards", "singing", "reading"],
            // methods (functions)
            fullname: function () {
                return this.firstname + " " + this.lastname;
            },
            salary: (bonus = 2000) => {
                return bonus + 15000;
            },
            // object
            family: {
                wife: "xyz",
                father: "FatherName",
                mother: "MotherName",
                childrens: ["Name1", "Name2", "Name3"],
                NoOfChildrends: 3,
                myName: () => {
                    return mainObj.fullname();
                }
            }
        };
```

- Firstname , lastname, email etc. all are known as property.
- And also array (hobby , childrens) are known as property.
- Functions which are defined in a object that's are called as a methods.
- To achieve the methods in a object ,we have 2 options.

1) Normal way

- funName : function(){…},

2) Arrow function

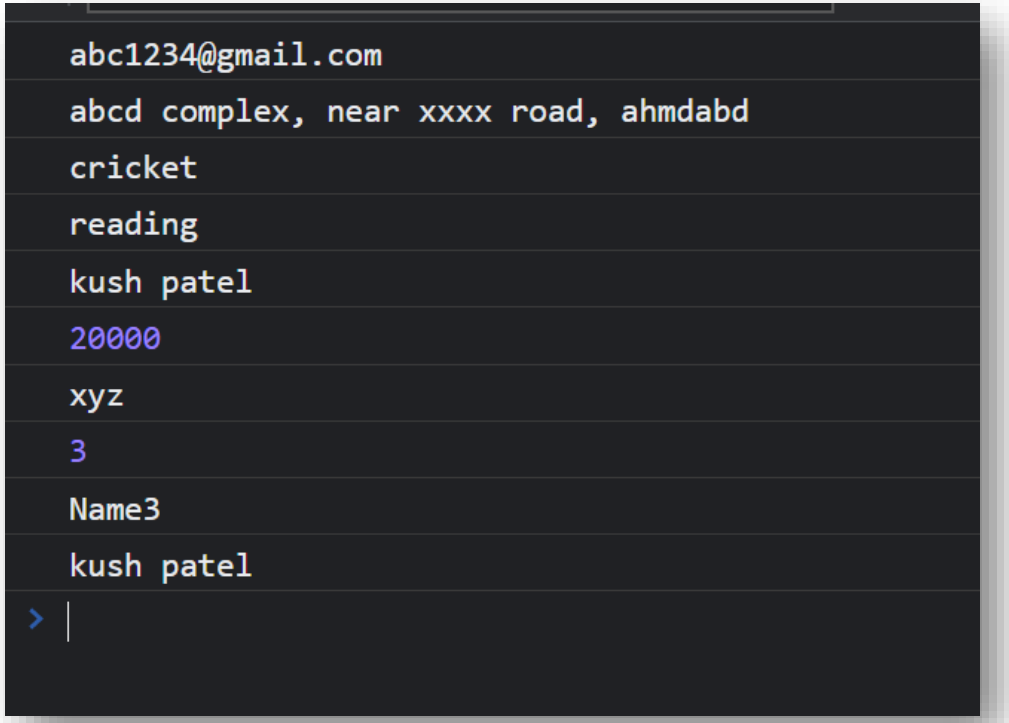
- funName : ( )=>{…},

- "kush" , "patel" , ["cricket", "cards", "singing", "reading"] , "abc1234@gmail.com " – which all are known as values of those properties.

In console field object display like this

- here, *this* represents the *mainObj*
- Means, this.firstname = mainObj.firstname
- how to access ?

```
// property
console.log(mainObj.email);
console.log(mainObj.Address);
// array(property)
console.log(mainObj.hobby[0]);
console.log(mainObj.hobby[3]);
// methods (functions)
console.log(mainObj.fullname());
console.log(mainObj.salary(5000));
// object
console.log(mainObj.family.wife);
console.log(mainObj.family.NoOfChildrends);
console.log(mainObj.family.childrens[2]);//#imp
console.log(mainObj.family.myName());//#imp
```




```
console.log(mainObj.email);
console.log(mainObj['email']);
console.log(mainObj.family.childrens[2]);
console.log(mainObj["family"]["childrens"][2]);
console.log(mainObj.family.myName());
console.log(mainObj["family"]["myName"]());
```

- Both are same.

Object – part-2

Cerate a new object by new keyword

Add all properties and methods externally

```
let mainObj = new Object( ) ;
```

```
mainObj.firstname = "kush";
mainObj.lastname = "patel";
mainObj.std = 10;
mainObj.Address = "abcd complex, near xxxx road, ahmdabd";
mainObj.email = "abc1234@gmail.com";
mainObj.hobby = ["cricket", "cards", "singing", "reading"];
// methods (functions)
mainObj.fullname = function () {
   return this.firstname + " " + this.lastname;
};
mainObj.salary = (bonus = 2000) => {
   return bonus + 15000;
};
```

```
mainObj.family = {
        wife: "xyz",
        father: "FatherName",
        mother: "MotherName",
        childrens: ["Name1", "Name2", "Name3"],
        NoOfChildrends: 3,
        myName: () => {
           return mainObj.fullname();
        }
    };
```

- how to access ?

```
// property
console.log(mainObj.email);
console.log(mainObj.Address);
// array(property)
console.log(mainObj.hobby[0]);
console.log(mainObj.hobby[3]);
// methods (functions)
console.log(mainObj.fullname());
console.log(mainObj.salary(5000));
// object
console.log(mainObj.family.wife);
console.log(mainObj.family.NoOfChildrends);
console.log(mainObj.family.childrens[2]);//#imp
console.log(mainObj.family.myName());//#imp
```

```
abc1234@gmail.com
abcd complex, near xxxx road, ahmdabd
cricket
reading
kush patel
20000
xyz
3
Name3
kush patel
>
```

All things (output & access methods) are as same as previous example

Array of Object

```
var student = [
        { Name : “Ram” , Age : 15 },
        { Name : “Karam” , Age : 16 },
        { Name : “Rahul” , Age : 14 },
];
```

student[0]

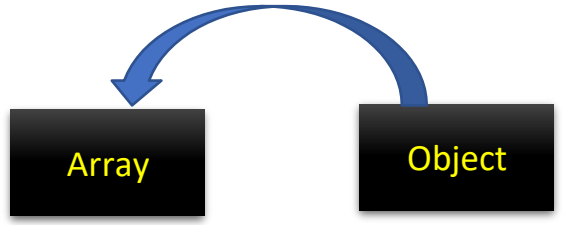
student[1]

student[2]

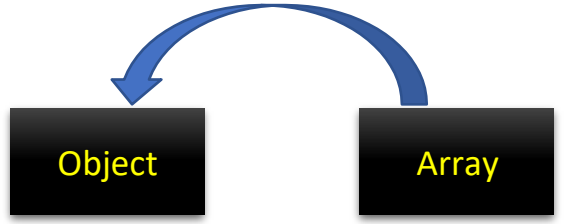
- How to access ?

```
student[0].name;
student[0].Age;

student[1].name;
student[1].Age;

student[2].name;
student[2].Age;
```

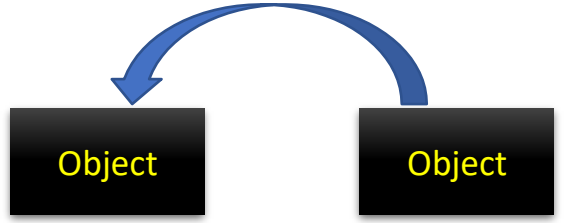
Array of object means



Object of array means



Object of object means



const array & object

```
const arr = [10, 20, 30];
arr = [10, 50, 40];

arr[1] = 50;
arr[2] = 40;

console.log(arr);
```

10,50,40

This is not a right way to change constant array's values

This is a right way to change constant array's values

(We have to require target a particular property and then change it's values.)

```
const Obj = {
   name: "kush",
   age: 18,
   email: "abcd@gmail.com"
}

Obj = {
   name: "kush",
   age: 24,
   email: "pqrst@gmail.com"
}

Obj.age = 24;
Obj.email = "pqrst@gmail.com";

console.log(Obj.age);
console.log(Obj.email);
```

24
pqrst@gmail.com

This is not a right way to change constant object's values

This is a right way to change constant object's values

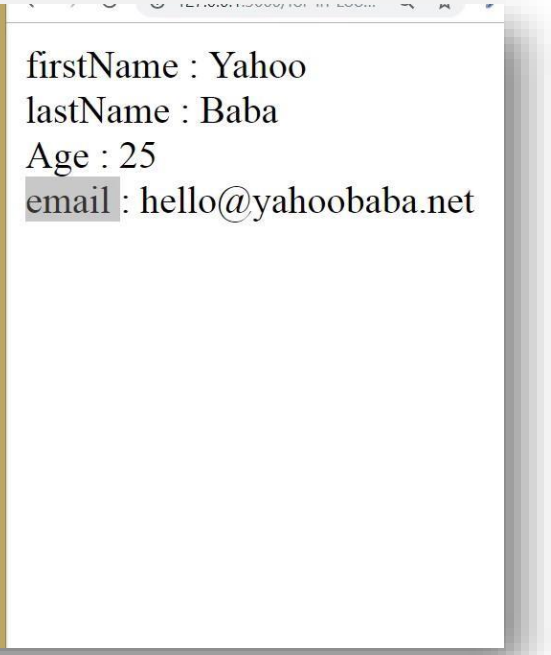
(We have to require target a particular property and then change it's values.)

For in loop

Syntax :

```
for( var property in objName )
{
    . . .
    . . .
}
```

```
<script>
  var obj = {
    firstName : "Yahoo",
    lastName : "Baba",
    Age : 25,
    email : "hello@yahoobaba.net"
  };

  for(var key in obj){
    document.write(key + " : " + obj[key] + "<
  }
</script>
```



map() method

- To perform a particular operation on each element , then we use this map() method

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript</title>
  <script>
    var ary = [11,4,9,16];

    var b = ary.map(test);
    document.write(b);

    function test(x){
      return x * 10;
    }
  </script>
</head>
```

110,40,90,160

```
<title>JavaScript</title>
<script>
  var ary = [
    {fname : "Yahoo" , lname : "Baba"},
    {fname : "Rahul" , lname : "Kumar"},
    {fname : "Karan" , lname : "Sharma"},
  ];

  var b = ary.map(test);
  document.write(b);

  function test(x){
    return x.fname;
  }
</script>
```

Yahoo,Rahul,Karan

```
<title>JavaScript</title>
<script>
  var ary = [
    {fname : "Yahoo" , lname : "Baba"},
    {fname : "Rahul" , lname : "Kumar"},
    {fname : "Karan" , lname : "Sharma"},
  ];

  var b = ary.map(test);
  document.write(b);

  function test(x){
    return x.fname + " " + x.lname;
  }
</script>
```

Yahoo Baba,Rahul Kumar,Karan Sharma

String methods – part-1 & 2

- length
- toLowerCase()
- toUpperCase()
- includes()
- startsWith()
- endsWith()
- search()
- match()
- indexOf()
- lastIndexOf()
- replace()
- trim()
- charAt()
- charCodeAt()
- fromCharCode()
- concat()
- split()
- repeat()
- slice()
- substr()
- substring()
- toString()
- valueOf()

```
var str = "Javascript is a GREAT is Language";
```

```
// length – give a total length of a string.(including space)
var a = str.length;
/* 33 */
```

```
// toLowerCase() - convert a all characters of string in to the lowercase.
// toUpperCase() - convert a all characters of string in to the uppercase.
var a = str.toLowerCase();
var b = str.toUpperCase();
/*
javascript is a great is language
JAVASCRIPT IS A GREAT IS LANGUAGE
*/
```

```
// includes("serachItem") -
//search the word in to the entire string. if find it out then return true otherwise return false.
// searching is case sensitive

var a = str.includes("Javascript");
var b = str.includes("javascript");
var c = str.includes("ipt");
/* true false true */
```

```
// startWith("serachItem") - if by given word , string is started then return true , Otherwise return flase.
// endWith("serachItem") - if by given word , string is ended then return true , Otherwise return flase.
var a1 = str.startsWith("java");
var a2 = str.startsWith("Java");
var a3 = str.endsWith("Javascript is");

var b1 = str.endsWith("age");
var b2 = str.endsWith("Age");
var b3 = str.endsWith("Language");
/* false true false true false true */
```

```
//search("serachItem") -
// if word is find it out then return "index of first character of searching word" , Otherwise return -1
// index starts from 0.
// searching is casesensitive
// note : if "is" occurs 2 times in a string then return index for first result(means for first "is").
var a = str.search("is");
var b = str.search("Is");
/* 11 -1 */
```

```
//match(/serachItem/g) - "g" denoted globally search in entire string
// if word is find it out then return array.
// this array is consits a searching word for "n" times. Otherwise return EMPTY ARRAY.
// searching is casesensitive

var a = str.match(/i/g);
var b = str.match(/is/g);
var c = str.match(/a/g);
var d = str.match(/Is/g);
/*
['i', 'i']
['is']
['a', 'a', 'a', 'a', 'a']
[ ]
*/
```

```
//indexOf("serachItem") -
// if word is find it out then return "index of first character of searching word" , Otherwise return -1
// searching starts from 0.
// searching is case sensitive
// left To right Traversal
// note : if "is" occurs 2 times in a string then return index for first result(means first "is").

//lastIndexOf("serachItem") -
// if word is find it out then return "index of first character of searching word" , Otherwise return -1
// searching starts from "lastIndex".
// searching is case sensitive
// right To left Traversal
// note : if "is" occurs 2 times in a string then return index for first result(means first "is").

var a = str.indexOf("is");
/* 11 */
var a = str.lastIndexOf("is");
/* 22 */
```

```
// replace("itemName","newWord") -
//it is used for replace word with a particular word.
//case sensitive rule is occured.
//if word is not found then string is remaining unchanged

var a = str.replace("Javascript", "PHP");
/* PHP is a GREAT is Language */

// special cases
var a = str.replace("is", "are");
/* Javascript are a GREAT is Language */
var a = str.replace(/is/g, "are"); // all "is" are changes from entire stirng.
/* Javascript are a GREAT are Language */

var a = str.replace("Is", "are"); //unchanged (because case-sensitive)
/* Javascript is a GREAT is Language */
var a = str.replace("ok", "are"); //unchanged (because not found)
/* Javascript is a GREAT is Language */
```

// trim() - it is used fir to trim a space. From starting & ending point of string.
var temp = " javascript ";
var a = temp.trim();
/* "javascript" */

// charAt(index) - It returns a character
var a = str.charAt(2);
var b = str.charAt(11);
/* a = "v" , b = "i" */

// charCodeAt(index) - It returns a ASCII Value of a character
var a = str.charCodeAt(2);
var b = str.charCodeAt(11);
/* a = 118 , b = 105 */

// fromCharCode(ASCII_CODE) - it returns character based on ASCII Value .(for this method we have not required a string.)
var a = *String.fromCharCode*(118);
var b = *String.fromCharCode*(65);
/* a = "v", b = "A" */

```
// concat(str1,str2,...) - by this method we can concatinate more than one string to each other.
var temp1 = "HTML is a very Good Langauge."
var temp2 = "C++ is an Object oriented programing language."
var a = str.concat(temp1, temp2);
/* Javascript is a GREAT is LanguageHTML is a very Good Langauge.C++ is an Object oriented programing language */
```

```
//split("Enter a word from where to split to engtire a string") - it is used for splitting of string.
var a = str.split("i");
var b = str.split("a");
var c = str.split("is");
var d = str.split(" ");
/*
['Javascr', 'pt ', 's a GREAT ', 's Language’]      → see carefully “ i “ removed from entire array.
['J', 'v', 'script is ', ' GREAT is L', 'ngu', 'ge’]   → see carefully “ a “ removed from entire array.
['Javascript ', ' a GREAT ', ' Language’]          → see carefully “ is “ removed from entire array.
['Javascript', 'is', 'a', 'GREAT', 'is', 'Language’]   → see carefully “ “ (space) removed from entire array.
*/
```

```
// repeat(n) - by this method we can get entire string n times repetively.
var a = str.repeat(2);
var b = str.repeat(5);
/*
a = "Javascript is a GREAT is LanguageJavascript is a GREAT is Language"
b = "Javascript is a GREAT is LanguageJavascript is a GREAT is LanguageJavascript is a GREAT is LanguageJavascript is a GREAT is
LanguageJavascript is a GREAT is Language"
*/
```

```
//slice(x,y) - as same as array.(return xth element to (y-1)th element)
var a = str.slice(2, 7);
var b = str.slice(12);
var c = str.slice(-4);
var d = str.slice(-6, -1);
/*
a = "vascr"
b = "s a GREAT is Language"
c = "uage"
d = "nguag"
*/
```

```
// sustr(x , n) -
// x - is a starting index.
// n - from x index how many elements are prints ?
var a = str.substr(2, 7);
var b = str.substr(1, 12);
var c = str.substr(3);
/*
a = "vascrip"
b = "avascript is"
c = "ascript is a GREAT is Language"
*/
```

```
// substring(x,y) -
// as same as slice(x , y)
// but substring can't deal with negative arguments.
```

```
// toStirng() - convert in to the string form
var temp1 = 20;
var temp2 = [20, 23, 3435, 54];
var temp3 = {
    name: "kush",
    age: 19,
    email: "abcd@gmail.com"
};
var a = temp1.toString() + 40;
var b = temp2.toString();
var c = temp3.toString();
/*
a = "2040" (string)    (not possible 20 + 40 = 60)
b = "20,23,3435,54" (string)
c = [object Object] (string)
*/
```

```
// valueOf() - it always returns value with it's "actual data type".
var temp = 90;
var a = str.valueOf();
var b = temp.valueOf();
/*
a = "Javascript is a GREAT is Language"(data type = string)
b = 90 (data type = number)
*/
```

Number methods

- number()
- parseInt()
- parseFloat()
- isFinite()
- isInteger()
- toFixed(x)
- toPrecision(x)

```
// Number(string) –

// By it's we can convert a string in to the number.
var str1 = "100";
var str2 = "100 marks";
var a = Number(str1);
var b = Number(str2);
/*
a = 100(number)
b = NaN(Not a Number)
*/
```

```
// parseInt(string) –

// By it's we can convert a string or floating number in to the Integer number.
var str1 = "100";
var str2 = "100 marks";
var str3 = "marks 100";
var str4 = "99 88";
var str5 = 989.3456
var str6 = -989.3456
var a = parseInt(str1);
var b = parseInt(str2);
var c = parseInt(str3);
var d = parseInt(str4);
var e = parseInt(str5);
var f = parseInt(str6);
/*
a = 100(number)
b = 100(number)
c = NaN(Not a Number)
d = 99(number)
e = 989(number)
f = -989(number)
*/
```

```
// parseFloat(string) –

// By it's we can convert a string or integer number in to the floating number.
var str1 = "100.345";
var str2 = "100.345 marks";
var str3 = "marks 100.345";
var str4 = "99.345 88.345";
var str5 = 989;
var str6 = -989;
var a = parseFloat(str1);
var b = parseFloat(str2);
var c = parseFloat(str3);
var d = parseFloat(str4);
var e = parseFloat(str5);
var f = parseFloat(str6);
/*
a = 100.345(number)
b = 100.345(number)
c = NaN(Not a Number)
d = 99.345(number)
e = 989(integer number)
f = -989(integer number)
*/
```

```
// Number.isInteger(value) –

//it returns true , if (value has datatypa is number) && (it's an integer)
// otherwise returns false
var str1 = "100.345";
var str2 = "100.345 marks";
var str3 = "marks 100.345";
var str4 = "99.345 88.345";
var str5 = 989;
var str6 = -989;
var str7 = -989.8689;
var a = Number.isInteger(str1);
var b = Number.isInteger(str2);
var c = Number.isInteger(str3);
var d = Number.isInteger(str4);
var e = Number.isInteger(str5);
var f = Number.isInteger(str6);
var g = Number.isInteger(str7);
/*
a = false
b = false
c = false
d = false
e = true
f = true
g = false
*/
```

```
// Number.isFinite(value) –

//it returns true , if value is finite
// otherwise returns false
var str1 = Infinity; //special case
var str2 = 989;
var str3 = -989.8689;
var str4 = "100.345";
var str5 = "100.345 marks";
var str6 = "marks 100.345";
var str7 = "99.345 88.345";
var a = Number.isFinite(str1);
var b = Number.isFinite(str2);
var c = Number.isFinite(str3);
var d = Number.isFinite(str4);
var e = Number.isFinite(str5);
var f = Number.isFinite(str6);
var g = Number.isFinite(str7);
/*
a = false
b = true
c = true
d = false
e = false
f = false
g = false
*/
```

```
// val.toFixed(n) –

//it returns "rounded value“
// it’s work only on numbers.
// n = up to how many numbers , you want to fixed
// value = abcd.xvwyz --> then it is started from "x" to count a "n".
var val1 = 989.56789;
var val2 = -989.8689;
var val3 = 989;
var a = val1.toFixed(3);
var b = val2.toFixed(3);
var c = val3.toFixed(3);
/*
a = 989.568
b = -989.869
c = 989.000
*/
```

```
// val.toPrecision(n) –

//it returns "rounded value“
// it’s work only on numbers.
// n = up to how many numbers , you want to fixed
// value = abcd.xvwyz --> then it is started from "a" to count a "n".
var val1 = 989.56789;
var val2 = -989.8689;
var val3 = 989;
var a = val1.toPrecision(7);
var b = val2.toPrecision(6);
var c = val3.toPrecision(7);
/*
a = 989.5679
b = -989.869
c = 989.0000
*/
```

Math methods

- ceil(x)
- floor(x)
- round(x)
- trunc(x)
- max(x, y, z, ..., n)
- min(x, y, z, ..., n)
- sqrt(x)
- cbrt(x)
- pow(x, y)
- random()
- abs(x)
- PI

```
var a = Math.ceil(2.56789); // give ceiling value
var b = Math.floor(2.56789); // give flooring value
var c0 = Math.round(2.43); // give rounded value
var c1 = Math.round(2.45);
var c2 = Math.round(2.5);
var c3 = Math.round(2.59);
var c4 = Math.round(2.54);
var d1 = Math.trunc(2.56789); // give only truncated value (trunc after ".")
var d2 = Math.trunc(9.99998);
/* a = 3, b = 2, c0 = 2, c1 = 2, c2 = 3, c3 = 3, c4 = 3, d1 = 2, d2 = 9 */
```

```
var a = Math.max(12, 345, 45, -234, 23, 29737);
var b = Math.min(12, -345, 45, -234, -23, 29737);
/* a = 29737, b = -345 */
```

```
var a = Math.sqrt(64); // give square root. (works only for positive numbers. If we enter negative number then we wiil get "NaN (Not a Number)")
var b = Math.cbrt(-64); // give cube root. (works only for positive & negative numbers.)
var c = Math.pow(-2, 5); // c = -2^5 = -32;
/* a = 8, b = -4, c = -32*/
```

```
var a = Math.random(); // it gives random number
var b1 = Math.abs(-98); // it gives positive number Or. 0
var b2 = Math.abs(-293.238);
var b3 = Math.abs(293.238);
var b4 = Math.abs(null); //special case
var c = Math.PI; // retunrs PI's value = 3.14....
/* a = 0.4110157029520962, b1 = 98, b2 = 293.238, b3 = 293.238, b4 = 0, c = 3.141592653589793 */
```

Date methods

- toDateString()
- getDate()
- getFullYear()
- getMonth()
- getDay()

- getHours()
- getMinutes()
- getSeconds()
- **getMilliseconds()**

- setDate()
- setFullYear()
- setHours()
- setMilliseconds()
- setMinutes()
- setMonth()
- setSeconds()

- By this line date variable become a Object for all Date's values.

```
var date = new Date();
```

- **Date**

```
var dateStr = date.toDateString();
var getDate = date.getDate();
var fullYear = date.getFullYear();
var month = date.getMonth(); // 0 = January, ... , 11 = December
var Day = date.getDay(); // 0 = sunday , ... ,6 = Saturday

/*
   date = Fri May 06 2022 12:31:07 GMT+0530 (India Standard Time)
   dateStr = Fri May 06 2022
   getDate = 6
   fullYear = 2022
   month = 4
   Day = 5
*/
```

- **Time**

```
var Time = date.getTime(); // it returns "whole time" in to the milliseconds
var hrs = date.getHours();
var min = date.getMinutes();
var sec = date.getSeconds();
var milliSec = date.getMilliseconds();// it returns seconds in to the milliseconds

/*
   Time = 1651820870490
   hrs = 12
   min = 37
   sec = 50
   milliSec = 490
*/
```

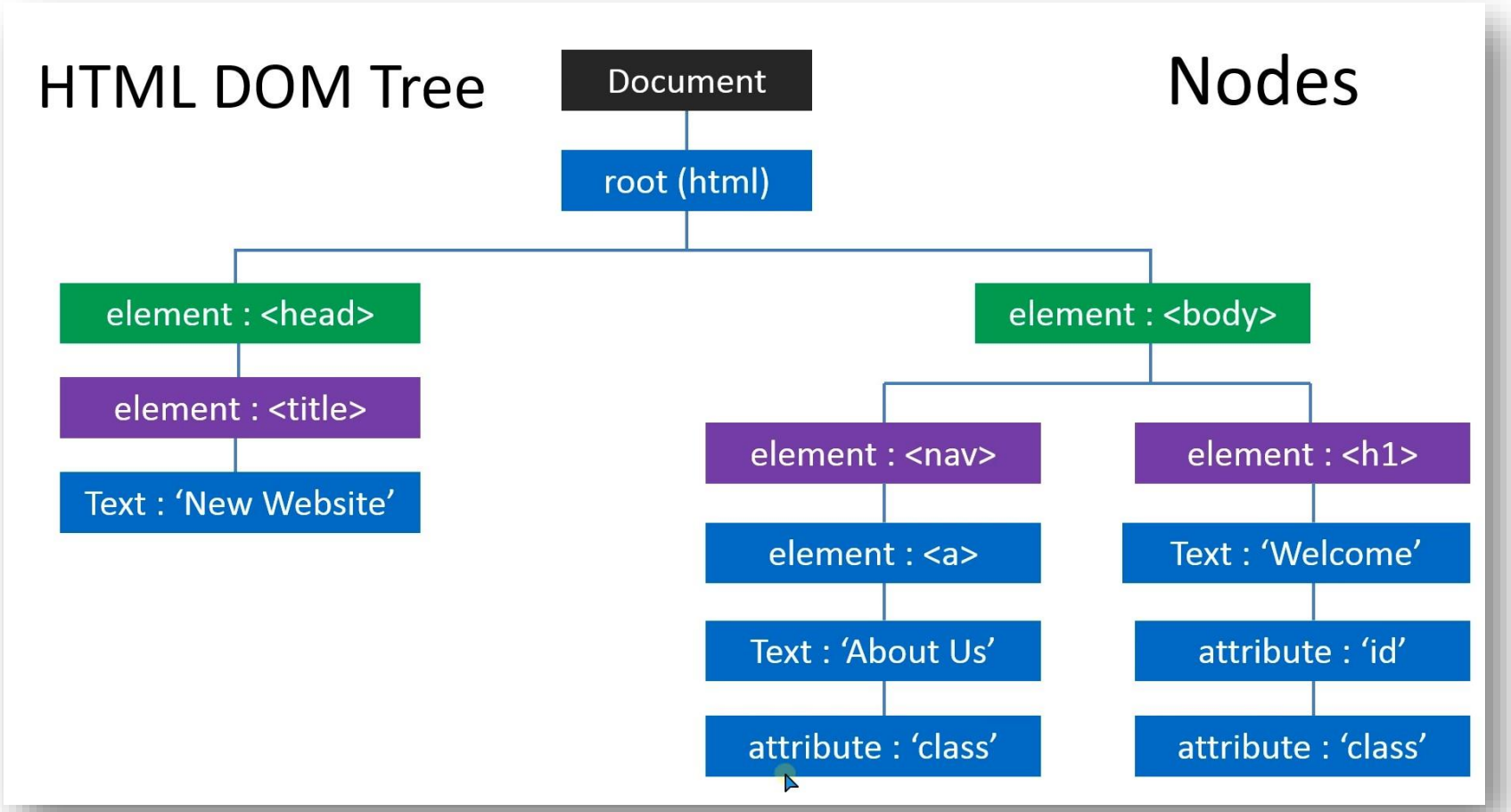
- **Special case**

```
var date = new Date("February 10 2025");
```

✓ Date returns all the data based on "February 10 2025".

```
/*
   date = Mon Feb 10 2025 00:00:00 GMT+0530 (India Standard Time)

   date.toDateString() = Mon Feb 10 2025
   date.getDate() = 10
   date.getFullYear() = 2025
   date.getMonth() = 1
   date.getDay() = 1

   // time always display 0. (for future Date Or. Past Date)
   date.getTime() = 1739125800000(get in milliseconds)
   date.getHours() = 0
   date.getMinutes() = 0
   date.getSeconds() = 0
   date.getMilliseconds() = 0
*/
```

DOM – Document Object Module



- By DOM we can – (Only do 4 operations)

- Get values of attributes
- Set values of attributes
- Add new HTML Element
- Delete HTML Element

DOM – targeting

Other DOM Targeting Methods :

- document
- document.all
- document.documentElement
- document.head
- document.title
- document.body
- document.images
- document.anchors
- document.links
- document.forms
- document.doctype
- document.URL
- document.baseURI
- document.domain

How to Target DOM Object :

- Id → document.getElementById(id)
- Class Name → document.getElementsByClassName(name)
- Tag Name → document.getElementsByTagName(name)

MAJORLY USED FROM TARGETING POINT OF YOU

- ✓ getElementById() always returns string.
- ✓ getElementsByClassName() & getElementsByTagName() always return array.

- ✓ document – return a <!Document html> whole tag
- ✓ document.all – return array which is consist all tags which are exists in a document
- ✓ document.documentElement – return <html>
- ✓ document.head – return <head>
- ✓ document.title – return title name
- ✓ document.body – return <body>
- ✓ document.images – return array which is consist all <img> tags which are exists in a document
- ✓ document.links – return array which is consist all <a> tags which are exists in a document
- ✓ document.forms – return array which is consist all <form> tags which are exists in a document
- ✓ document.doctype – return string like this … → <!Document html>
- ✓ document.URL – return a whole current URL
- ✓ document.baseURI – as same as Above
- ✓ document.domain – return website name e.g. www.abcd.in

❖ how to write ?

- var element = document;
- var element = document.body;
- var element = document.forms;

DOM – get, set, add HTML tag, Remove HTML tag

DOM Get Methods :

- innerText
- innerHTML
- getAttribute
- getAttributeNode
- Attributes

DOM Set Methods :

- innerText
- innerHTML
- setAttribute
- Attribute
- removeAttribute

#remark : these all methods are written after target methods.

Get methods

- ✓ innerText – return a text of targeted element.
- ✓ innerHTML – return text with HTML TAGS of targeted element.
- ✓ getAttribute(“attributeName”) – return attribute value.
- ➢ e.g.

document.getElementById("main").getAttribute("id") → “main”

document.getElementById("main").getAttribute(“class") → “abc”

document.getElementById("main").getAttribute(“style") → “border: .1rem solid red;”

- ✓ attributes – return array which is consist ALL ATTRIBUTES which are exists in a targeted HTML tag
- ➢ e.g.

document.getElementById("main").attributes → [“id” , “class” , “style”] // return array

document.getElementById("main").attributes[0] → id= "main"

document.getElementById("main").attributes[1] → class= "abc"

document.getElementById("main").attributes[2] → style= "border: .1rem solid red; "

document.getElementById("main").attributes[0].name → "id"

document.getElementById("main").attributes[1]. name → "class"

document.getElementById("main").attributes[2]. name → "style“

document.getElementById("main").attributes[0].value → “main"

document.getElementById("main").attributes[1]. value → “abc"

document.getElementById("main").attributes[2]. value → " border: .1rem solid red; "

✓ `getAttributeNode`(“attributeName”) – return attribute Name & Value.

➢ e.g.

document.getElementById("main").`getAttributeNode`("id") → id= "main"

document.getElementById("main").`getAttributeNode`(“class") → class= "abc"

document.getElementById("main").`getAttributeNode`(“style") → style= "border: .1rem solid red; "

document.getElementById("main"). `getAttributeNode`("id") .name → "id"

document.getElementById("main").`getAttributeNode`(“class") . name → "class"

document.getElementById("main").`getAttributeNode`(“style") . name → "style“

document.getElementById("main"). `getAttributeNode`("id") .value → “main"

document.getElementById("main").`getAttributeNode`(“class") . value → “abc"

document.getElementById("main").`getAttributeNode`(“style") . value → " border: .1rem solid red; "

Set methods

- ✓ innerText – change text of targeted element
- ✓ innerHTML – change text with HTML TAGS of targeted element.
- ➢ e.g.
- ➢ document.getElementById("main").innerText = "this is my first div."
- ➢ document.getElementById("main").innerHTML = "<h3>this is my first div.</h3>"
- ✓ setAttribute(“attributeName”) – change attribute value.
- ➢ e.g.

document.getElementById("main"). setAttribute("id") = “parent”

document.getElementById("main"). setAttribute(“class") = “xyz”

document.getElementById("main"). setAttribute(“style") = “border: .2rem dotted black;”

- ✓ attributes – return array which is consist ALL ATTRIBUTES which are exists in a targeted HTML tag
- ➢ e.g.

document.getElementById("main").attributes → [“id” , “class” , “style”] // return array

document.getElementById("main").attributes[0] → id = "main"

document.getElementById("main").attributes[1] → class = "abc"

document.getElementById("main").attributes[2] → style = "border: .1rem solid red; “

document.getElementById("main").attributes[0].value = “parent"

document.getElementById("main").attributes[1]. value = “xyz"

document.getElementById("main").attributes[2]. value = " border: .2rem dotted yellow; "

✓ `removeAttribute`(“attributeName”) – remove attribute

➢ e.g.

```
<div id="main" class="abc"> this is my first container </div>

.abc{
color : green;
}

document.getElementById("main").removeAttribute(“class");
```

After :

```
<div id="main" class="abc"> this is my first container </div>
```

- After removing class, text lost it’s green color.

querySelector() & querySelectorAll()

In case of class return a first occurance

document.querySelector(); → always returns string

document.querySelectorAll(); → always returns array

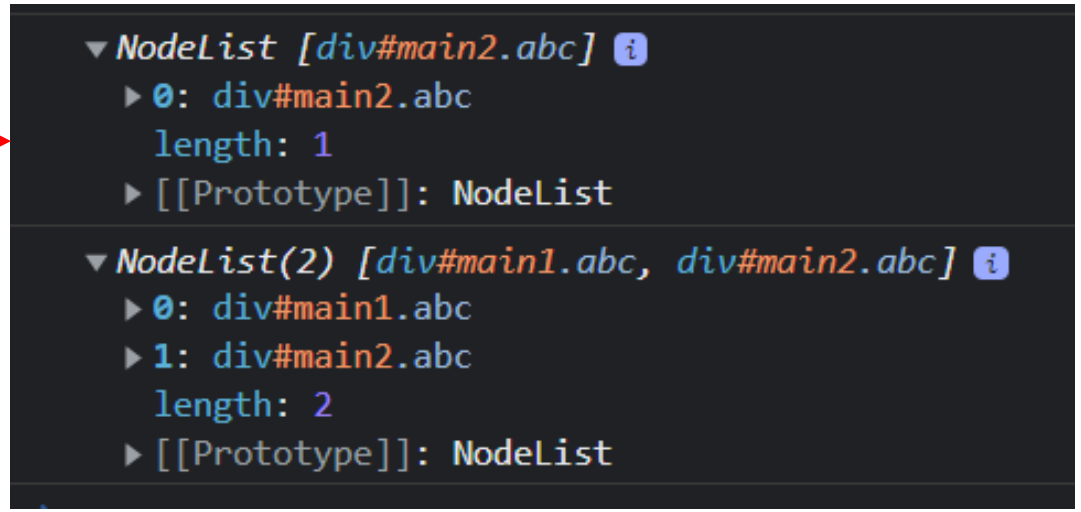
```
<div id="main1" class="abc" style="border: .1rem solid red;">My first div.</div>
<div id="main2" class="abc" style="border: .1rem dotted blue;">My second div.</div>
```

```
<div id="main1" class="abc" style="border: .1rem solid red;">My first div.</div>
<div id="main1" class="abc" style="border: .1rem solid red;">My first div.</div>
```

returns a first occurance

```
document.querySelector("#main1");
document.querySelector(".abc");
```

```
document.querySelectorAll("#main2");
document.querySelectorAll(".abc");
```



DOM – CSS styling methods


```
<div id="main1" class="abc" style="border: .1rem solid red;">My first div.</div>
<div id="main2" class="abc xyz efg" style="border: .1rem dotted blue;">My second div.</div>
```

```
document.getElementById("main1").style.border //0.1rem solid red

document.getElementById("main1").style.backgroundColor = "yellow"; // set background color yellow of "mian1" div
```

```
document.getElementById("main2").className

document.getElementById("main2").classList
```

```
abc xyz efg
```

Return string

```
▼DOMTokenList(3) ['abc', 'xyz', 'efg', value: 'abc xyz efg']
   0: "abc"
   1: "xyz"
   2: "efg"
   length: 3
   value: "abc xyz efg"
  ▶[[Prototype]]: DOMTokenList
```

Return array

```
<div id="main2" class="abc xyz efg" style="border: .1rem dotted blue;">My second div.</div>
```

```
document.getElementById("main2").className = "pqr mnp";
```

```
<div id="main2" class="pqr mnp" style="border: .1rem dotted blue;">My second div.</div> == $0
```

```
document.getElementById("main2").classList.add("pqrst");
```

```
<div id="main2" class="abc xyz efg pqrst" style="border: .1rem dotted blue;">My second div.</div>
```

```
document.getElementById("main2").classList.remove("abc");
```

```
<div id="main2" class="xyz efg pqrst" style="border: .1rem dotted blue;">My second div.</div> ==
```

DOM – addEventListner() & removeEventListner()

Assign Events Using the HTML DOM :

```
document.getElementById(Id).onclick = functionName;
```

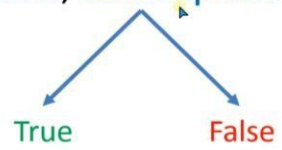
```
document.getElementById("main1").onclick = functionName1;
function functionName1() {
    // write statements
};
document.getElementById("main1").onmouseover = functionName2;
function functionName2() {
    // write statements
};

/* using arrow function */
document.getElementById("main1").onclick = () => {
    // write statements
}
```

DOM addEventListener() Method :

```
document.getElementById(Id). addEventListener("click", functionName);

document.getElementById(Id). addEventListener("click", function(){
        Statement
});
```

UseCapture:

```
addEventListener(event, function, useCapture);
```



DOM removeEventListener() Method :

```
element.removeEventListener("ondblclick" ,functionName);
```

```html
<div id="main1" class="abc" style="border: .1rem solid red;">My first div.</div>
<div id="main2" class="abc xyz efg" style="border: .1rem dotted blue;">My second div.</div>
```

```js
document.getElementById("main1").addEventListener("click", fun1);
document.getElementById("main2").addEventListener("mouseover", fun2);
function fun1() {
    console.log("fun1() is called");
}
function fun2() {
    console.log(document.getElementById("main2").style.boeder);
    // console.log(this.style.border); --> this = document.getElementById("main2")
}

/* using arrow function -----------------*/
document.getElementById("main1").addEventListener("click", () => {
    console.log("fun1() is called");
});

document.getElementById("main2").addEventListener("mouseover", () => {
    console.log("fun2() is called");
});
```

- This line is #imp
- See, the use of ***this*** operator.
- ***this*** operator don't work in an arrow function.

- *removeEventListner()*

```
1) first of all click on div "main1".
2) then you can see that mouseover event is not worked.
▪ (But you do mouse hover on div "main1" before click on it then it is worked.)
```

```
document.getElementById("main1").addEventListener("mouseover", fun);
document.getElementById("main1").addEventListener("click", removeFun);

function fun() { console.log("fun() is called"); }

function removeFun() {
    this.removeEventListener("mouseover", fun);
    alert("Now, mouseOver Event is not worked");
}
```

DOM – classList() Methods

- add(class1, class2, ...)
- remove(class1, class2, ...)
- toggle(class)
- contains(class)
- item(index)
- Length

```
<div id="main2" class="abc xyz efg" style="border: .1rem dotted blue;">My second div.</div>
```

```
/*add class*/
document.getElementById("main2").classList.add("pqr");/* class="abc xyz efg pqr" */

/*remove class*/
document.getElementById("main2").classList.remove("xyz");/* class="abc efg pqr" */

/*return total no.of class = 3(in our case)*/
document.getElementById("main2").classList.length;
```

```html
<div id="main2" class="abc xyz efg" style="border: .1rem dotted blue;">My second div.</div>
```

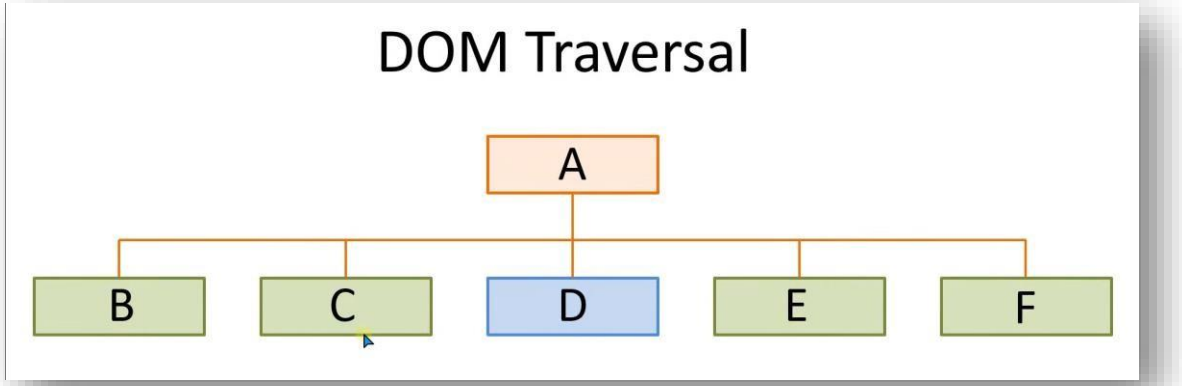
```javascript
/* return element.
classList.contains(index)
*/
document.getElementById("main2").classList.item(0);/* abc */
document.getElementById("main2").classList.item(1);/* xyz */
document.getElementById("main2").classList.item(2);/* efg */

/*you can't set like this --> give error*/
// document.getElementById("main2").classList.item(1) = "ok"
```

```javascript
/* return true(if class exist) or false(otherwise) */

document.getElementById("main2").classList.contains("abc");/* true */

document.getElementById("main2").classList.contains("ABC");/* false */
```

```
/*
❑ Toggle method

▪ by using this method we can add & remove "classes" on every click
```

DOM – traversal Methods

- parentNode
- ParentElement
- Children
- childNodes
- firstChild
- firstElementChild
- lastChild
- lastElementChild
- nextElementSibling
- nextSibling
- previousElementSibling
- previousSibling



- Some important terms :

✓ Parent node of (B, C, D, E, F) : A

✓ Child nodes of ‘A’ : (B, C, D, E, F)

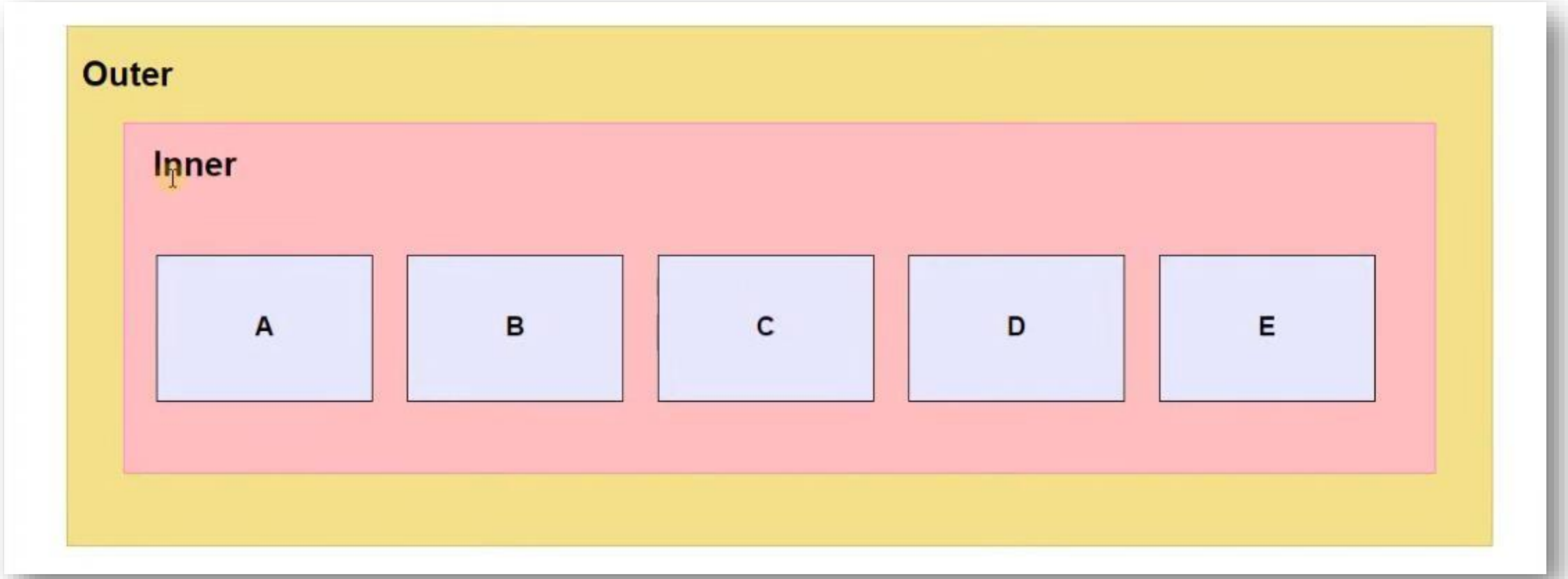
✓ First child node of ‘A’ : B

✓ Last child node of ‘A’ : F

✓ Siblings : (B, C, D, E, F)

✓ Previous sibling of ‘D’ : C

✓ Next sibling of ‘D’ : E


Outer
Inner
A
B
C
D
E

```html
<!DOCTYPE html>
<html id="main">
<head>
    <title>DOM Navigation</title>
</head>
<body>
    <div id="outer">
        <h2>Outer</h2>
        <div id="inner">
            <h2 id="child-head">Inner</h2>
            <div id="child-A">A</div>
            <div id="child-B">B</div>
            <div id="child-C">C</div>
            <div id="child-D">D</div>
            <div id="child-E">E</div>
        </div>
    </div>
</body>
</html>
```

```css
<style>
    #outer{
        width: 550px;
        height: 300px;
        padding:10px 10px;
        margin: 0 auto;
        background: lightsalmon;
    }
    #inner{
       width: 500px;
        height: 200px;
        padding:10px 10px;
        margin:0 auto ;
        background: mediumorchid;
    }
    #inner div{
        display: inline-block;
        background: #fff;
        width: 95px;
        height: 50px;
        line-height: 50px;
        text-align: center;
    }
</style>
```

- Here ,

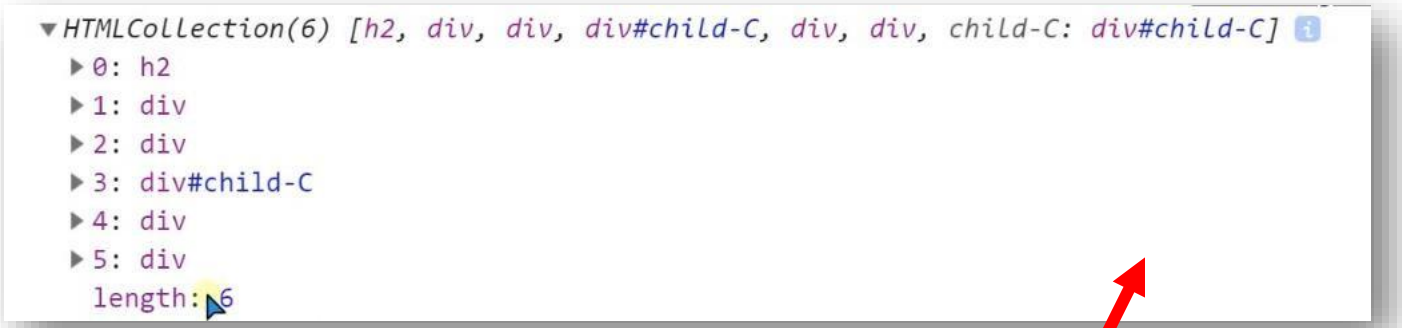
```
<html id="main"> ... </html>
```

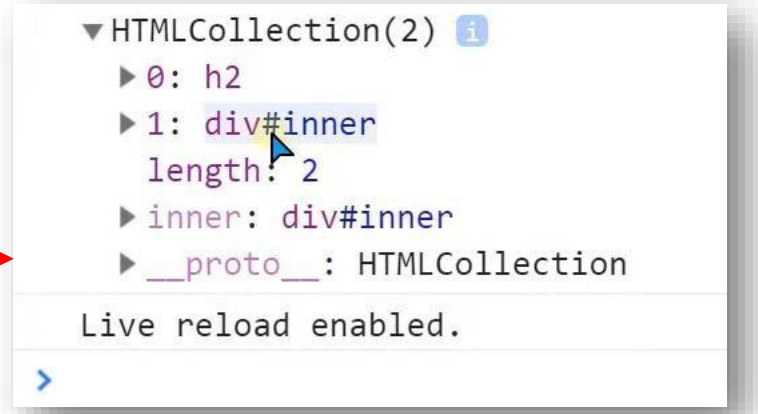
```
/*

difference between parentNode & parentElement
parentElement : if targeted element hasn't child node then give null
parentNode : if targeted element hasn't child node then #document (means return whole document)

e.g.
document.getElementById("main").parentNode; // return #document (means return whole document)
document.getElementById("main").parentElement; // return null

*/
```

```
var a = document.getElementById("inner").parentElement; // return <div id="outer"></div>
var a = document.getElementById("outer").parentElement; // return <body></body>
var a = document.body.parentElement; // return <html></html>


// change the background color of <div id="outer"></div>
var a = document.getElementById("inner").parentElement.style.background = "red";
```

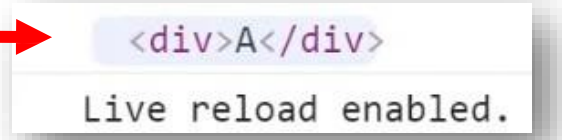
```
// change the background color of <div id="inner"></div>
document.getElementById("child-c").parentElement.style.background = "red";
document.getElementById("main").parentNode; // return #document (means return whole document)
```

DOM – children & childNodes






```
var a = document.getElementById("inner").children;
var a = document.getElementById("outer").children;
var a = document.getElementById("inner").children[1];
```




```
//change the background color of <div>A</div>
document.getElementById("inner").children[1].style.background = "red";
```

```
/*
▪ children() consider childnode to a HTML tag.
▪ childNode() consider childnode to a text(space/characters), Enter key, HTML tags.
▪ both return array.


*/
```

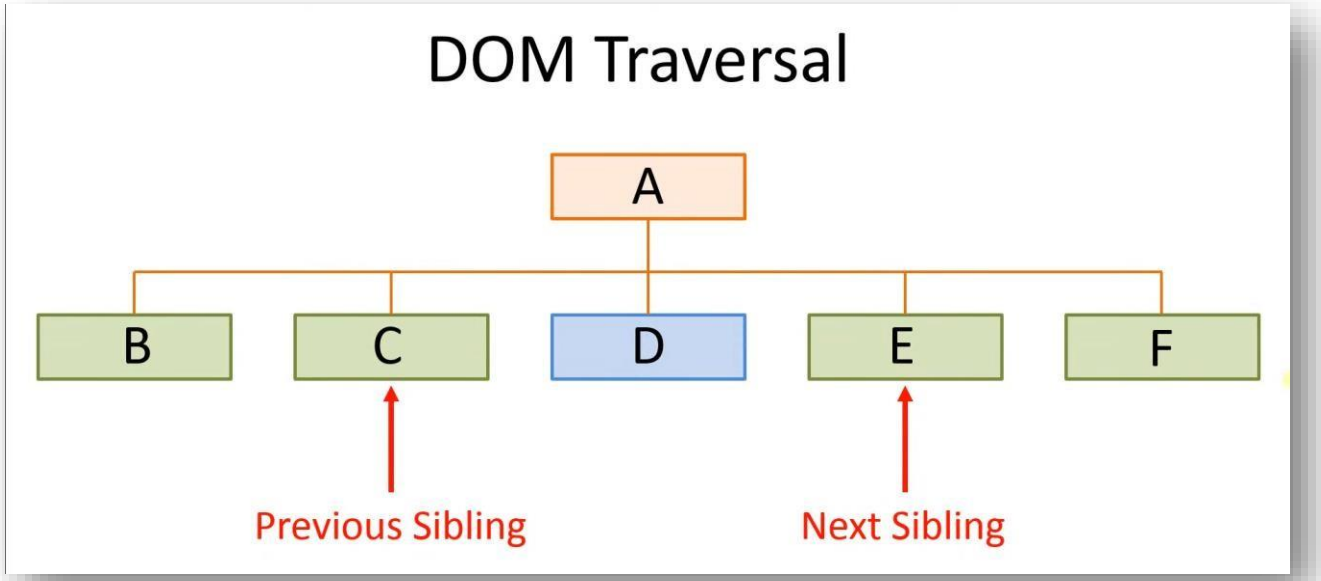
DOM – firstElementChild, lastElementChild ,firstChild & lastChild

```
//return <div id="header"> Inner </div>
var a = document.getElementById("inner").firstElementChild;
//return "Inner"
var a = document.getElementById("inner").firstElementChild.innerHTML;
// change the bg-color of <div id="header"> Inner </div>
document.getElementById("inner").firstElementChild.style.background = "red";
```

```
//return <div> E </div>
var a = document.getElementById("inner").lastElementChild;
//return "E"
var a = document.getElementById("inner"). lastElementChild.innerHTML;
// change the bg-color of <div id="inner"> </div>
document.getElementById("outer").lastElementChild.style.background = "red";
```

```
/*
✓ firstElementChild() & lastElementChild() consider childnode to only a HTML tag.
✓ firstChild() & lastChild() consider childnode to both (a text & a HTML tag.)
✓ firstElementChild() & lastElementChild() return HTML tag.
✓ firstChild() & lastChild() return text(character/space/Enter) or HTML tag.

*/
```

DOM – previousElementSibling, nextElementSibling , previousSibling & nextSibling



- Some important terms :
- ✓ Previous sibling of ‘D’ : C
- ✓ Next sibling of ‘D’ : E
- ✓ Previous sibling of ‘B’ : ***null***
- ✓ Next sibling of ‘F’ : ***null***

```
// return <div>B</div>
var a = document.getElementById("child-C").previousElementSibling;
// return "B"
var a = document.getElementById("child-C").previousElementSibling.innerHTML;
// change bg-color of <div>B</div>
document.getElementById("child-C").previousElementSibling.style.background = "red";
```

```
// return <div>D</div>
var a = document.getElementById("child-C").nextElementSibling;
// return “D"
var a = document.getElementById("child-C").nextElementSibling.innerHTML;
// change bg-color of <div>D</div>
document.getElementById("child-C").previousElementSibling.style.background = "red";
```

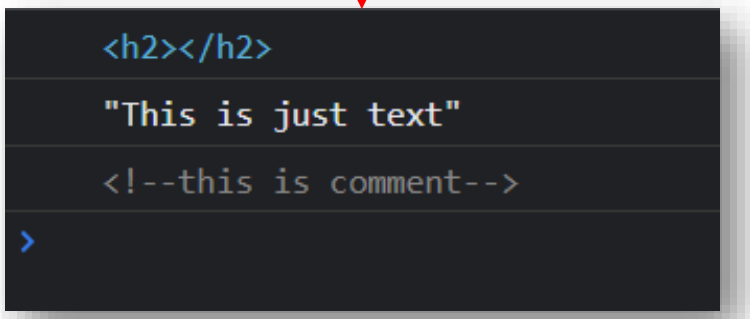
```
//return null
var a = document.getElementById("child-E").nextElementSibling;
var a = document.getElementById("child-head").previousElementSibling; //#imp
var a = document.getElementById("chld-A").previousElementSibling;
```

```
/*
✓ previousElementSibling & nextElementSibling consider sibling to only a HTML tag.
✓ previousSibling & nextSibling consider siblings to both (a text & a HTML tag.)
✓ previousElementSibling & nextElementSibling return HTML tag.
✓ previousSibling & nextSibling return text(character/space/Enter) or HTML tag.


*/
```

DOM – create methods

- createElement
- createTextNode
- createComment

```
/*create a new HTML Tag*/
var newElement = document.createElement("h2");
/*create a new Text Node*/
var newText = document.createTextNode("This is just text");
/*create a new comment*/
var newComment = document.createComment("this is comment");
```



DOM – appendChild & insertBefore

- appendChild
- insertBefore

- ***appendChild** always add new element or textNode at the end of the element*
- ***By insertBefore we can add new element Or. New TextNode at any position of element***

```
/* By appendChild append a TextNode in a HTML Tag.*/

var newElement = document.createElement("h2");
var newText = document.createTextNode("This is just text");
newElement.appendChild(newText);
console.log(newElement);
```

```
<h2>This is just text</h2>
```

```
/* By appendChild append a HTML tag in a Another HTML Tag.*/


<div id="test">
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos vel consequatur.
        Delectusvoluptas dolorem id exercitationem</p>
    <h3>yahoo baba</h3>
</div>

<script>
    document.getElementById("test").appendChild(newElement);
</script>
```

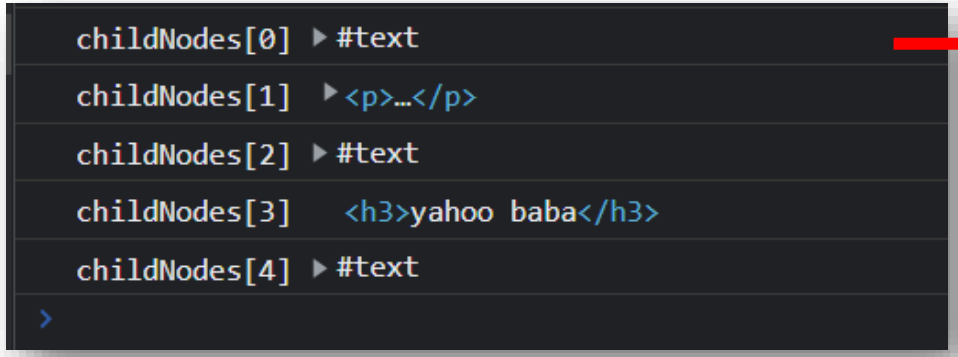
```
<h2>This is just text</h2>
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos vel consequatur. Delectus voluptas dolorem id exercitationem

yahoo baba

This is just text

- insertBefore

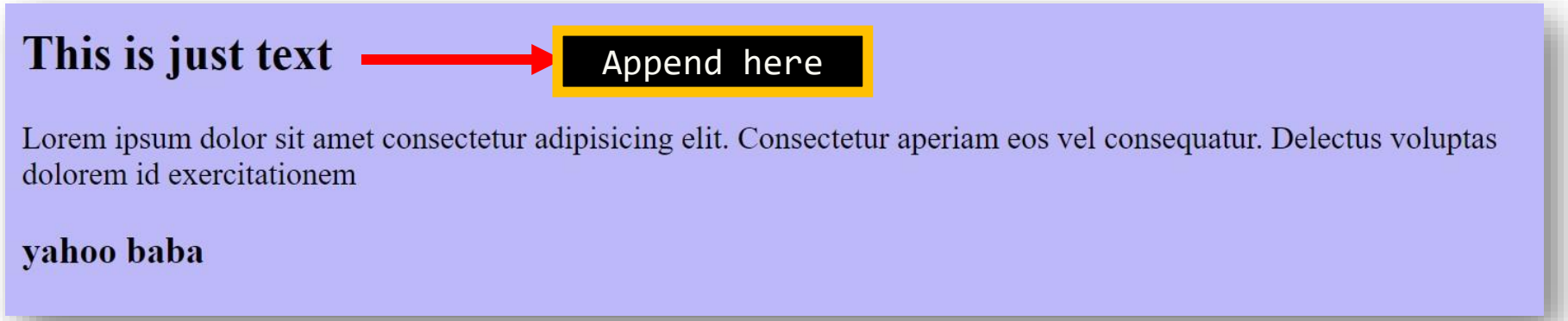


Structure of childElements of the ...

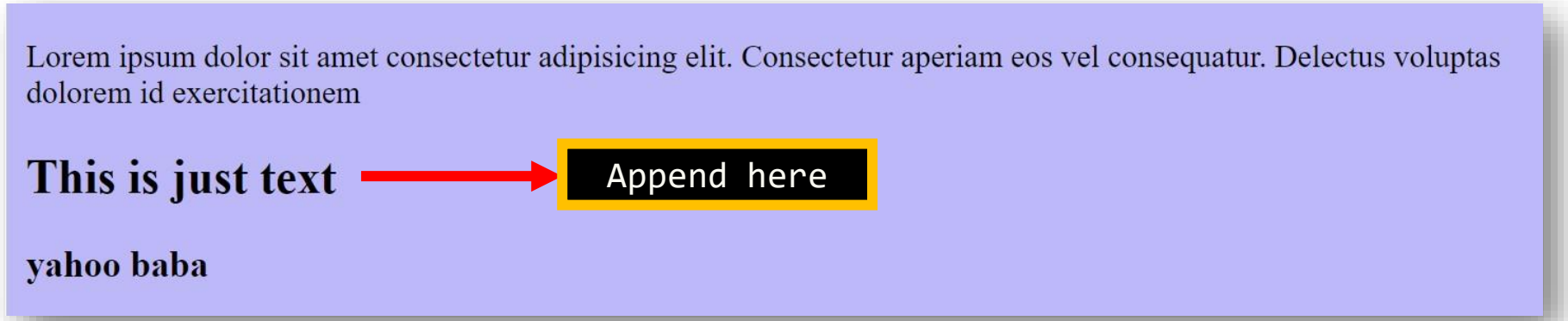
<h2>This is just text</h2> → newElement

```
/* By insertBefore append a HTML tag in a Another HTML Tag.*/
<div id="test">
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos vel consequatur.
        Delectusvoluptas dolorem id exercitationem</p>
    <h3>yahoo baba</h3>
</div>
<script>
    var target = document.getElementById("test");
    target.insertBefore(newElement, target.childNodes[0]);
    target.insertBefore(newElement, target.childNodes[1]);
    target.insertBefore(newElement, target.childNodes[2]);
    target.insertBefore(newElement, target.childNodes[3]);
    target.insertBefore(newElement, target.childNodes[4]);
</script>
```

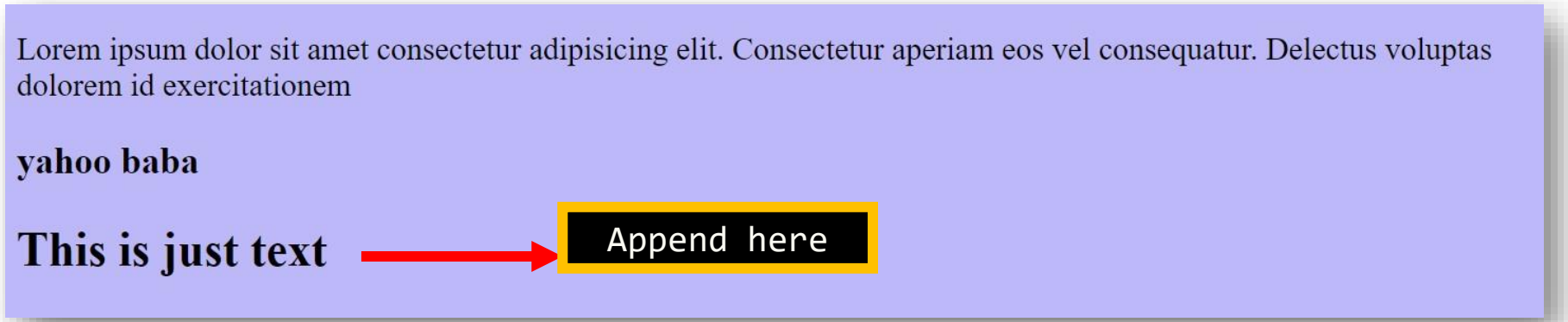
```
target.childNodes[0]
target.childNodes[1]
Output for this 2 cases
```



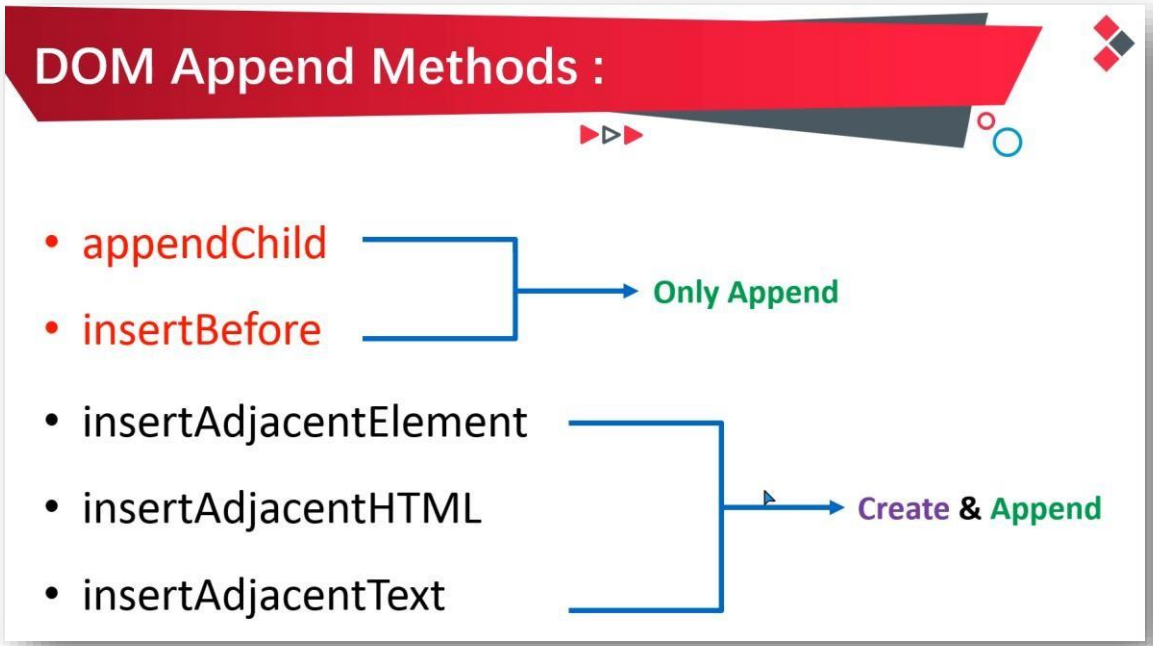

```
target.childNodes[2]
target.childNodes[3]
Output for this 2 cases
```

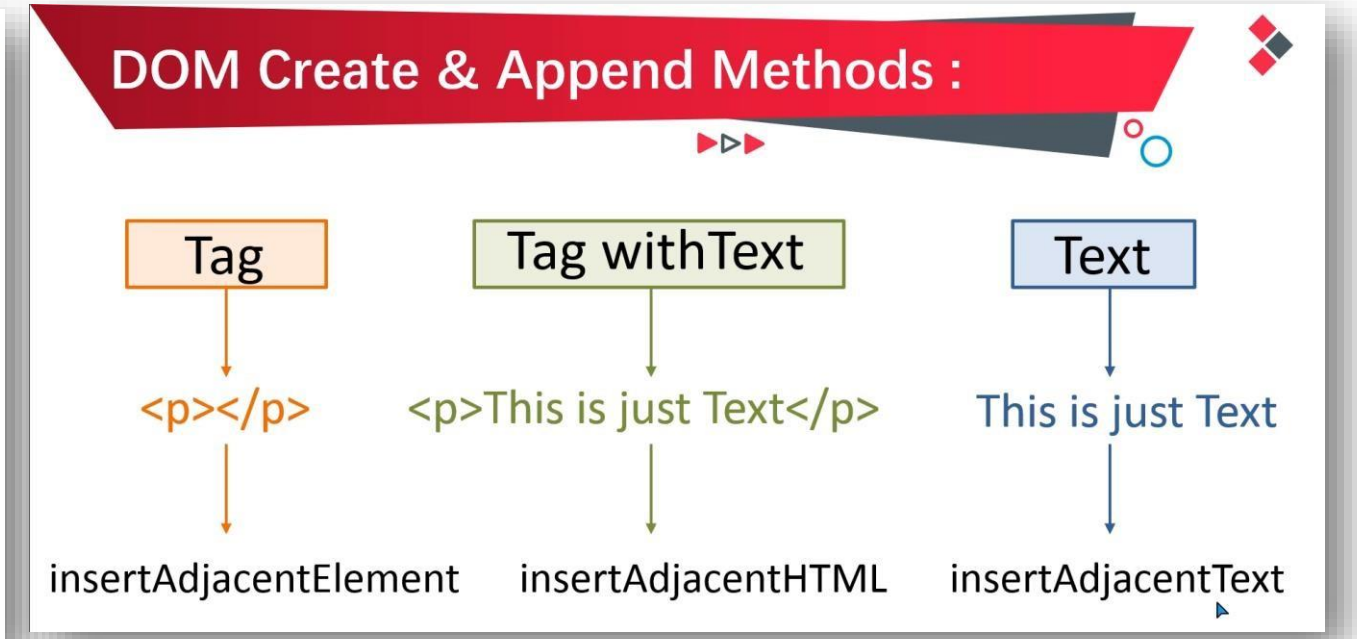



```
target.childNodes[4]
Output for this case
```

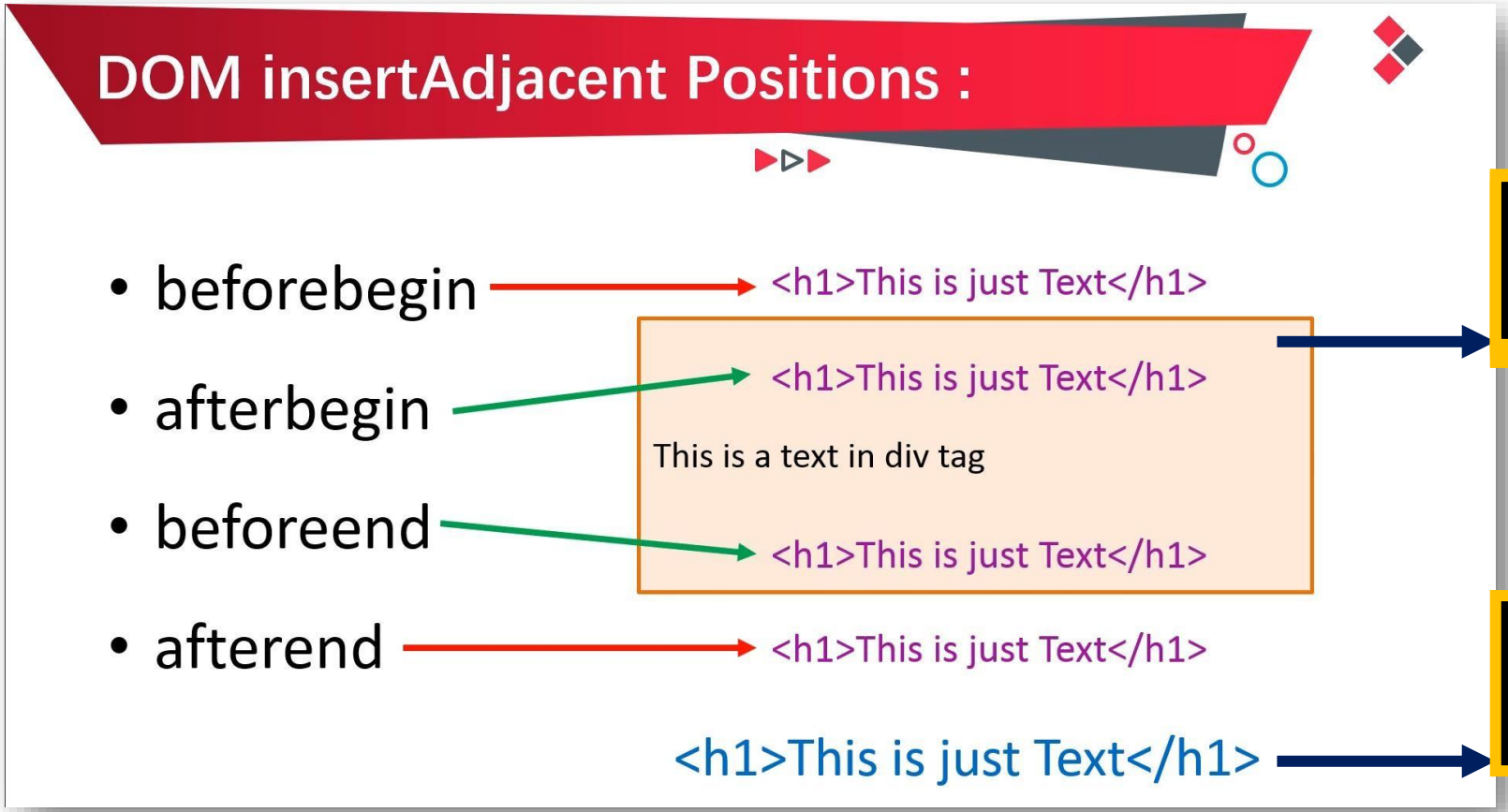


DOM – insertAdjacentElement, insertAdjacentHTML & insertAdjacentText





DOM insertAdjacent Positions



Tagrgeted div

```
<div>This is a text in a div tag.</div>
```

inserted div

```
<h1>This is just Text</h1>
```

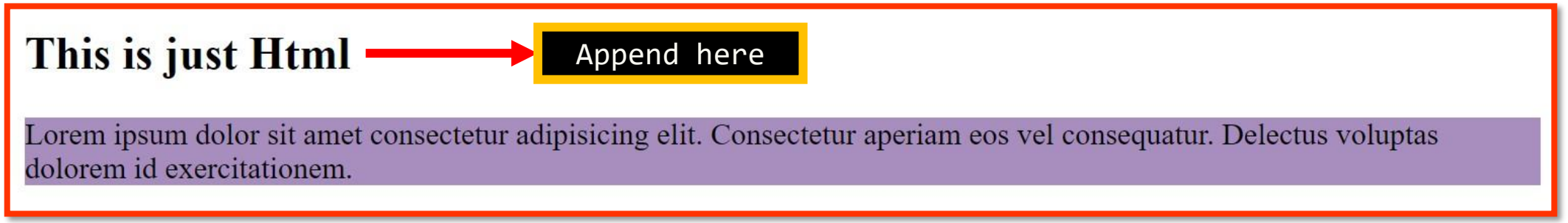
```html
<div id="test">
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos
    vel consequatur. Delectusvoluptas dolorem id exercitationem.
</div>
```

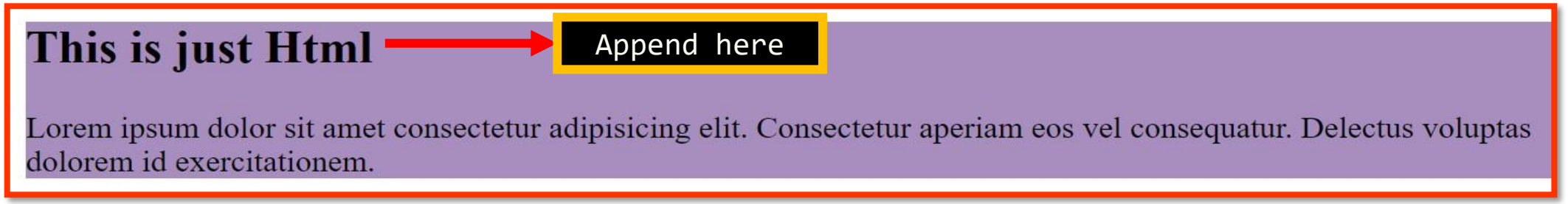
```html
<script>
    var target = document.getElementById("test");
    var newElement = "<h2>This is just Html</h2>";
    target.insertAdjacentHTML("beforebegin", newElement);
    target.insertAdjacentHTML("afterbegin", newElement);
    target.insertAdjacentHTML("beforeend", newElement);
    target.insertAdjacentHTML("afterend", newElement);
</script>
```

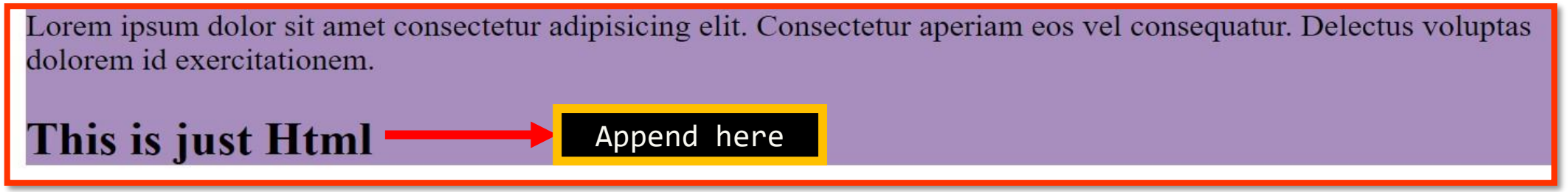
```
<script>
    var target = document.getElementById("test");
    var newTag = document.createElement("h2");
    target.insertAdjacentHTML("beforebegin", newTag);
    target.insertAdjacentHTML("afterbegin", newTag);
    target.insertAdjacentHTML("beforeend", newTag);
    target.insertAdjacentHTML("afterend", newTag);
</script>
```

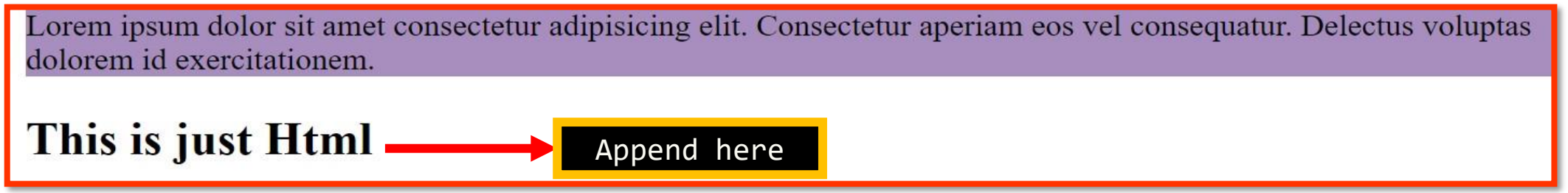
```
<script>
    var target = document.getElementById("test");
    var newText = "This a new text";
    target.insertAdjacentText("beforebegin", newText);
    target.insertAdjacentText("afterbegin", newText);
    target.insertAdjacentText("beforeend", newText);
    target.insertAdjacentText("afterend", newText);
</script>
```

insertAdjacentHTML

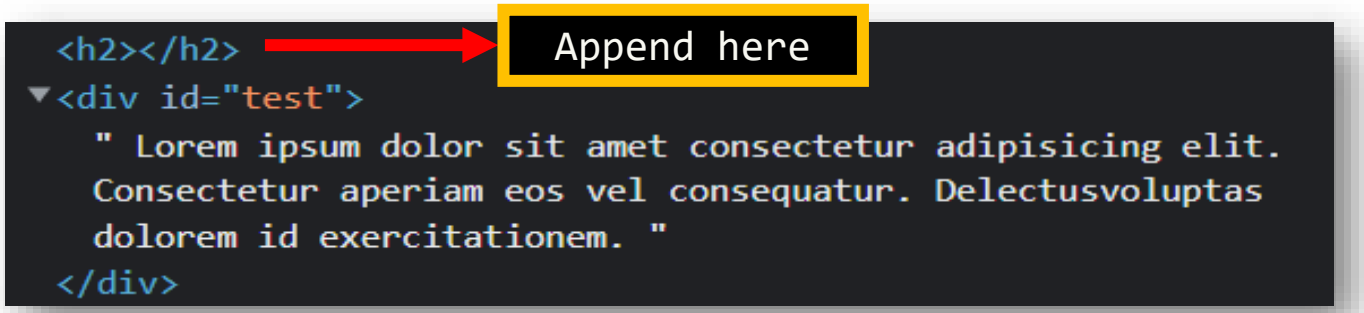


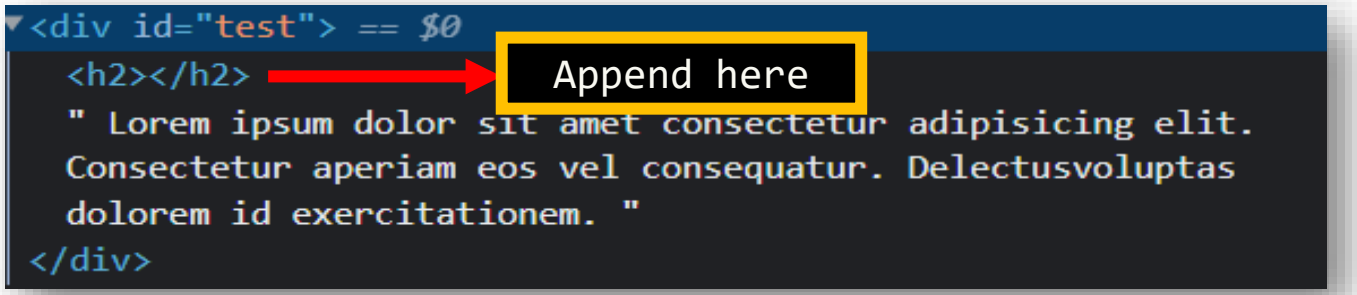


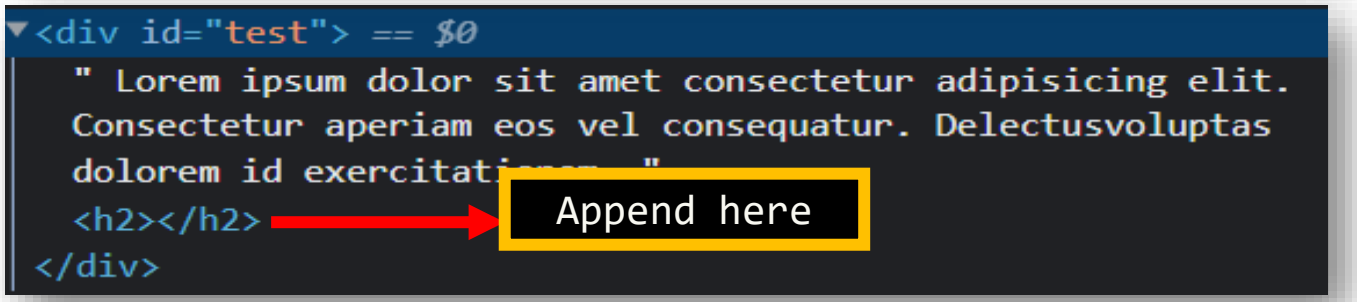


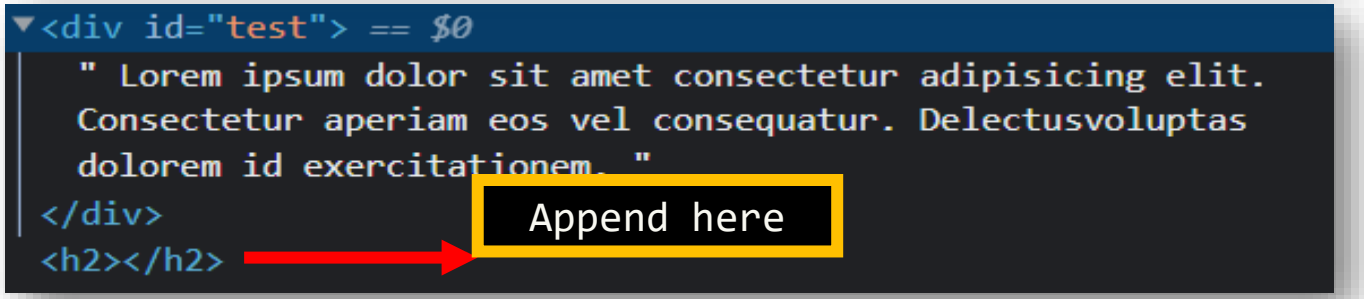


insertAdjacentElement

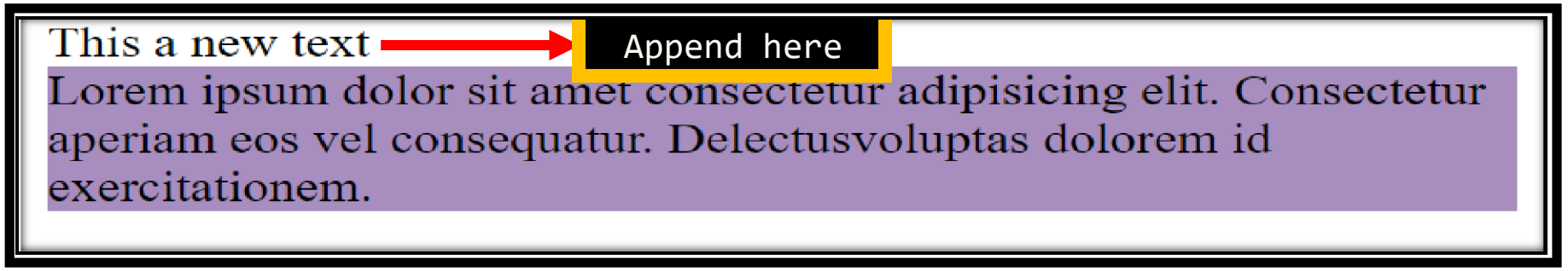


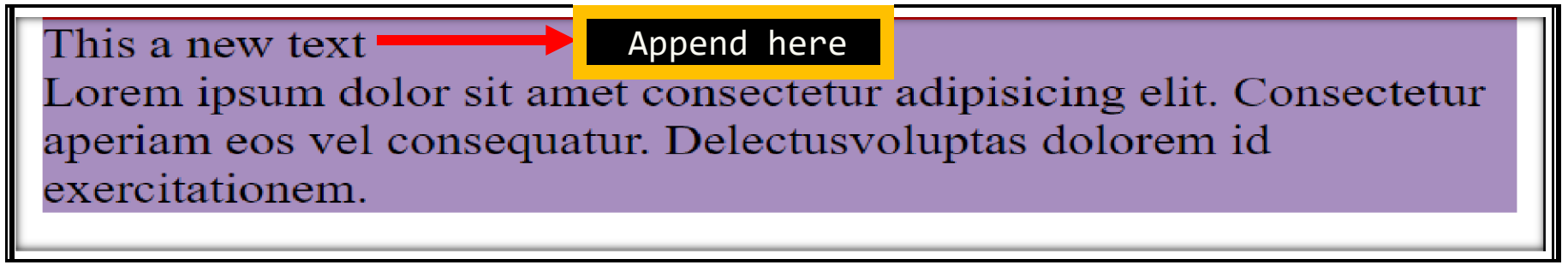






insertAdjacentText

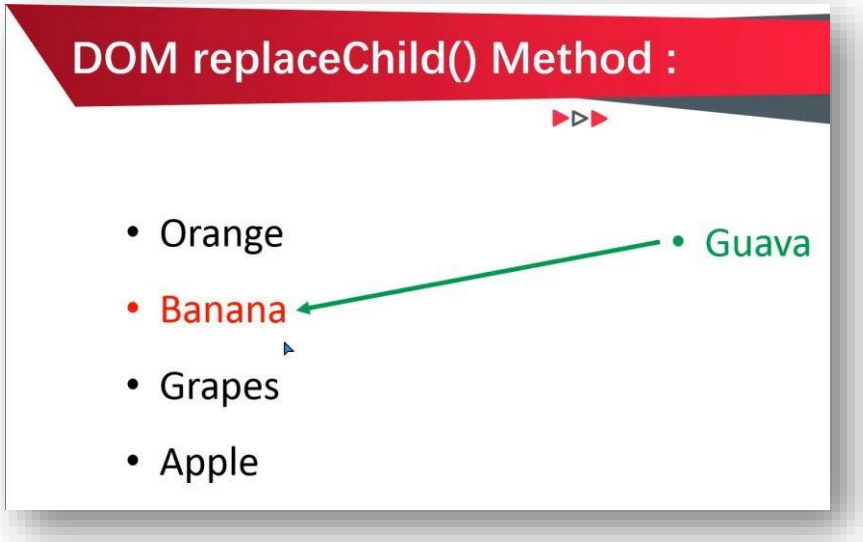




DOM – removeChild & replaceChild

- replaceChild()
- removeChild()

```
<ul id="list">
    <li class="firstItem">Orange</li>
    <li>Apple</li>
    <li>Grapes</li>
    <li>Banana</li>
</ul>
```



- replaceChild :

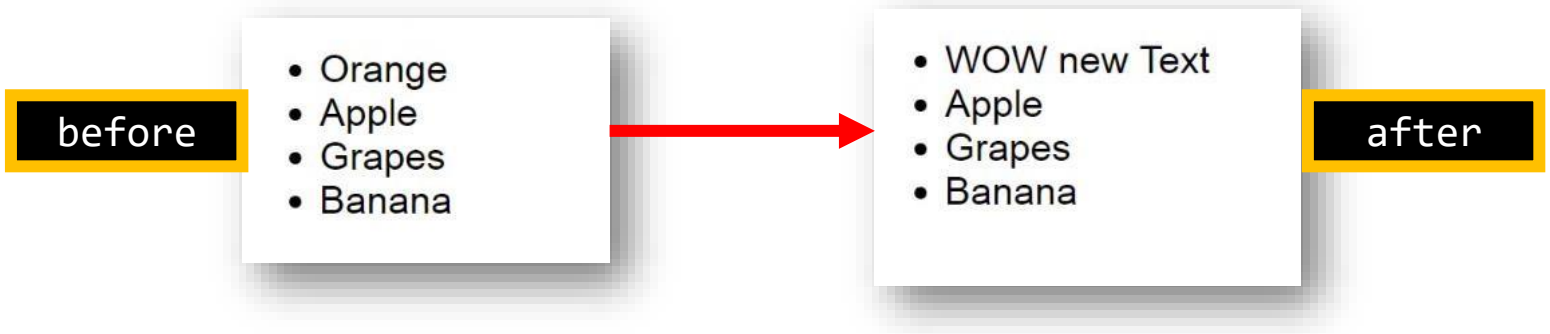
✓ Syntax :

```
parentNode.replaceChild(newElement, OldElement)
```

```
var newElement = document.createElement("li");
var newText = document.createTextNode("WOW new Text");
newElement.appendChild(newText);

var target = document.getElementById("list");
var oldElement = target.children[0];

target.replaceChild(newElement, oldElement);
```

 WOW new Text

 Orange



- removeChild :

✓ Syntax :

```
parentNode.removeChild(targetedElement);
```

```
var target = document.getElementById("list");
var oldElement = target.children[0];
target.removeChild(oldElement);
```

```
<li class="firstItem">Orange</li>
```

before

- Orange
- Apple
- Grapes
- Banana

- Apple
- Grapes
- Banana

after

DOM – cloneNode

- Orange
- Banana
- Grapes
- Apple

- Carrot
- Reddish
- Grapes

- cloneNode :

✓ Syntax :

```
parentNode.cloneNode(true/false)
```

```
var target = document.getElementById("list1").children[0];
var copyElement = target.cloneNode(true);
var copyElement = target.cloneNode(false);
```

```
<li class="firstItem"></li>
```

Flase –
- returns only HTML Tag & Attributes

```
<li class="firstItem">orange</li>
```

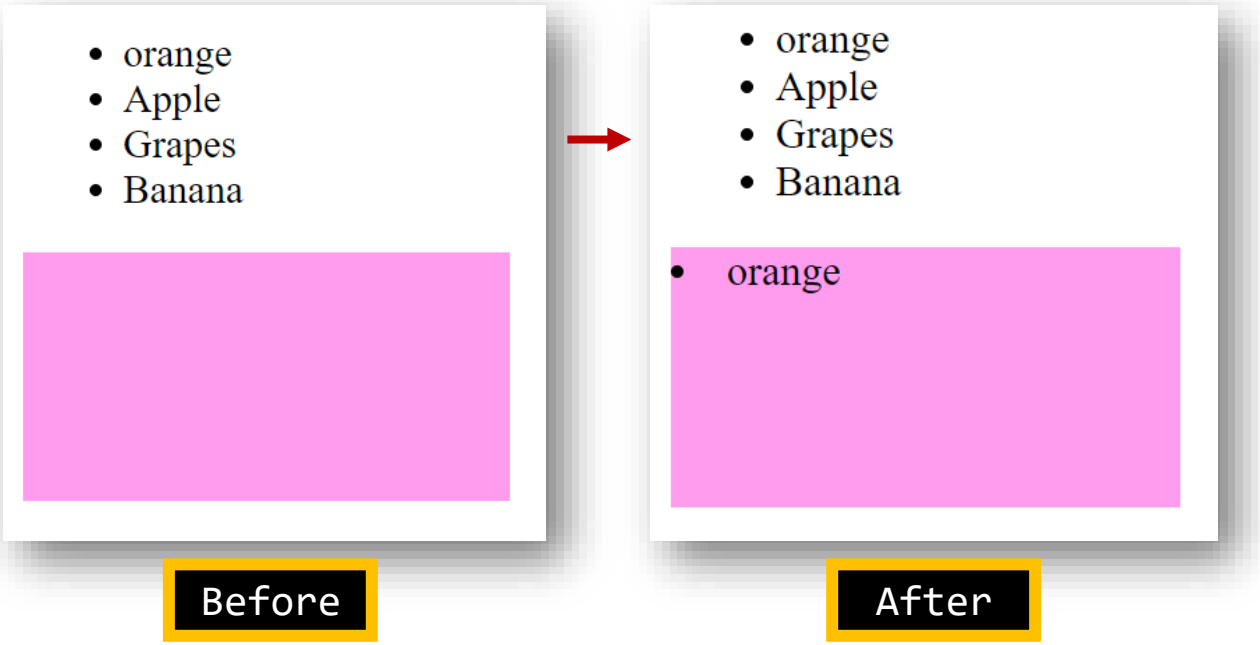
True –
- Returns whole tag with inner Text

```html
<ul id="list1">
    <li class="firstItem">orange</li>
    <li>Apple</li>
    <li>Grapes</li>
    <li>Banana</li>
</ul>
<ul id="list2">
    <li>Carrot</li>
    <li>Reddish</li>
</ul>
<script>
var target = document.getElementById("list1").children[0];
var copyElement = target.cloneNode(true);
document.getElementById("list2").appendChild(copyElement);
</script>
```

e.g. - 1


```html
<ul id="list1">
    <li class="firstItem">orange</li>
    <li>Apple</li>
    <li>Grapes</li>
    <li>Banana</li>
</ul>
<div id="test"></div>
<script>
var target = document.getElementById("list1").children[0];
var copyElement = target.cloneNode(true);
document.getElementById("test").appendChild(copyElement);
</script>
```

e.g. - 2



DOM – contains()

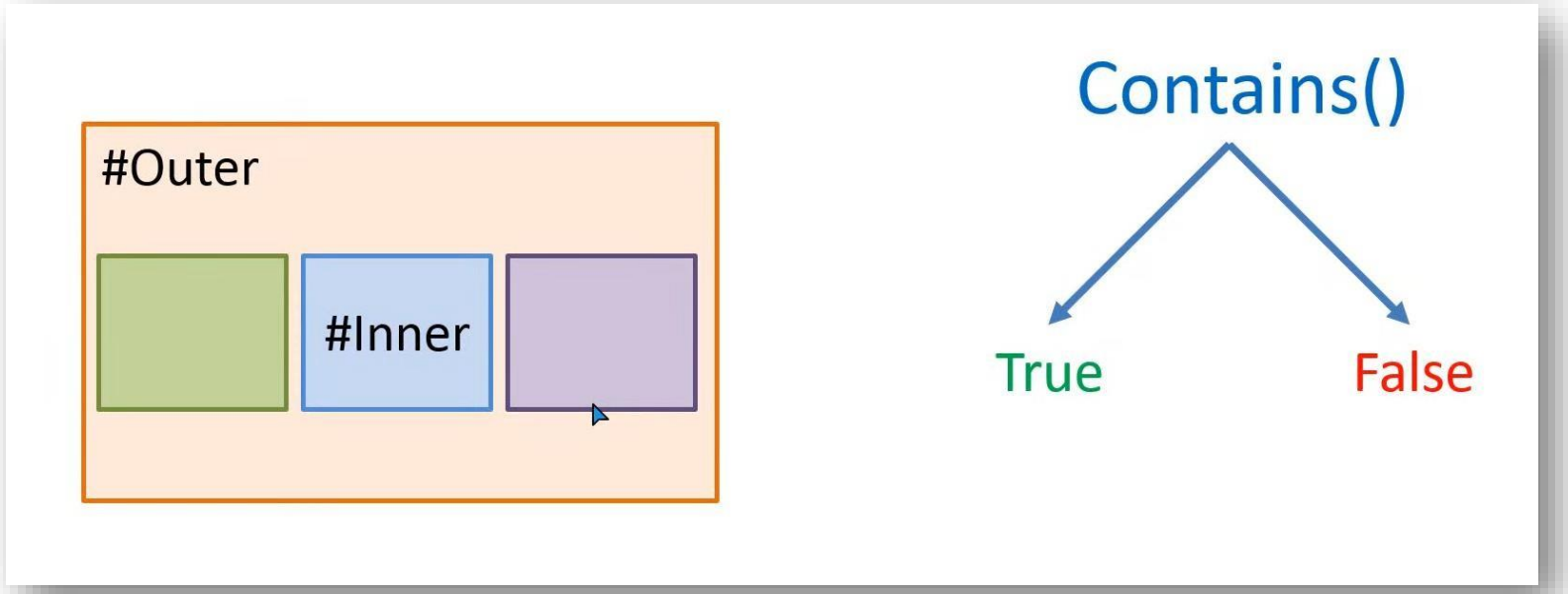
- cloneNode : return only true Or. false

✓ Syntax :

```
parentNode.contains(tagetedTag)
```

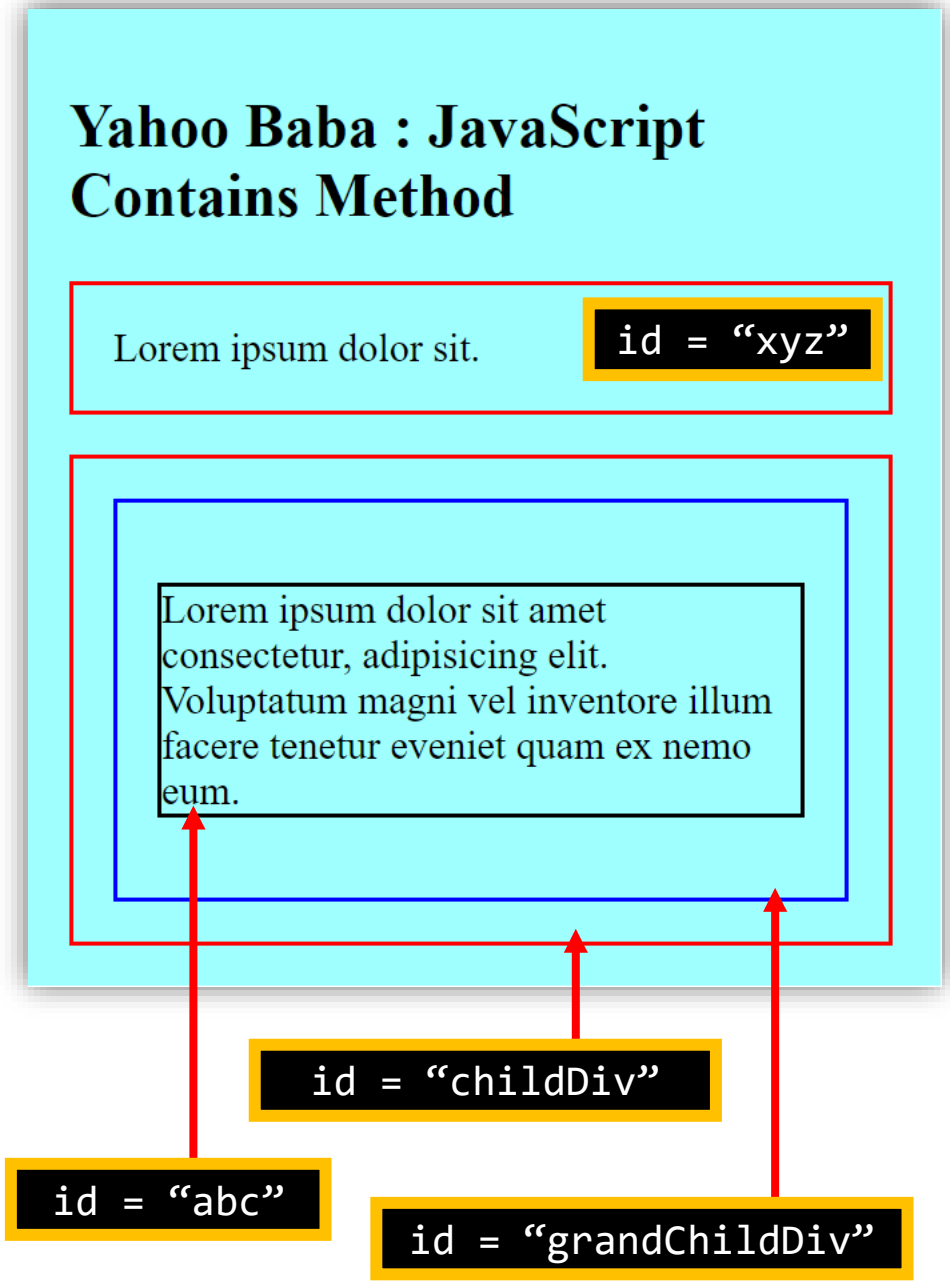
✓ Use :

- Check this parent node consist this tag or not ?



```
<div id="test">
    <h2>Yahoo Baba : JavaScript Contains Method</h2>
    <p id="xyz">
        Lorem ipsum dolor sit.
    </p>
    <div id="childDiv">
        <div id="grandChildDiv">
            <p id="abc">
                Lorem ipsum dolor sit amet consectetur,
                adipisicing elit. Voluptatum magni vel
                inventore illum facere tenetur eveniet
                quam ex nemo eum.
            </p>
        </div>
    </div>
</div>
<script>
    var parentElement = document.getElementById("test");
    var target1 = document.getElementById("pqr");
    var target2 = document.getElementById("xyz");
    var target3 = document.getElementById("abc");
    parentElement.contains(target1);
    parentElement.contains(target2);
    parentElement.contains(target3);
</script>
```

```
false
true
true
```



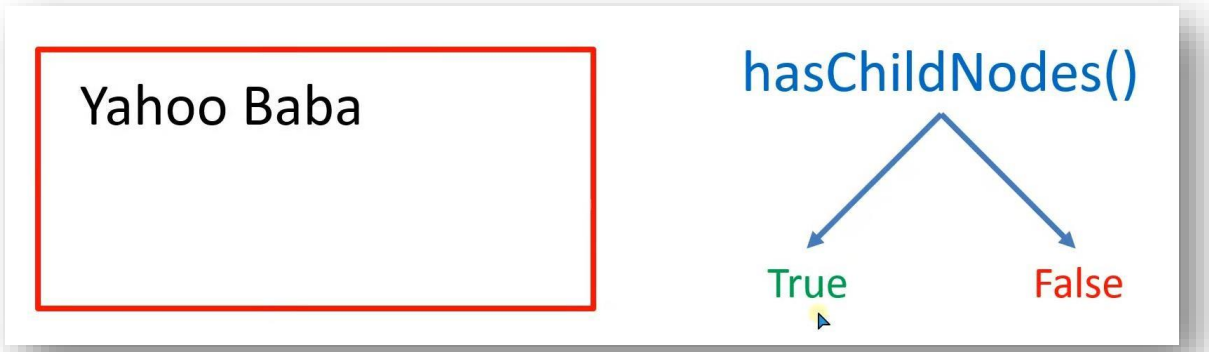
DOM – hasChildNodes() & hasAttributes()

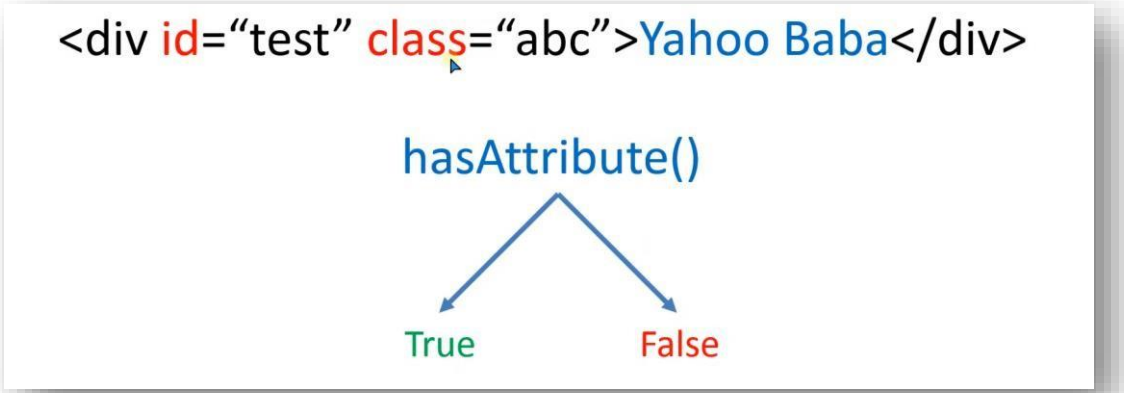
- hasChildNodes() : return only true Or. false
- ✓ Syntax :

  parentNode.hasChildNodes();
- ✓ Use :
- Check is parent node consist text Or. Any type of HTML Tag ? If yes then return true ,Otherwise return false.

- hasAttributes() : return only true Or. false
- ✓ Syntax :

  parentNode.hasAttributes("attributeName");
- ✓ Use :
- Check is targeted node consist Any type of Attributes ? If yes then return true ,Otherwise return false.





- hasChildNode()

```
<div id="test">
</div>
```

Case-1

true
Because we pressed before “enter key”

```
<div id="test"></div>
```

Case-2

False
Beacause not consist anything.

```
<div id="test">
 This is only just a text.
</div>
```

Case-3

True
Beacause consist Text.

```
<div id="test">
    <p>Lorem ipsum dolor sit amet consectetur
       adipisicing elit. Necessitatibus,
       perspiciatis.
    </p>
</div>
```

Case-4

True
Beacause consist HTML Tag .

```
<script>
    var target = document.getElementById("test");
    var find = target.hasChildNodes();
</script>
```

- hasAttribute()

```
<div id="test" class="abc">This is just Only Text</div>
<script>
    var target = document.getElementById("test");
    var find = target.hasAttribute("class");
</script>
<script>
    var target = document.getElementsByClassName("abc")[0];
    var find = target.hasAttribute("id");
</script>
```

true

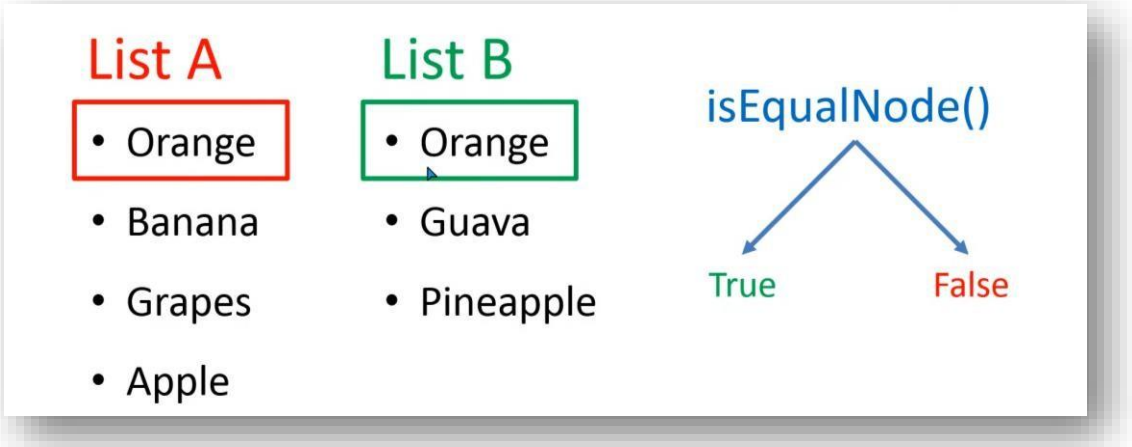
true

```
<button style="border: .1rem soklid red;">this is only just a button</button>
<script>
    var target = document.getElementsByTagName("button")[0];
    var find = target.hasAttribute("style");
</script>
```

true

DOM – isEqualNode()

- Syntax :
- ✓ `target1.isEqualNode(target2);`
- It returns true Or. False
- If two nodes are equal then return true
- Otherwise return false



❑ Based on this parameters we get true Or. False

- Same Types Of Nodes.

```
<li></li>  <⟹>  <li></li>      true
<li></li>  <⟹>  <p></p>        false
```

- Same Text

```
<p>this is just a text</p>  <⟹>  <p>this is just a text</p>      true
<p>this is just a text</p>  <⟹>  <p>this is just a paragraph</p>  false
```

- Same Children Nodes

```
<div>                <⟹>  <div>
    <p></p>                   <p></p>
</div>                    </div>
                                                  true

<div>                <⟹>  <div>
    <p></p>                   <h3></h3>
</div>                    </div>
                                                  false
```

- Same Attributes

```
<div id="" class=""></div>   ⟺   <div id="" class=""></div>    true
<div id="" class=""></div>   ⟺   <div id="" style=""></div>    false
```

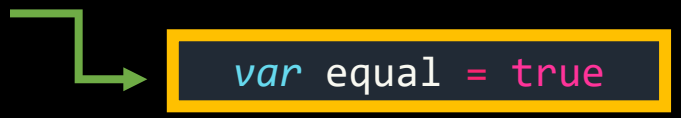
- Same Attributes' Values

```
<div id="main" class="abc"></div>   ⟺   <div id="main" class="abc"></div>      true
<div id="main" class="abc"></div>   ⟺   <div id="general" class="xyz"></div>   false
```

- Note :
- These all rules apply on target node & also on it's child nodes

- Example to see for syntax

```
<ul id="list-1">
    <li class="abc">orange</li>
    <li>Banana</li>
    <li>Apple</li>
    <li>Grapes</li>
</ul>
<ul id="list-2">
    <li>Guava</li>
    <li class="abc">orange</li>
    <li>PineApple</li>
</ul>
<script>
    var target1 = document.getElementById("list-1").firstElementChild;
    var target2 = document.getElementById("list-2").children[1];
    var equal = target1.isEqualNode(target2);
</script>
```



Form Events – part-1 & part-2

- keydown
- keypress
- keyup
- focus()
- blur()
- input
- change
- select
- submit
- invalid

- Note : *onkeypress , onkeydown & onkeyup* – these events are already disucussed.

- onfocus() : when we are focus on any form tag , this event is fired.

- onblur() : when we are click anyway on screen , by this event kill the focus Event.

- oninput() : when we typed on input Or. Textarea tag, by using this event we can parallelly do some task.
- *e.g.* : parallelly write something in another HTML Tag. For Example, in `<div></div>`

- Note : this event works only on input Or. Text area tag

- oninput() : when we select of option Tag in Select Tag, by using this event we can parallelly do some task.
- *e.g.* : we can parallelly print selected option value in another HTML Tag. For Example, in `<div></div>`

➢ when we typed on input Or. Textarea tag, by using this event we can't parallelly do some task. When we do loss focus from input Or. Text area Tag then we can see it's value in another HTML Tag.

- Note : this event works only on **option tag** , input & Text area tag .
  (Majorly used for option tag for doing parallel work)

- onselect() : when we select of any text like following image, then this event is called.

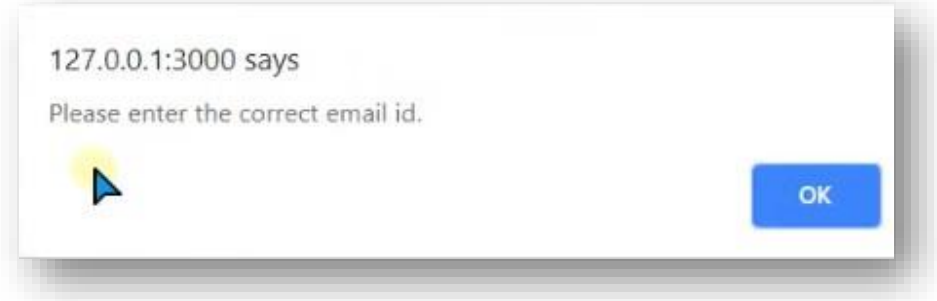
- When we select any text like this, this event is called.
- e.g. by using this event we can call alert() box. Like this ...
  - ✓ alert("You Can't Copy Anything From Here.")

- onsubmit() : when we are submitting form , this is event is called.
- *Note :* This event works only on `<form></form> Tag.`
- *e.g.* `<form onsubmit="funName()"> ... </form>`

- oninvalid() : when we want to do something on invalid activity , we can do by using this event.
- *e.g.* when we write invalid Text (in opposite manner) then , we have to require to display that "That is invalid".

```
<input type="text" id="fname1" oninvalid="alert('Please fill the First Name.')" required>
```


```
<input type="email" id="fname2" oninvalid="alert('Please enter the correct email id.')" required>
```



etInterval() & clearInterval()

```
<script>
    var a = 0;
    var id = setInterval(Anim, 500);

    function Anim() {
        a = a + 10;
        if (a == 200) {
            clearInterval(id);
        } else {
            var target = document.getElementById("test");
            //target.style.marginLeft = a + 'px';
            target.style.width = a + 'px';
        }
    }
</script>
```

```
<style>
    #test{
        width:150px;
        height:150px;
        background: red;
    }
</style>
```

- setInterval() :

- Syntax :
✓ setInterval(functionName, n);
✓ Where : n = time (in milliseconds)

- After every n milliseconds this targeted thing is occurred *Again & Again*.

- clearInterval() :

- Syntax :
- clearInterval();

- By using this we can unset the setInterval().

setTimeoutl() & clearTimeout()

```
<div id="test"></div>
<button onclick="stopAnimation()">Stop Animation</button>
<script>
    var id = setTimeout(Anim, 5000);
    function Anim() {
        var target = document.getElementById("test");
        target.style.width = "500px";
        //console.log("Hello");
    }
    var id = setTimeout(function () {
        var target = document.getElementById("test");
        target.style.width = "500px";
    }, 5000);
    /* JavaScript clearTimeout*/
    function stopAnimation() {
        clearTimeout(id);
    }
</script>
```

```
<style>
    #test{
        width:150px;
        height:150px;
        background: red;
    }
</style>
```

- setInterval() :

- Syntax :
✓ setTimeout(functionName, n);
✓ Where : n = time (in milliseconds)

- After n milliseconds this targeted thing is occurred *Only 1 Time*.

- clearTimeout() :

- Syntax :
- clearTimeout();

- By using this we can unset the setTimeout().

➢ What should I do to Cancel setTimeout() ?

✓ Ans :
We have to require to press button before n milliseconds. So, After n milliseconds targeted thing is not occurred.

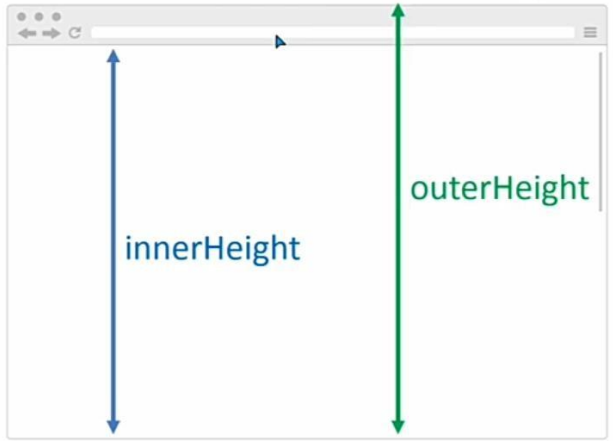
BOM – Introduction (Browser Object Model)
inner width & height , Outer width & height

Browser Object Model

- Get Width & Height of Browser Window
- Open & Close Browser Window
- Move & Resize Browser Window
- Scroll to Browser Window
- Get URL, Hostname, Protocol of Browser Window
- Get History of Browser Window

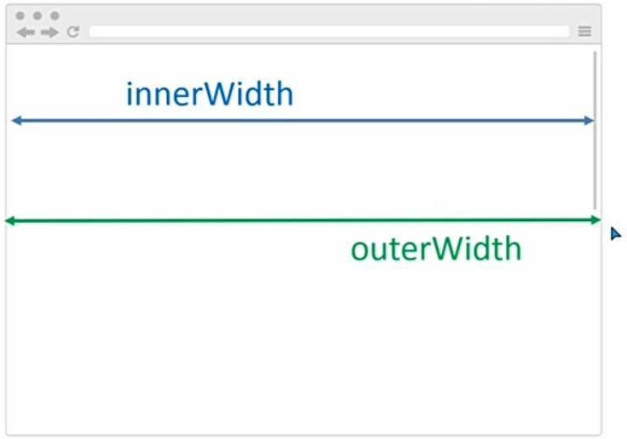
Window Height & Width Methods :

- innerHeight
- innerWidth
- outerHeight
- outerWidth

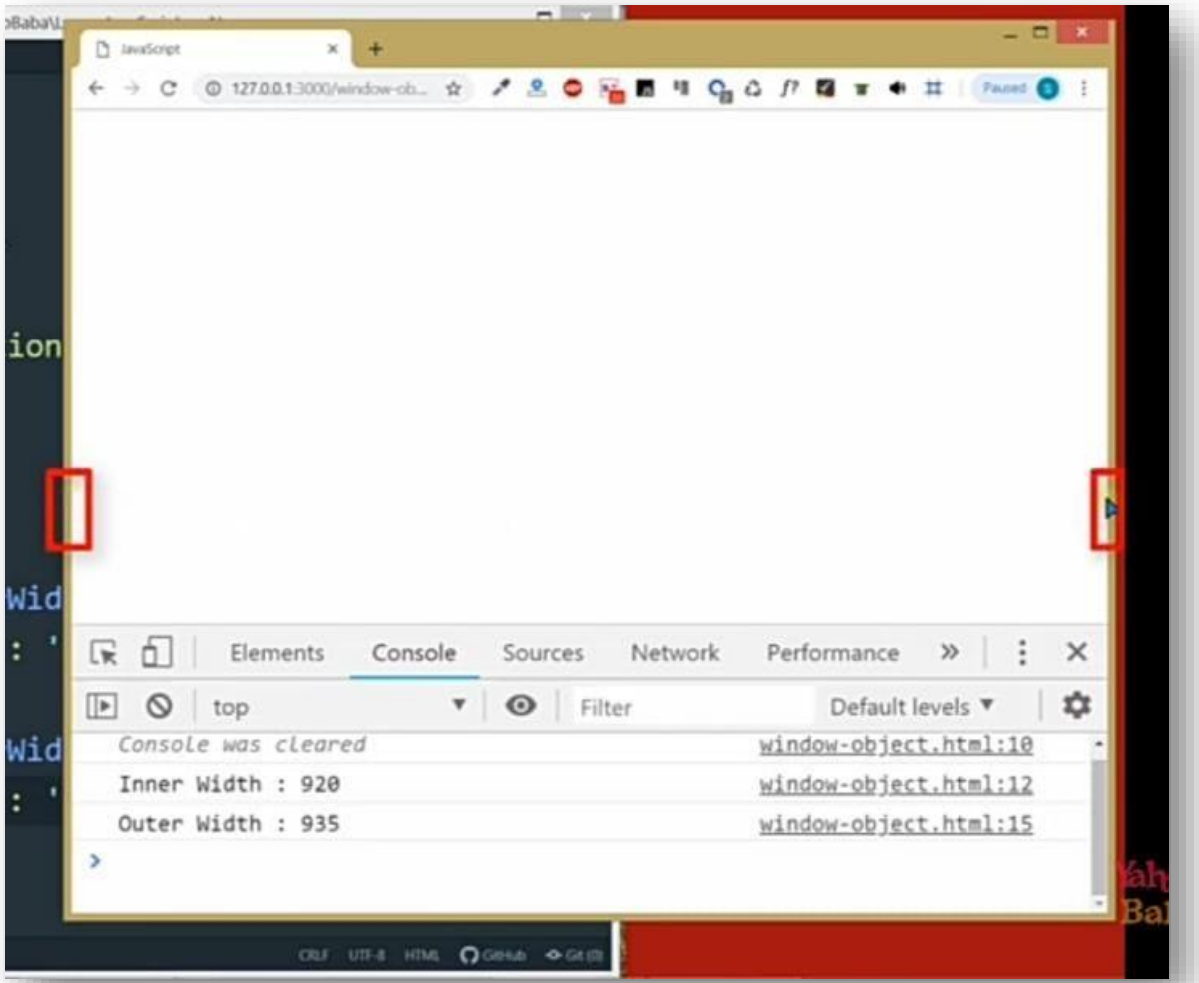


Window Height & Width Methods :

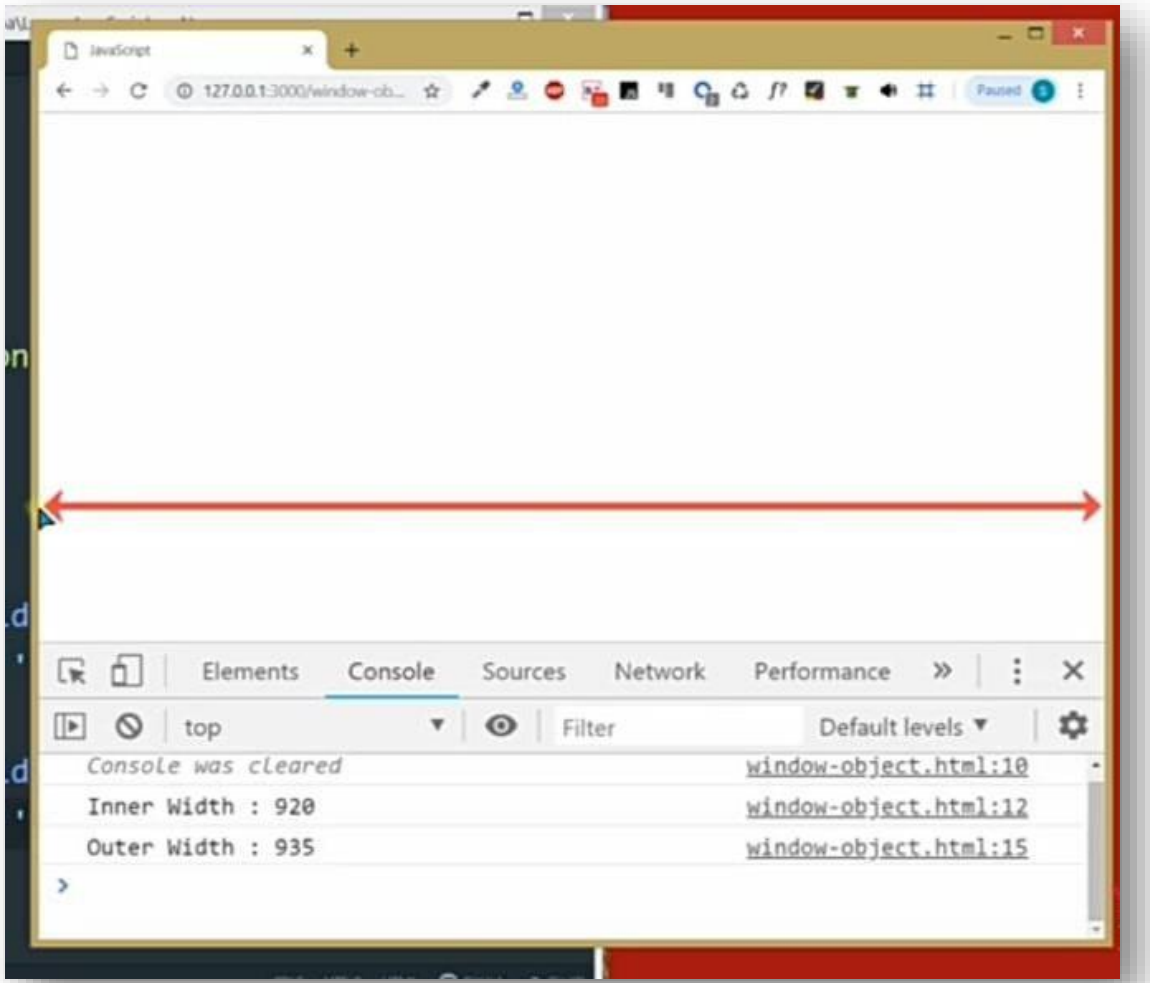
- innerHeight
- innerWidth
- outerHeight
- outerWidth



Outer Width & Inner Width

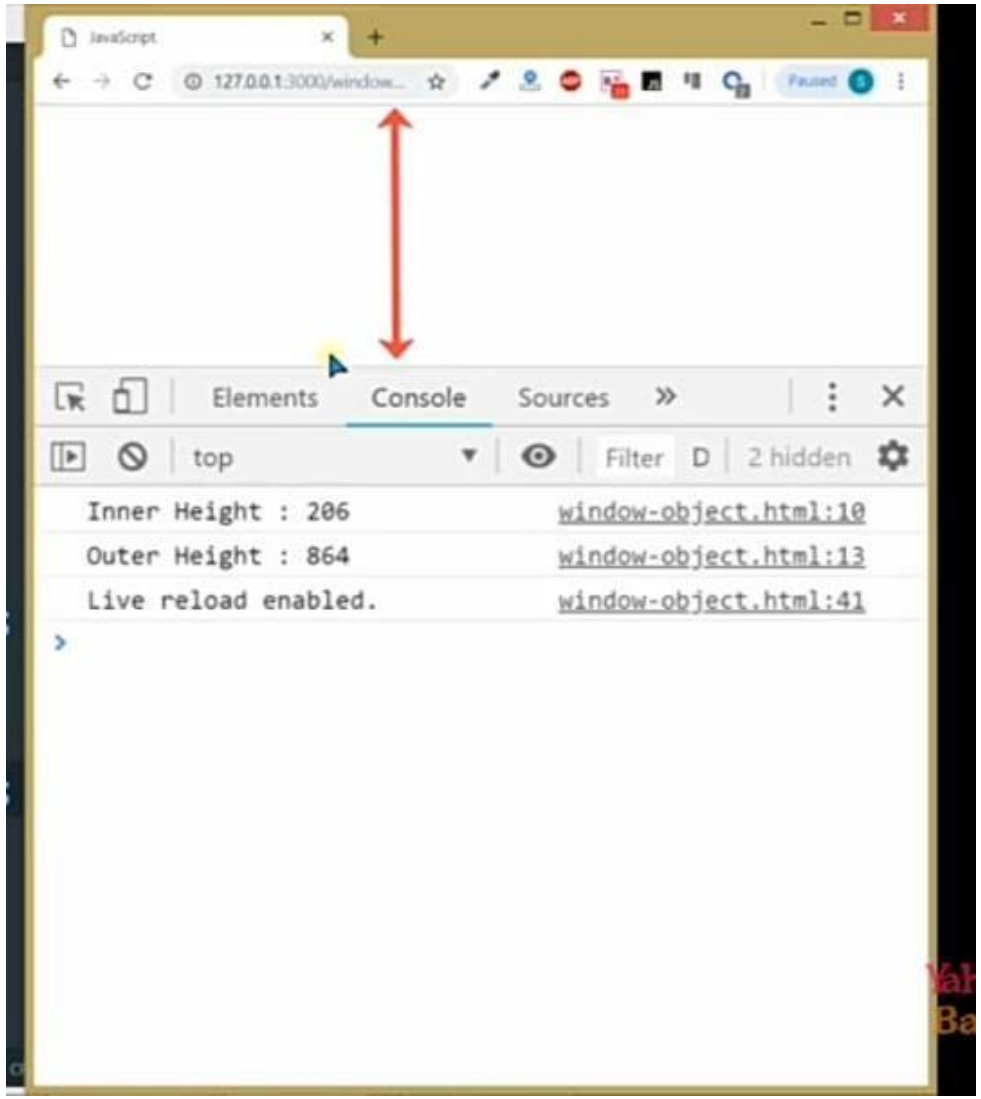


Outer width

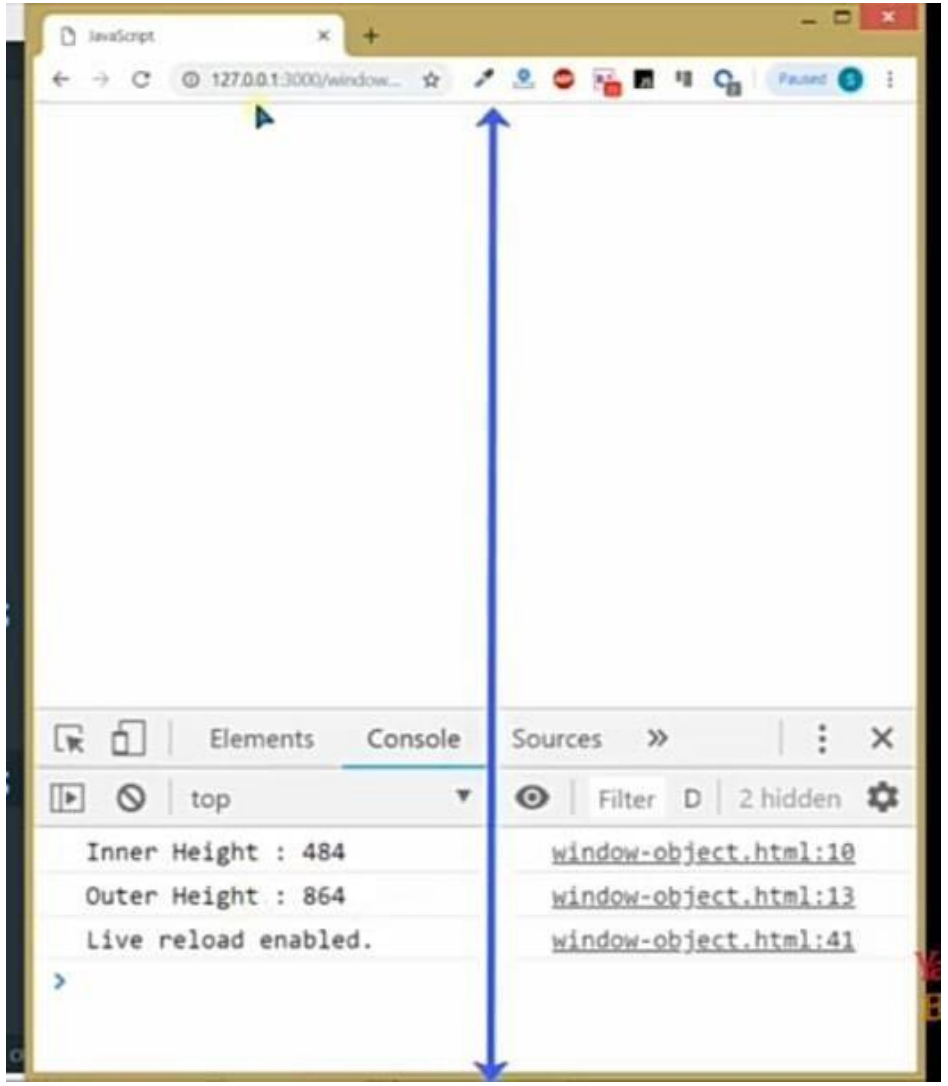


Inner width

Outer Height & Inner Height



Inner Height



Outer Height

Window open & close

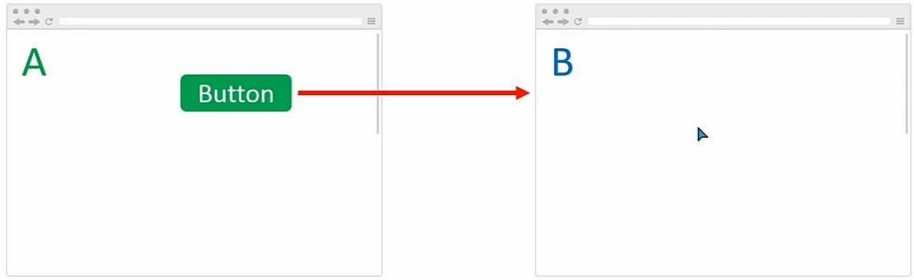
Window Open & Close Methods :

window.open(URL, name, specs)

- URL: http://www.yahoobaba.net
- name: firstWindow or
  - _blank
  - _parent
  - _self
  - _top
- specs:
  - Width
  - Height
  - Left
  - Top

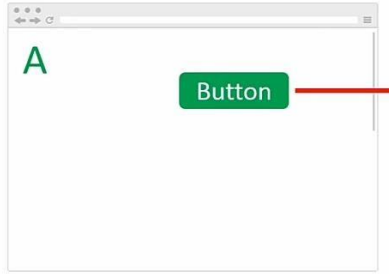
```
URL (by default) = current URL
name (by default) = "_blank";
Specifications (by default):
    width = device width;
    height = device heigth;
    left = 0px;
    top = 0px;
```

Window Open & Close Methods :



Open() Method

Window Open & Close Methods :



Close() Method

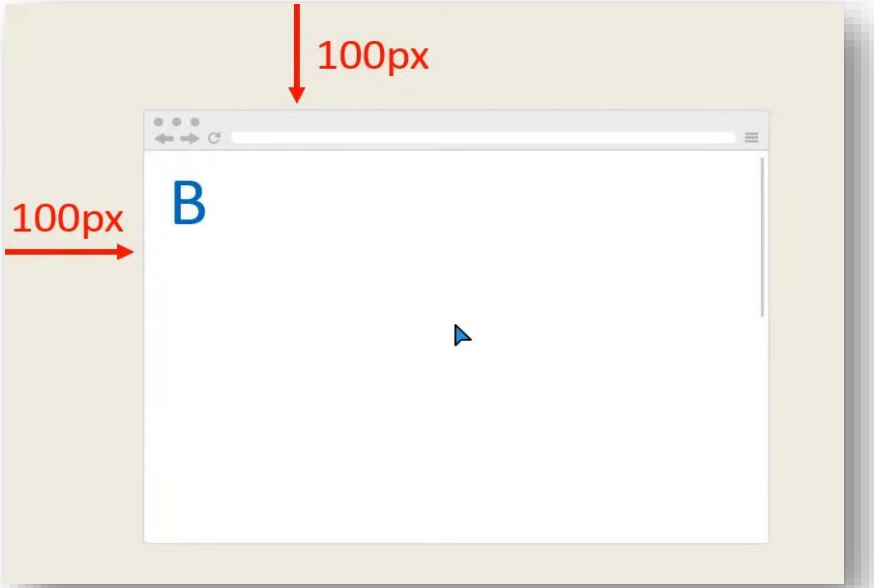
```
<button onclick="openWindow()">Open Window</button>
<button onclick="closeWindow()">Close Window</button>

<script>
    /* JavaScript Open Close Window Method */
    var myWindow;
    function openWindow() {
        //window.open("http://www.yahoobaba.net","yahoobaba");
        //window.open("http://www.yahoobaba.net","_blank");
        //window.open("http://www.yahoobaba.net","_parent");
        myWindow = window.open("http://www.yahoobaba.net", "", "width=500px,height=200px,left=100px,top=200px");
    }

    function closeWindow() {
        myWindow = window.close();
    }
</script>
```

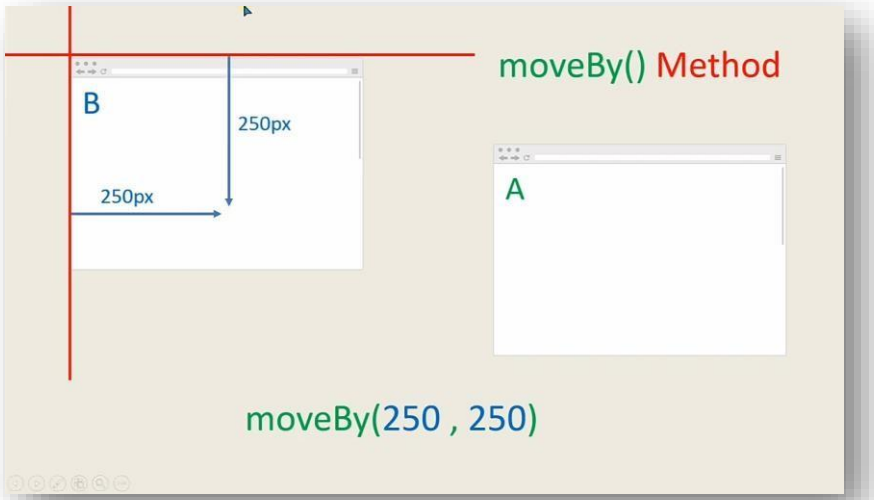
Window moveTo() & moveBy()

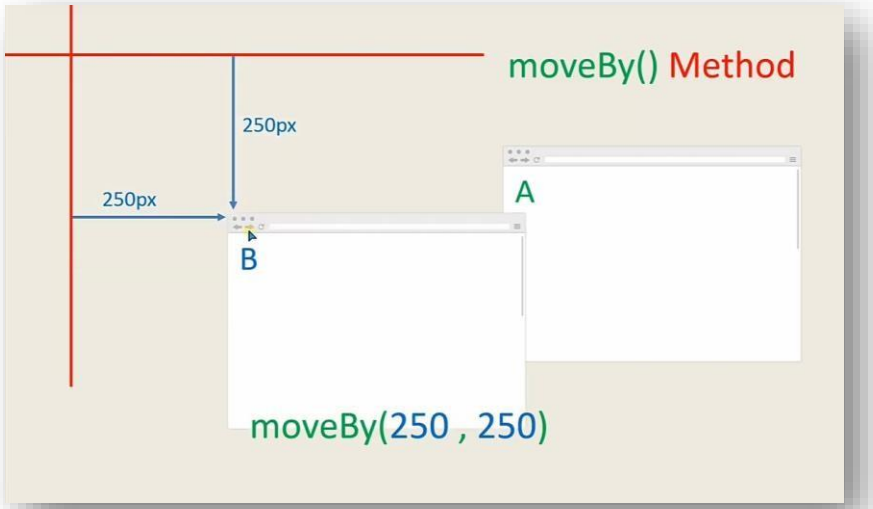
1. moveTo(new width , new height)

1. moveTo(new width , new height) –
   - ✓ Applied width : (new width) – (already given width)
   - ✓ Applied height : (new height) – (already given height)

2. moveBy(new width , new height) –
   - ✓ Applied width : (already given width) + (new width)
   - ✓ Applied height : (already given height) + (new height)

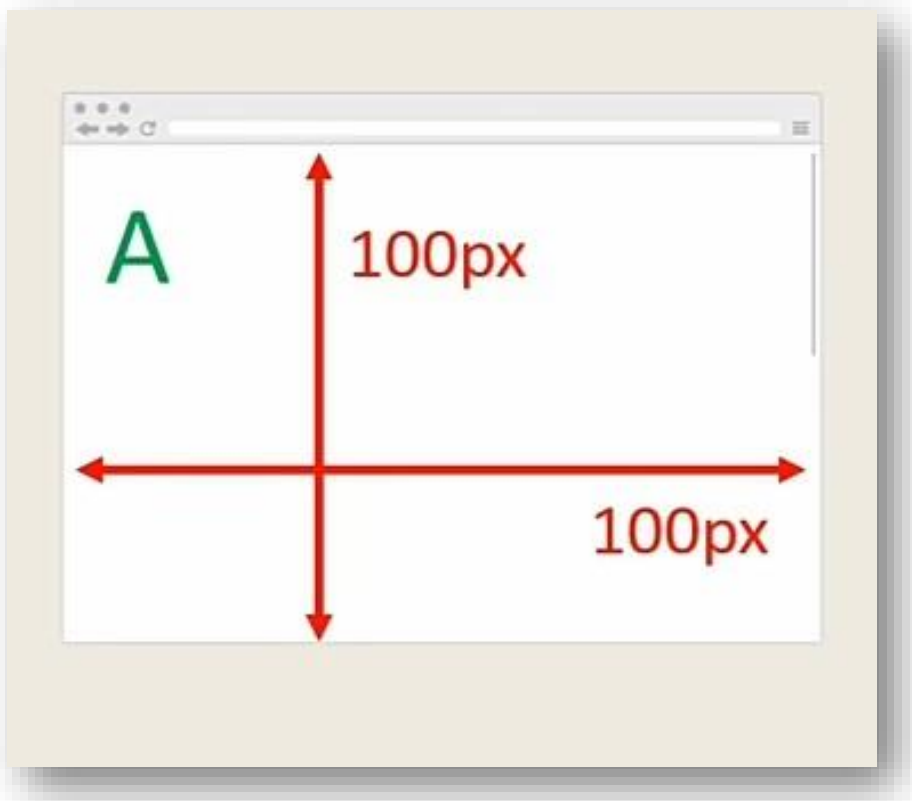


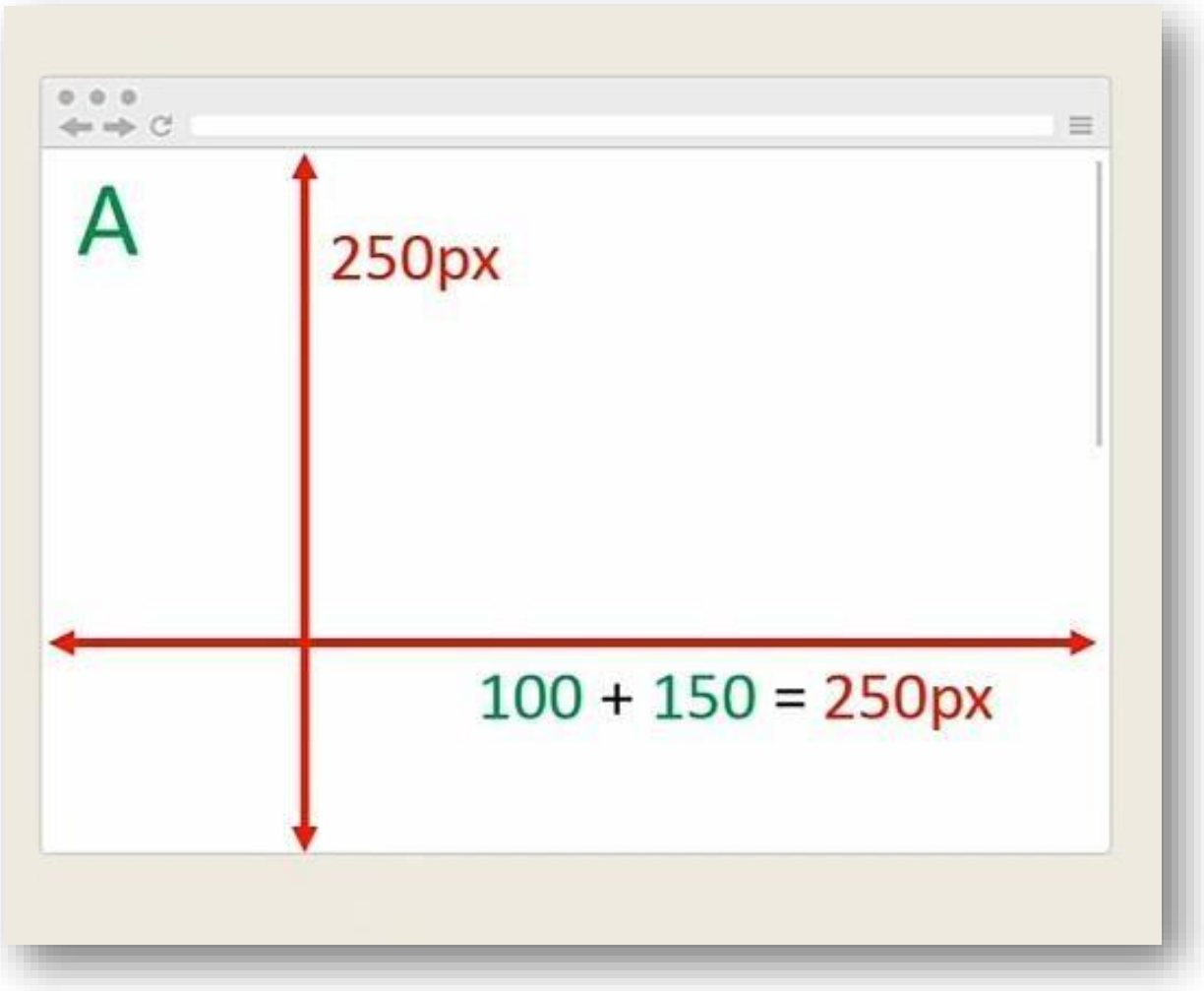




2. moveBy(new width , new height)

Window resizeTo() & resizeBy()






```
resizeTo(new width , new height)
```

```
resizeBy(new width , new height)
```

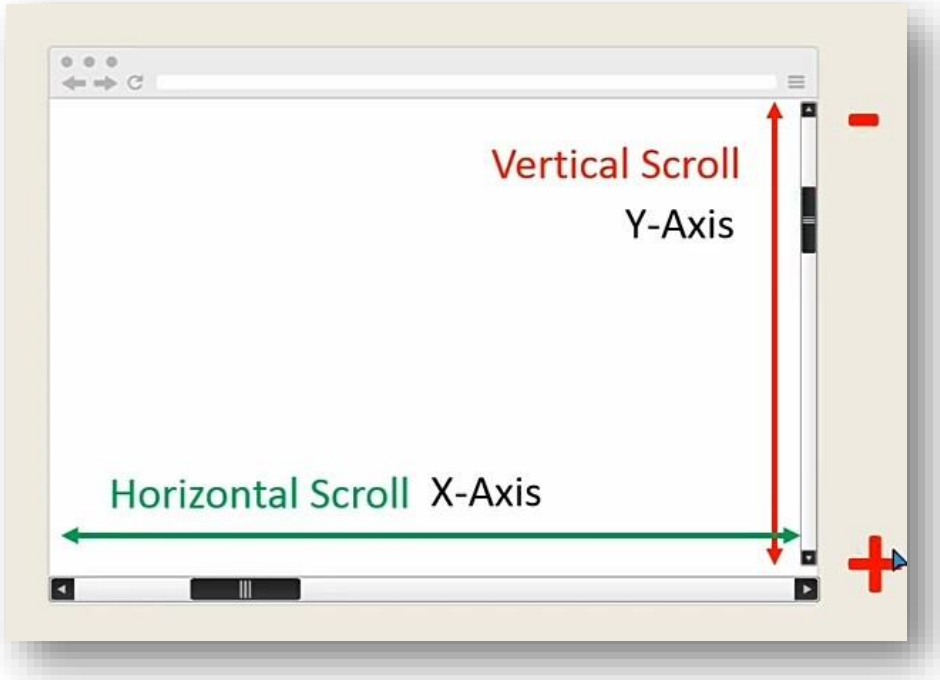
1. resizeTo(new width , new height) –
   - ✓ Applied width : (new width) – (already given width)
   - ✓ Applied height : (new height) – (already given height)

2. resizeBy(new width , new height) –
   - ✓ Applied width : (already given width) + (new width)
   - ✓ Applied height : (already given height) + (new height)

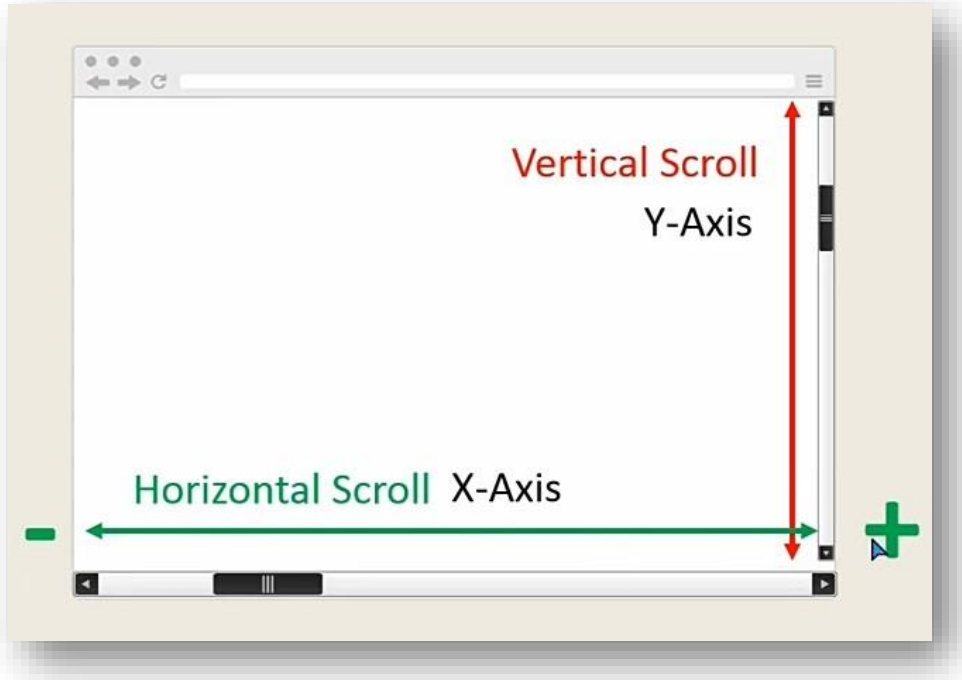
Window scrollTo() & scrollBy()

scrollTo() & scrollBy()

Direction for vertical scrollBar

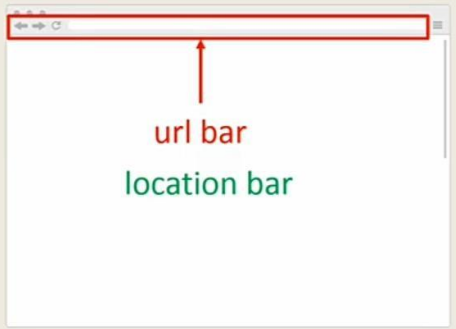


Direction for horizontal scrollBar



Location object & 3 – methods

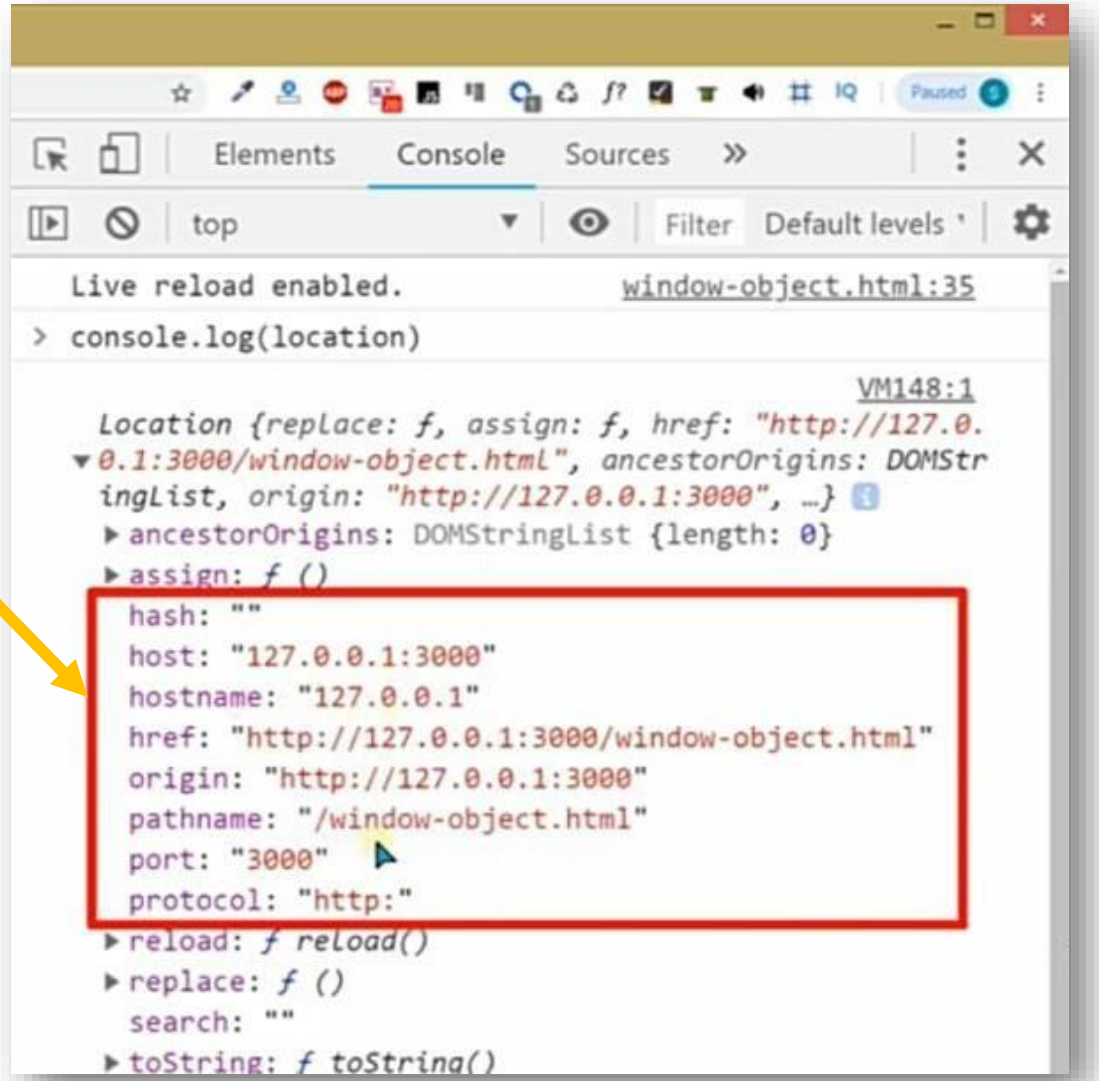
- hash
- host
- hostname
- href
- origin
- pathname
- port
- protocol
- search

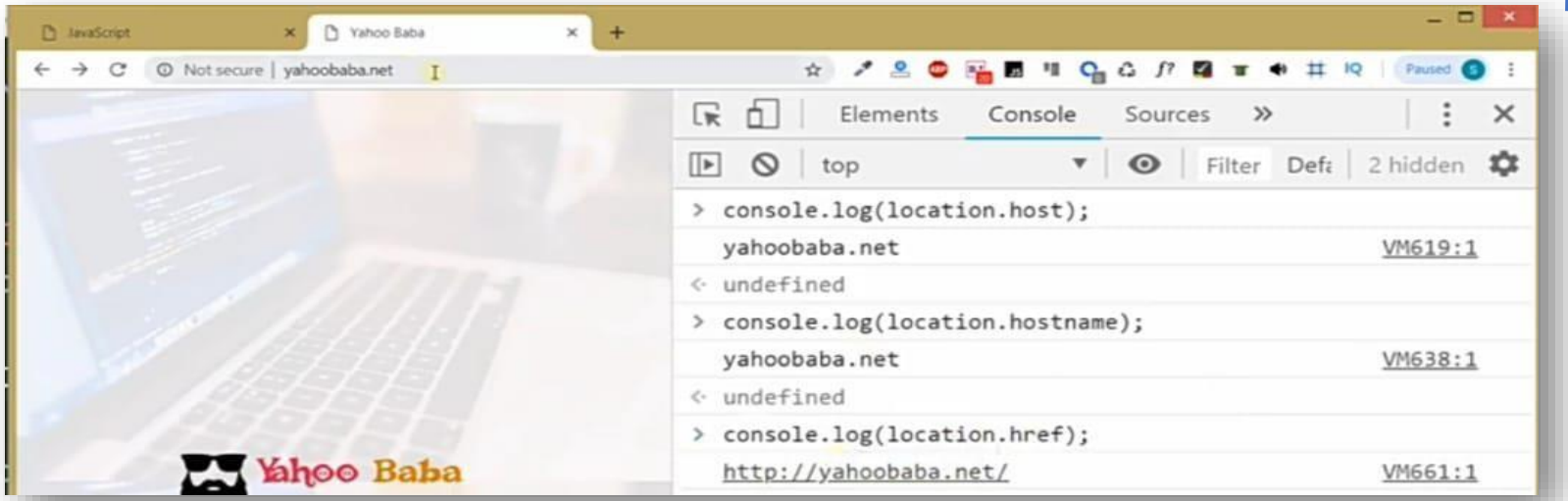


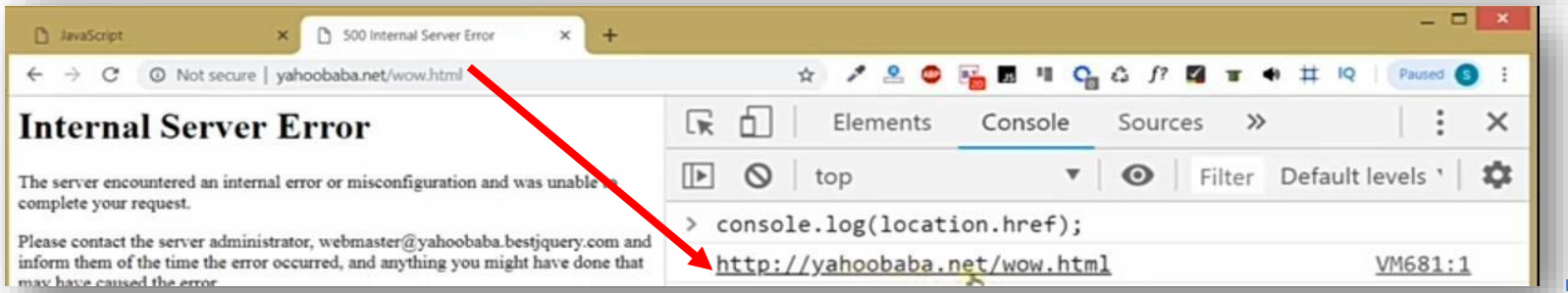
Location Object' Properties

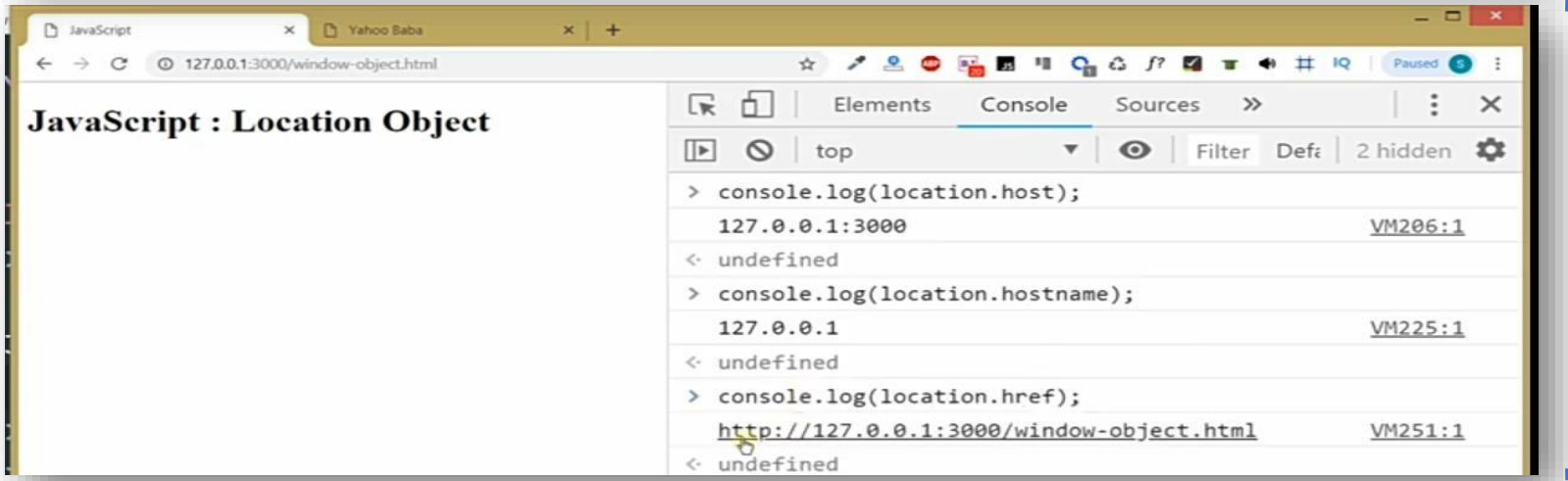
- Assign()
- Reload()
- Replace()

Location Object' Methods



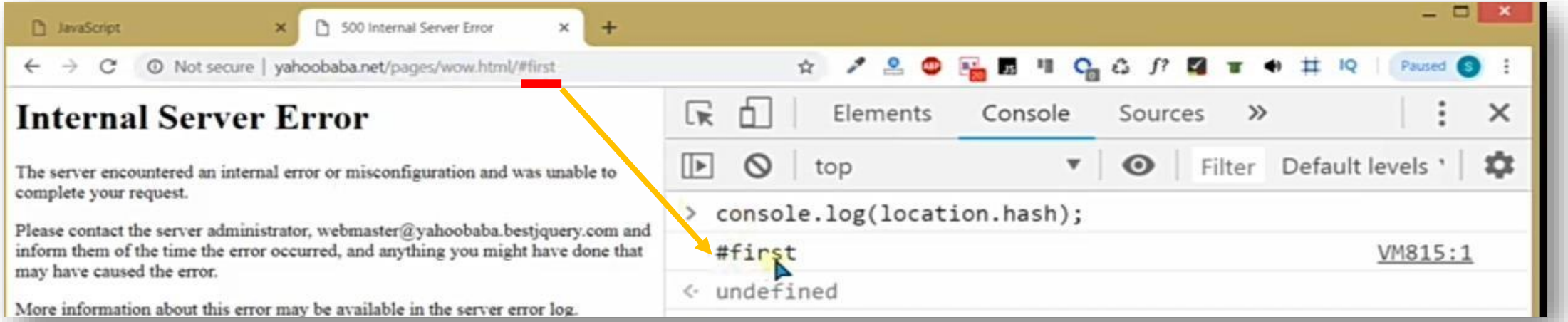


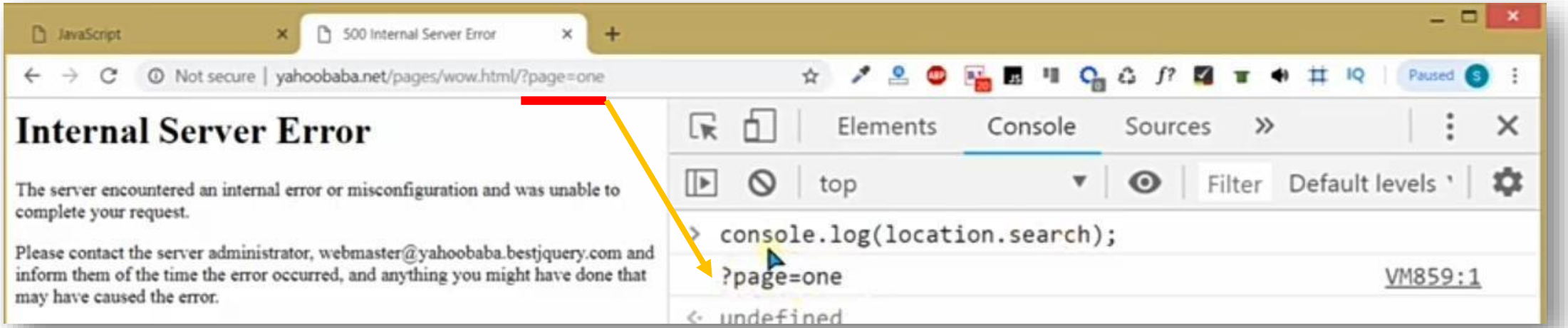


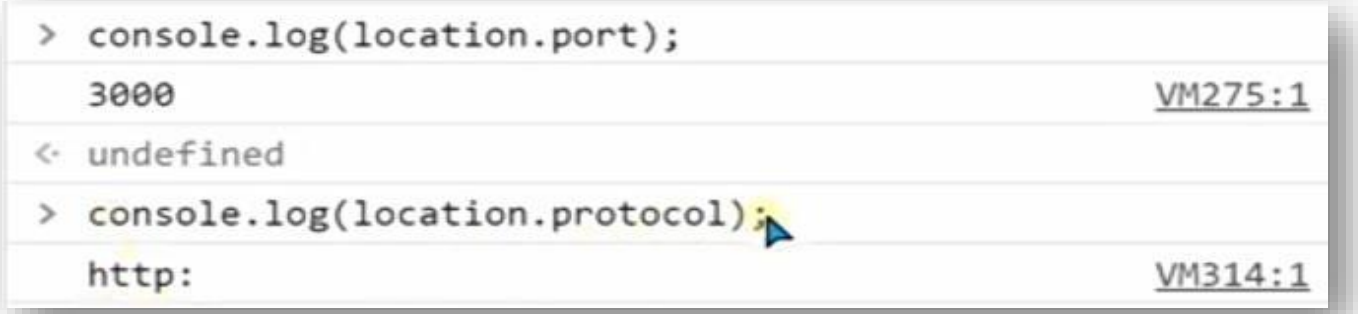


On website

On localhost


JavaScript
500 Internal Server Error
Not secure | yahoobaba.net/pages/wow.html/#first
Paused
Internal Server Error
The server encountered an internal error or misconfiguration and was unable to complete your request.
Please contact the server administrator, webmaster@yahoobaba.bestjquery.com and inform them of the time the error occurred, and anything you might have done that may have caused the error.
More information about this error may be available in the server error log.
Elements
Console
Sources
top
Filter
Default levels
> console.log(location.hash);
#first
VM815:1
undefined


JavaScript
500 Internal Server Error
Not secure | yahoobaba.net/pages/wow.html/?page=one
Paused
Internal Server Error
The server encountered an internal error or misconfiguration and was unable to complete your request.
Please contact the server administrator, webmaster@yahoobaba.bestjquery.com and inform them of the time the error occurred, and anything you might have done that may have caused the error.
Elements
Console
Sources
top
Filter
Default levels
> console.log(location.search);
?page=one
VM859:1
undefined


> console.log(location.port);
3000
VM275:1
undefined
> console.log(location.protocol);
http:
VM314:1

Location Object’ Methods

```
<button onclick="newFunction()">Click</button>
<script>

    /* JavaScript Reload Function*/
    function newFunction() {
        location.reload(); //By using this method we can reload the page
    }

    /* JavaScript Assign Function*/
    function newFunction() {
        location.assign("https://www.google.com"); //By Using this we can redirect on the another URL
    }

    /* JavaScript Replace Function*/
    function newFunction() {
        location.replace("https://www.google.com");
    }
    //By Using this we can redirect on the another URL. But we can’t reach previous page.

</script>
```

- We can also SET this *.href property.*
- Work as same as *assign*() method.

```
<script>
    location.href = "http://www.yahoobaba.net";

    function newFunction() {
        location.href = "http://www.yahoobaba.net";
    }
</script>
```

History object

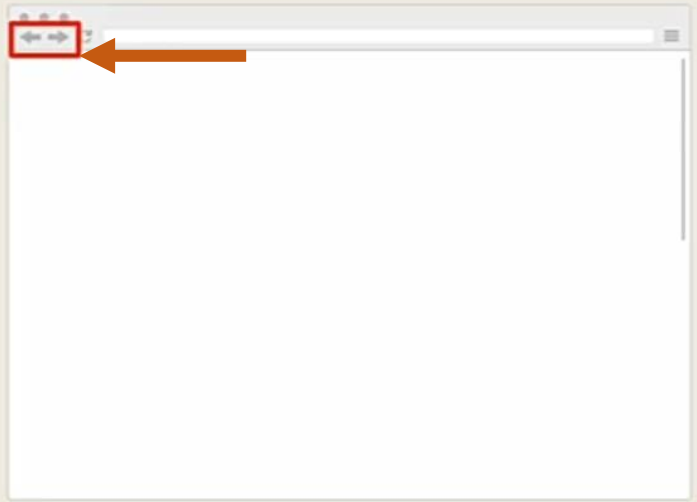
History Object' property

- length

- It returns length.
- Length = How many total pages are in foreward/backward direction ? (from current page)

Hisory Object' Methods

- back()
- forward()
- go()



```html
<button onclick="backFunction()">Back</button>
<button onclick="forwardFunction()">Forword</button>

<button onclick="goFunction()">Go Button</button>
<script>
    /* JavaScript Back Function*/
    function backFunction() { history.back(); }

    /* JavaScript Forward Function*/
    function forwardFunction() { history.forward(); }

    /* JavaScript Go Function*/
    function goFunction() { history.go(1); }
</script>
```

#important thing about go()

```
/* JavaScript Go Function*/
    function goFunction() {
        history.go(n);
    }
```

- Where, n = how many pages , you want to jump in forward/backward direction.
- If(n is -ve) ➔ jumped in backward direction
- If(n is +ve) ➔ jumped in forward direction


Thank You!