

jQuery Cheat Sheet



Methods

Object
 toToString
 toLocaleString
 valueOf
 hasOwnProperty
 isPrototypeOf
 propertyIsEnumerable

String
 charAt
 charCodeAt
 fromCharCode
 concat
 indexOf
 lastIndexOf
 localeCompare
 match
 replace
 search
 slice
 split
 substring
 substr
 toLowerCase
 toUpperCase
 toLocaleLowerCase
 toLocaleUpperCase

RegEx
 test
 match
 exec

Array
 concat
 join
 push
 pop
 reverse
 shift
 slice
 sort
 splice
 unshift

Number
 toFixed
 toExponential
 toPrecision

Date
 parse
 toDateString
 toTimeString
 getDate
 getDay
 getFullYear
 getHours
 getMilliseconds
 getMinutes
 getMonth
 getSeconds
 getTime
 getTimezoneOffset
 getYear
 setDate
 setHours
 setMilliseconds
 setMinutes
 setMonth
 setSeconds
 setYear
 toLocaleTimeString

JavaScript

XMLHttpRequest

Safari, Mozilla, Opera:
`var req = new XMLHttpRequest();`
Internet Explorer:
`var req = new ActiveXObject("Microsoft.XMLHTTP");`

XMLHttpRequest Object Methods

abort()
 getAllResponseHeaders()
 getResponseHeader(header)
 open(method, URL)
 send(body)
 setRequestHeader(header, value)

XMLHttpRequest Object Properties

onreadystatechange
 readyState
 responseText
 responseXML
 status
 statusText

XMLHttpRequest readyState Values

0 Uninitiated
 1 Loading
 2 Loaded
 3 Interactive
 4 Complete

JAVASCRIPT IN HTML

External JavaScript File
`<script type="text/javascript"
 src="javascript.js"></script>`
Inline JavaScript
`<!--
 // JavaScript Here
 -->`
`</script>`

Functions

Window	Built In
alert	eval
blur	parseInt
clearTimeout	parseFloat
close	isNaN
focus	isFinite
open	decodeURI
print	decodeURIComponent
setTimeout	encodeURI
	encodeURIComponent
	escape
	unescape

REGULAR EXPRESSIONS - FORMAT

Regular expressions in JavaScript take the form:
`var RegEx = /pattern/modifiers;`

REGULAR EXPRESSIONS - MODIFIERS

/g Global matching
 /i Case insensitive
 /s Single line mode
 /m Multi line mode

REGULAR EXPRESSIONS - PATTERNS

^ Start of string
 \$ End of string
 . Any single character
 (a|b) a or b
 (...) Group section
 [abc] Item in range (a or b or c)
 [^abc] Not in range (not a or b or c)
 a? Zero or one of a
 a* Zero or more of a
 a+ One or more of a
 a{3} Exactly 3 of a
 a{3,} 3 or more of a
 a{3,6} Between 3 and 6 of a
 !(pattern) "Not" prefix. Apply rule when URL does not match pattern.

EVENT HANDLERS

onAbort	onMouseDown
onBlur	onMouseMove
onChange	onMouseOut
onClick	onMouseOver
onDoubleClick	onMouseUp
onDragDrop	onMove
onError	onReset
onFocus	onResize
onKeyDown	onSelect
onKeyPress	onSubmit
onKeyUp	onUnload
onLoad	

FUNCTIONS AND METHODS

A method is a type of function, associated with an object. A normal function is not associated with an object.

Available free from
www.ILoveJackDaniels.com

DOM Methods

Document
 clear
 createDocument
 createDocumentFragment
 createElement
 createEvent
 createEventObject
 createRange
 createTextNode
 getElementsByTagName
 getElementById
 write

Node
 addEventListener
 appendChild
 attachEvent
 cloneNode
 createTextRange
 detachEvent
 dispatchEvent
 fireEvent
 getAttributeNS
 getAttributeNode
 hasChildNodes
 hasAttribute
 hasAttributes
 insertBefore
 removeChild
 removeEventListener
 replaceChild
 scrollIntoView

Form
 submit

DOM Collections
 item

Range
 collapse
 createContextualFragment
 moveEnd
 moveStart
 parentElement
 select
 setStartBefore

Style
 getPropertyValue
 setProperty

Event
 initEvent
 preventDefault
 stopPropagation

XMLSerializer
 serializeToString

XMLHTTP
 open
 send

XMLElement
 loadXML

DOMParser
 parseFromString

Regular Expressions Syntax		Pattern Modifiers (cont)		JavaScript Arrays	
^	Start of string	x *	Allow comments and whitespace in pattern	concat()	slice()
\$	End of string	e *	Evaluate replacement	join()	sort()
.	Any single character	U *	Ungreedy pattern	length	splice()
(a b)	a or b	* PCRE modifier		pop()	toSource()
(...)	Group section			push()	toString()
[abc]	In range (a, b or c)			reverse()	unshift()
[^abc]	Not in range			shift()	valueOf()
\s	White space				
a?	Zero or one of a				
a*	Zero or more of a				
a*?	Zero or more, ungreedy				
a+	One or more of a				
a+?	One or more, ungreedy				
a{3}	Exactly 3 of a				
a{3,}	3 or more of a				
a{,6}	Up to 6 of a				
a{3,6}	3 to 6 of a				
a{3,6}?	3 to 6 of a, ungreedy				
\	Escape character				
[:punct:]	Any punctuation symbol				
[:space:]	Any space character				
[:blank:]	Space or tab				
There's an excellent regular expression tester at: http://regexpal.com/					
Pattern Modifiers		JavaScript RegExp Object		JavaScript Numbers and Maths	
g	Global match	compile()	lastParen	abs()	min()
i *	Case-insensitive	exec()	leftContext	acos()	NEGATIVE_INFINITY
m *	Multiple lines	global	multiline	asin()	PI
s *	Treat string as single line	ignoreCase	rightContext	atan()	POSITIVE_INFINITY
		input	source	atan2()	pow()
		lastIndex	test()	ceil()	random()
		lastMatch		cos()	round()
JavaScript Event Handlers		JavaScript Event Handlers		JavaScript Event Handlers	
		onabort	onmousedown	E	sin()
		onblur	onmousemove	exp()	sqrt()
		onchange	onmouseout	floor()	SQRT1_2
		onclick	onmouseover	LN10	SQRT2
		ondblclick	onmouseup	LN2	tan()
		ondragdrop	onmove	log()	toSource()
		onerror	onreset	LOG10E	toExponential()
		onfocus	onresize	LOG2E	toFixed()
		onkeydown	onselect	max()	toPrecision()
		onkeypress	onsubmit	MAX_VALUE	toString()
		onkeyup	onunload	MIN_VALUE	valueOf()
		onload		NaN	

JavaScript Booleans		JavaScript Strings	
toSource()	valueOf()	charAt()	slice()
toString()		charCodeAt()	split() x
JavaScript Dates		concat()	
Date()	setMonth()	fromCharCode()	substr()
getDate()	setFullYear()	indexOf()	toLowerCase()
getDay()	setHours()	lastIndexOf()	toUpperCase()
getMonth	setMinutes()	length	toLocaleLowerCase()
getFullYear	setSeconds()	localeCompare()	toLocaleUpperCase()
getYear	setMilliseconds()	match() x	toSource()
getHours	setTime()	replace() x	valueOf()
getMinutes	setUTCDate()	search() x	
getSeconds	setUTCDay()	String object methods with an x support regular expressions.	
getMilliseconds	setUTCMonth()		
getTime	setUTCFullYear()		
getTimezoneOffset()	setUTCHours()	decodeURI()	isNaN()
getUTCDate()	setUTCMinutes()	decodeURIComponent()	Number()
getUTCDay()	setUTCSeconds()	encodeURI()	parseFloat()
getUTCMonth()	setUTCMilliseconds()	encodeURIComponent()	parseInt()
getUTCFullYear()	toSource()	escape()	String()
getUTCHours()	toString()	eval()	unescape()
getUTCMinutes()	toGMTString()	isFinite()	
getUTCSeconds()	toUTCString()		
getUTCMilliseconds()	toLocaleString()		
parse()	UTC()		
setDate()	valueOf()		

D:\js\jquery\index.html - Notepad++

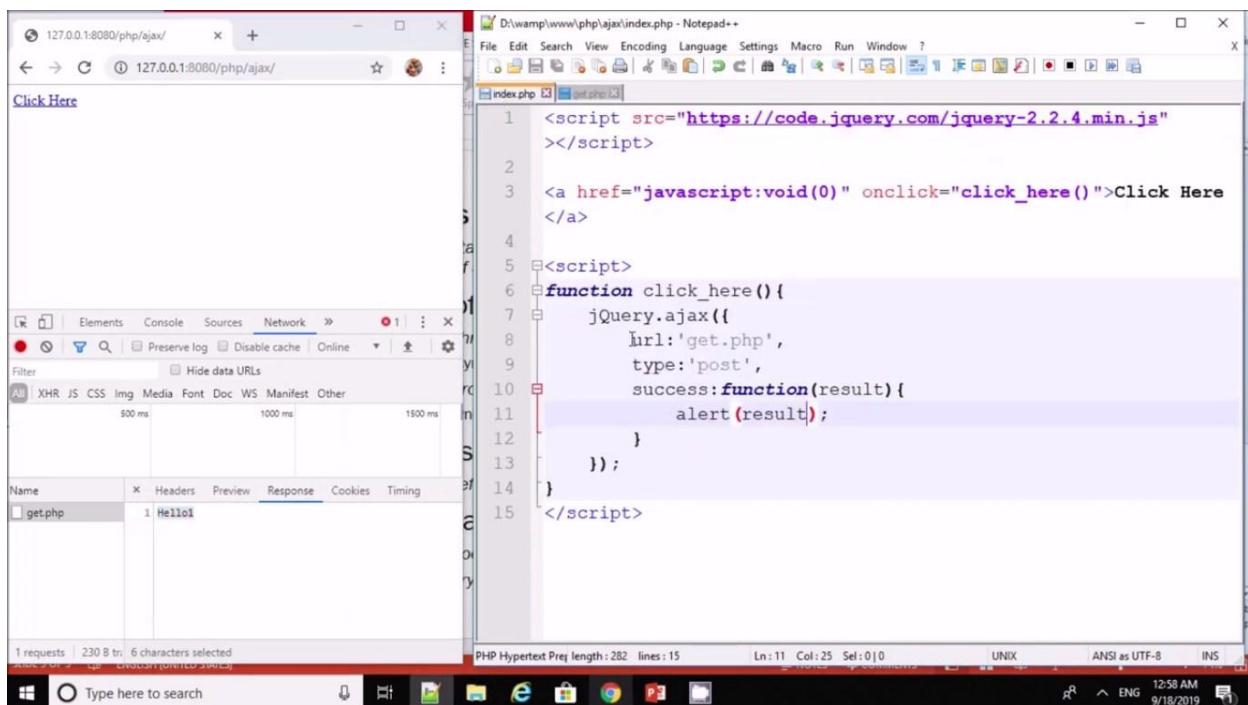
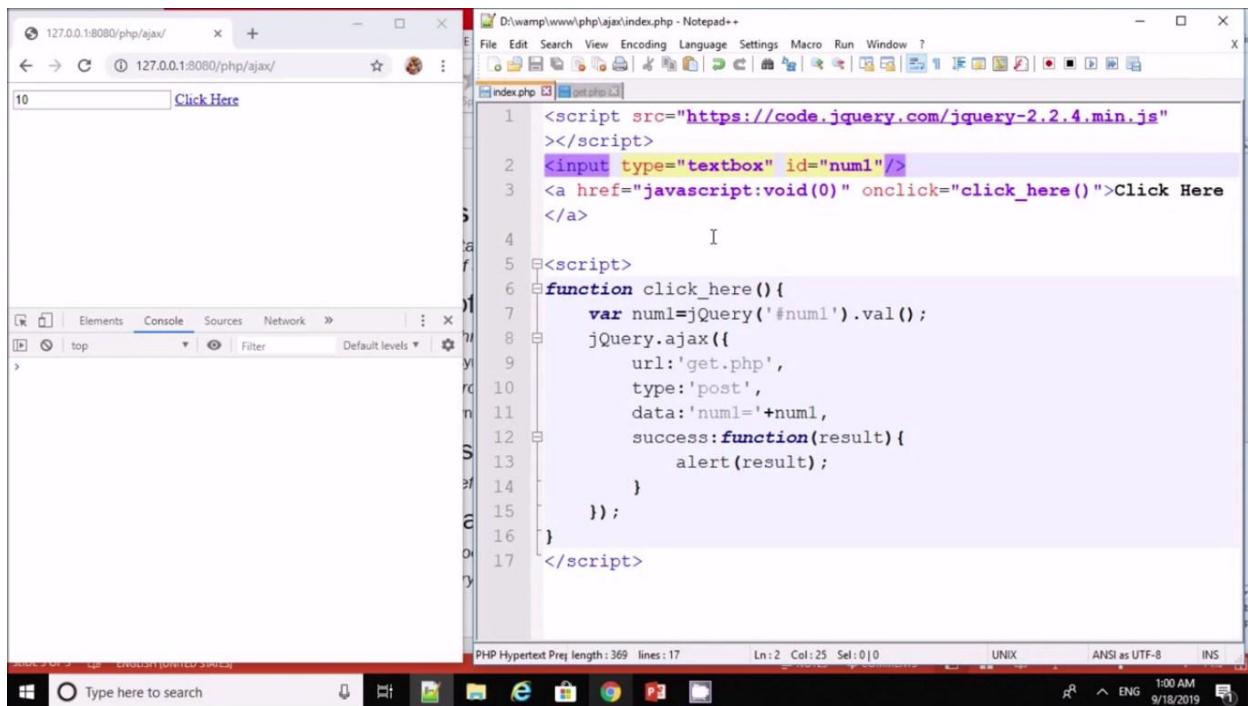
```
</div>
4
5 <script src="js/jquery-3.6.0.min.js"></script>
6 <script>
7 //jQuery('#box').slideUp(5000);
8 //$('#box').slideUp(5000);
9
10 //jQuery('div').slideUp(5000);
11 //jQuery('#box1').slideUp(5000);
12 jQuery('.box').slideUp(5000);
13
14
15 </script>
```

Hyper Text Markup Language file length : 417 lines : 15 Ln : 3 Col : 17 Pos : 111 Windows (CR LF) UTF-8 INS
Type here to search 25°C Light rain ENG

D:\js\jquery\effects.html - Notepad++

```
7
8 <script src="js/jquery-3.6.0.min.js"></script>
9 <script>
10 jQuery('#hide').click(function() {
11     jQuery('#box').hide();
12 });
13 jQuery('#show').click(function() {
14     jQuery('#box').show();
15 });
16 jQuery('#toggle').click(function() {
17     jQuery('#box').toggle();
18 });
19 </script>
```

Hyper Text Markup Language file length : 530 lines : 26 Ln : 17 Col : 26 Sel : 6 | 1 Windows (CR LF) UTF-8 INS
Type here to search 24°C Rain ENG



The screenshot shows a dual-pane Notepad++ interface. The left pane displays a browser window with the URL `127.0.0.1:8080/php/ajax/types_ajax.php`. The right pane contains the source code for `types_ajax.php`:

```
1 <script src="https://code.jquery.com/jquery-2.2.4.min.js">
2 </script>
3 hit1();
4 hit2();
5 function hit1(){
6     jQuery.ajax({
7         url:'get1.php',
8         type:'post',
9         async:false,
10        success:function(result){
11            console.log(result);
12        }
13    });
14 }
15 function hit2(){
16     jQuery.ajax({
17         url:'get2.php',
18         type:'post',
19         success:function(result){
20             console.log(result);
21         }
22    });
23 }
```

Notepad++ status bar details: PHP Hypertext Prej length: 396 lines: 24 Ln: 8 Col: 9 Sel: 0|0 DosWindows ANSI as UTF-8 INS.

The screenshot shows a dual-pane Notepad++ interface. The left pane displays a browser window with the URL `127.0.0.1:8080/html5_validation_withajax/index.html`. The right pane contains the source code for `index.html`:

```
111 <input type="text" class="form-control" name="mobile" placeholder="Mobile*" required pattern="[6-9]{1}[0-9]{9}">
112 </div>
113 <div class="form-group">
114     <input type="password" class="form-control" name="password" placeholder="Password*" required>
115 </div>
116
117 <div class="form-group">
118     <input type="submit" class="btn btn-success btn-lg btn-block" value="Submit Now">
119 </div>
120 </form>
121 </div>
122 <script>
123     jQuery('#contactForm').on('submit',function(e){
124         jQuery.ajax({
125             url:'submit.php',
126             type:'post',
127             data:jQuery('#contactForm').serialize()
128         });
129     });
130 
```

Notepad++ status bar details: Paused Test Markup Language file length: 3515 lines: 133 Ln: 127 Col: 45 Sel: 0|0 DosWindows 1280x720 06:34 / 21:38 ANSI as UTF-8 INS.

The screenshot shows a development environment with two main windows. On the left is a Sublime Text window titled "classes.html" containing the following JavaScript code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Advance JS</title>
    <script>

        class hello{
            message(){
                console.log("Hello Everyone");
            }
            sorry(){
                console.log("Sorry Everyone");
            }
        }

        let a = new hello();

        a.message();
        a.sorry();

    </script>
</head>
<body>
```

On the right is a browser developer tools window titled "Advance JS" showing the "Console" tab. It displays the following log entries:

Message	Source
Hello Everyone	classes.html:9
Sorry Everyone	classes.html:12

JS Class & Object in PHP

```
class hello{  
    message(){  
        console.log("Hello Everyone");  
    }  
}  
  
let a = new hello();  
a.message();
```

JS Type of Methods :

Constructor

```
constructor(){  
    console.log("hello");  
}
```

Prototype

```
message(){  
    console.log("hello")  
}
```

Static

```
static name(){  
    console.log("hello")  
}
```

JS Type of Methods :

Constructor

```
constructor(){
    console.log("hello");
}
```

Prototype

```
message(){
    console.log("hello")
}
```

Static

```
static name(){
    console.log("hello")
}
```

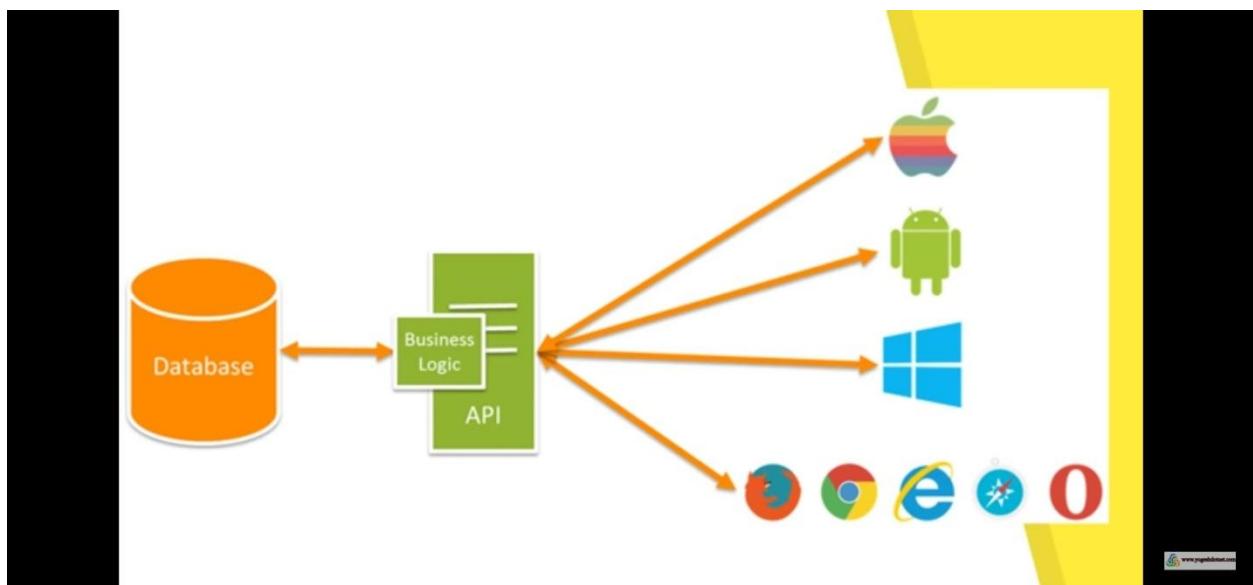
Pillar of OOPS

- ▶ **Data Abstraction**(**Data hide**)
- ▶ **Inheritance.** (**Reusability**)
- ▶ **Polymorphism.** (**Object to take many forms**)
- ▶ **Encapsulation.** (**Data hide**)

REST API

CRUD Operations:-

Operation	HTTP Methods	Description
Create	POST	Creating/Posting/Inserting Data
Read	GET	Reading/Getting/Retrieving Data
Update	PUT, PATCH	Updating Data Complete Update - PUT Partial Update - PATCH
Delete	DELETE	Deleting Data



JSON.stringify()

Transforms a JavaScript object into a JSON string.



JSON.parse()

Takes a JSON string and transforms it into a JavaScript object.



```
> const student = {  
  name:"Alex",  
  age:20,  
  email:"alex@email.com"  
};  
  
> const studentStr = JSON.stringify(student);  
  
> console.log(studentStr);  
  
{"name":"Alex", "age":20, "email":"alex@email.com"}  
  
> const jsObject = JSON.parse(studentStr);  
  
> console.log(jsObject);  
  
Object { age: 20, email: "alex@email.com", name: "Alex" }  
  
> |
```

Data Format of JSON & XML

JSON

```
{"students":[  
  {"name":"Ram", "age":23, "city":"Agra"},  
  {"name":"Sonam", "age":28, "city":"Delhi"},  
]}
```

XML

```
<students>  
  <student>  
    <name>Ram</name> <age>23</age> <city>Agra</city>  
  </student>  
  <student>  
    <name>Sonam</name> <age>28</age> <city>Delhi</city>  
  </student>  
</students>
```

 [Yahoo Baba](http://www.yahooibaba.net) www.yahooibaba.net



Difference Between JSON & XML

JSON

- 1) JavaScript Object Notation
- 2) Text Based Format
- 3) Light Weight
- 4) Does not support comments and namespaces

XML

- 1) Extensible Markup Language
- 2) Markup Language
- 3) Heavier
- 4) Supports comments and namespaces



Javascript Object Literals Vs JSON

JavaScript Object

```
var person = { firstName : "Yahoo", lastName : "Baba" };
```

JSON

```
var person = { "firstName" : "Yahoo", "lastName" : "Baba" };
```



Javascript Object Literals Vs JSON

JavaScript Object

```
var person = { firstName : "Yahoo", lastName : "Baba" };
alert(person.firstName + " " + person.lastName);
```

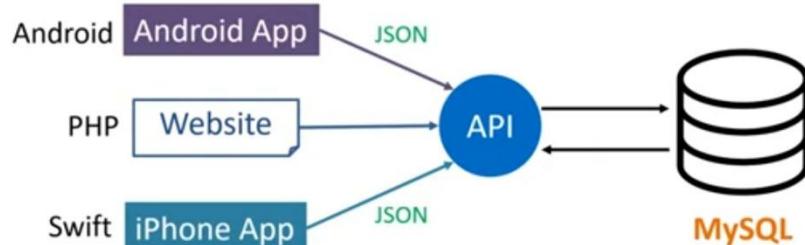
JSON

```
var person = { "firstName" : "Yahoo", "lastName" : "Baba" };
alert(person.firstName + " " + person.lastName);
```



What is API ?

Application Program Interface





Advantages of JSON :

- Human Readable Format
- Language Independent
- Supports all Popular Programming languages
- Easy to organise and access
- It is light-weight

Can not use it for transfer video, audio, images or any other binary information.



What we learn in next videos ?

- How to read JSON data in Website Front End
- How to produce JSON data in PHP
- How to work with APIs

What is Object ?

▶▶▶

0 1 2 3

```
var a = ["Ram", "Kumar", 18 , "India"];
```

```
var a = {  
    firstName : "Ram" ,  
    lastName : "Kumar" ,  
    age : 18 ,  
    country : "India"  
};
```

Yahoo
Baba



How to Target DOM Object :

▶▶▶

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

Yahoo
Baba



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



Yahoo
Baba

DOM Get Methods :



- innerText
- innerHTML
- getAttribute
- getAttributeNode
- Attributes

Yahoo
Baba

DOM Set Methods :



- innerText
- innerHTML
- setAttribute
- Attribute
- removeAttribute



Yahoo
Baba

The screenshot displays a dual-pane interface. On the left, an Atom code editor window titled 'variables.html' contains the following code:

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>
<script>
var firstname = "Yahoo";
firstname = "Baba";

document.write(firstname);

</script>
</head>
<body>

</body>
</html>
```

On the right, a browser window titled 'JavaScript' shows the output of the script: 'Baba'. Below the browser is a DevTools console window with the message 'Live reload enabled.' and the file path 'variables.html:40'.

Difference between Variable Types:



Var

Let

Const

```
var x = "Hello";
```

```
var x = "World";
```

```
x = "WoW";
```

```
let x = "Hello";
```

```
let x = "World";
```

```
x = "WoW";
```

```
const x = "Hello";
```

```
const x = "World";
```

```
x = "WoW";
```

Different Type of Assignment Operators :



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



Different Type of Data Types :

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



Different Type of Comparison Operators :

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



JavaScript Basic Events

- Click
- Double Click
- Right Click
- Mouse Hover
- Mouse Out
- Mouse Down
- Mouse Up
- Key Press
- Key Up
- Load
- Unload
- Resize
- Scroll



For Loop Syntax in JavaScript :

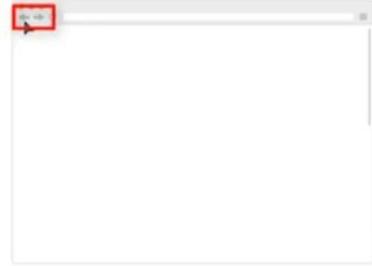
Initialization Condition Increment / Decrement

```
for(var a = 1; a <= 10; a++){
    document.write("Yahoo Baba");
}
```



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



Yahoo
Baba



JavaScript Math Methods :



Yahoo
Baba

- 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

JavaScript Math Methods :

- `toDateString()`
- `getDate()`
- `getFullYear()`
- `getMonth()`
- `getDay()`
- `getHours()`
- `getMinutes()`
- `getSeconds()`
- `getMilliseconds()`
- `setDate()`
- `setFullYear()`
- `setHours()`
- `setMilliseconds()`
- `setMinutes()`
- `setMonth()`
- `setSeconds()`

JavaScript Number Methods :

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

JavaScript Array Methods :

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

For / in Loop :

```
var a = {  
    firstName : "Ram" ,  
    lastName : "Kumar" ,  
    age : 18 ,  
    country : "India"  
};
```

```
for(var i in a){  
    Statement  
}
```

Array – Map() Function :

```
var a = [1 , 3 , 5 , 8 , 10];  
a.map(function(){  
    Statement  
});
```



How to Target DOM Object :

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

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



JavaScript Basic Events



- 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)

jQuery Quick API Reference		3.0	Search...	Preferences...
SELECTORS		ATTRIBUTES / CSS		MANIPULATION
Basics * .class .element .id selector1, selectorN, ...	Visibility Filters .hidden .visible Attribute [name] = "value" [name^= "value"] [name\$= "value"] [name*= "value"] [name~="value"] [name^~="value"] [name="value"] [name2="value2"]	Forms .button .checkbox .checked .disabled .enabled .focus .file .image .input .password .radio .reset .selected .submit .text	Attributes .attr() .prop() .checked .removeAttr() .removeProp() .val()	Dimensions .height() .innerHeight() .innerWidth() .outerHeight() .outerWidth() .width()
Hierarchy parent > child ancestor descendant prev + next prev ~ siblings			CSS .addClass() .css() jQuery.cssHooks jQuery.cssNumber jQuery.escapeSelector() .hasClass() .removeClass() .toggleClass()	Offset .offset() .offsetParent() .position() .scrollLeft() .scrollTop()
Basic Filters .animated .eq() .even .first .gt() .header .lang() .last .lt() .not() .odd .root .target	Child Filters .first-child .first-of-type .last-child .last-of-type .nth-child() .nth-last-child() .nth-last-of-type() .nth-of-type() .only-child .only-of-type()			Data jQuery.data() .data() jQuery.hasData() jQuery.removeData() .removeData()
Content Filters .contains() .empty .has() .parent				DOM Insertion, Inside .append() .appendTo() .html() .prepend() .prependTo() .text()
				DOM Insertion, Outside .after() .before() .insertAfter() .insertBefore()
				DOM Removal .detach() .empty() .remove() .unwrap()
				DOM Replacement .replaceAll() .replaceWith()

 Why use jQuery ?

- Short Selectors
- Variety of Animation Functions
- Easy DOM Manipulation
- Easy CSS Styling
- Easy DOM Traversing
- Simple Ajax Code

Yahoo Baba



Short Selectors Example :

```
document.getElementsByClassName('classname')  
$('.classname')
```

```
document.getElementById('idname')  
$('#idname')
```

```
document.getElementsByTagName('tagname')  
$('tagname')
```

① ② ③ ④ ⑤ ⑥

Yahoo
Baba

JavaScript

```
<script>  
    var a = document.getElementById("idName").innerHTML;  
    console.log(a);  
</script>
```

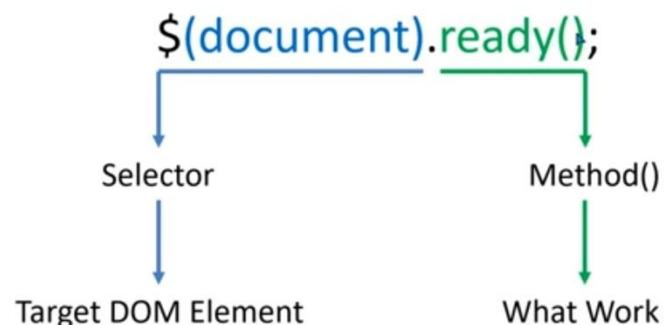
jQuery

```
<script>  
    $(document).ready(function(){  
        Further jQuery Code  
    });  
</script>
```

Yahoo
Baba



jQuery Basic Syntax



① ② ③ ④ ⑤ ⑥

Yahoo
Baba



Another Method to start jQuery Code :

```
$(document).ready(function(){
```

```
});
```

```
$(function(){
```

```
});
```

Yahoo
Baba



Targeting DOM Element with ID, Class & Tag

Select by Id : `$("#idName")`

Select by Class Name : `$(".className")`

Select by Tag Name : `$("tagName")` → `$("p")`

Yahoo
Baba



Advance Selectors

`$("*")`

`$("p:first")`

`$("ul li")`

`$("p:last")`

`$(".abc , .xyz")`

`$("li:even")`

`$("h1, div, p")`

`$("li:odd")`

Yahoo
Baba



Get Methods

- `text()`
- `html()`
- `attr()`
- `val()`

5:49 ④

VoIP LTE1 .11 VoIP LTE2 4G .11 69%

```
Syntax
$(document).ready(function(){
  $(".demo").click(function(){
    $(this).hide(200);
  });
});
$(function(){
  // Short Document Ready
});

Each
$(".demo").each(function() {
  document.write($(this).text() + "\n"); // output
});

Trigger
$("a#mylink").trigger("click"); // triggers event or

noConflict
var jq = $.noConflict();           // avoid conflict
jq(document).ready(function(){
  jq("#demo").text("Hello World!");
});
```

HTML
CSS
JS
jq
AMP
SEO

Selectors

```
$( "*")          // all elements
$("p.demo")      // <p> elements with class="demo"
$("p:first")     // the first <p> element
$("p span")      // span, descendant of p
 $("p > span")   // span, direct child of p
 $("p + span")   // span immediately preceding p
 $("p ~ span")   // strong element preceding p
 $("ul li:first") // the first <li> element
 $("ul li:first-child") // the first <li> element
 $("ul li:nth-child(3)") // third child
 $("[href]")       // any element with an href attribute
 $("a[target='_blank']") // <a> elements with a target attribute
 $("a[target!='_blank']") // <a> elements with a target attribute not '_blank'
 $(":input")        // all form elements
 $(":button")      // <button> and <input>
 $("tr:even")       // even <tr> elements
 $("tr:odd")        // odd <tr> elements
 $("span:parent")  // element which has children
 $("span:contains('demo')") // element containing the string "demo"
```

Actions

```
$(selector).action()
$(this).hide()      // the current element
$("div").hide()     // all <div> elements
$(".demo").hide()  // all elements with class="demo"
$("#demo").hide()  // the element with id="demo"
```

Events

```
$(".demo").click(function(){
  $(this).hide(200);
});
```

Mouse

scroll, click, dblclick, mousedown, mouseup, mousemove,
 mouseover, mouseout, mouseenter, mouseleave, load, resize,
 scroll, unload, error

Keyboard

keydown, keypress, keyup

Form

submit, change, focus, blur

DOM Element

blur, focus, focusin, focusout, change, select, submit

Syntax

```
$(document).ready(function(){
  $(".demo").click(function(){
    $(this).hide(200);
  });
  $(function(){
    // Short Document Ready
  });
});
```

HTML

CSS

JS

jq

AMP

Each

```
$(".demo").each(function() {
  // part of document
  document.write($(this).text() + "\n"); // output SEO
});
```

Trigger

```
$("#mylink").trigger("click"); // triggers event or
```

noConflict

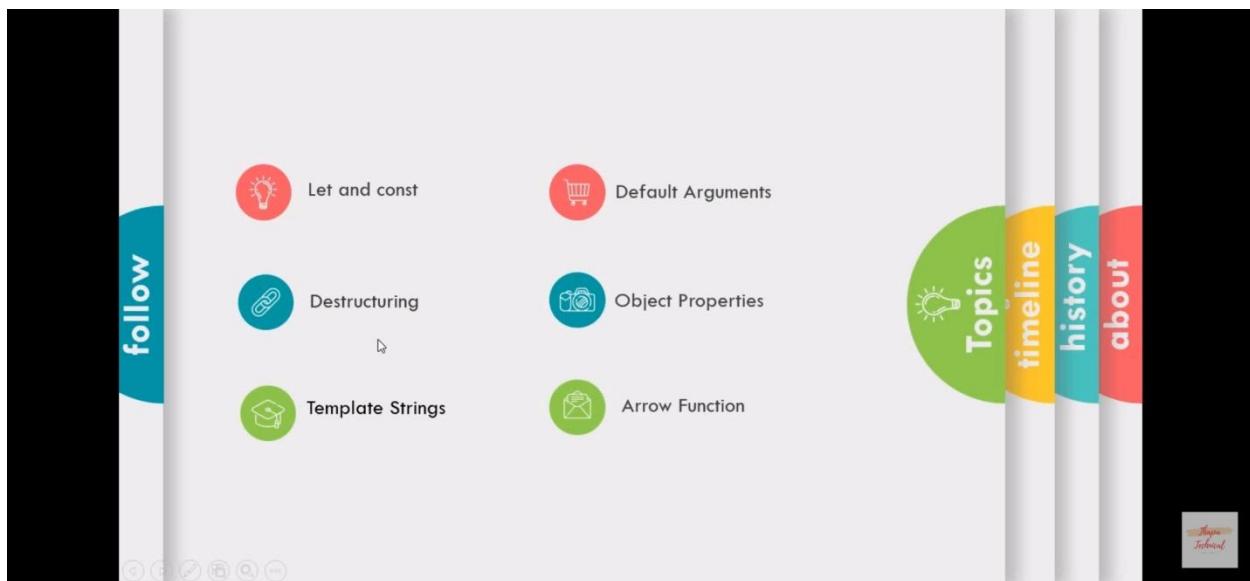
```
var jq = $.noConflict(); // avoid conflict
jq(document).ready(function(){
  jq("#demo").text("Hello World!");
});
```

Selectors

\$(*)	// all elements
\$(p.demo)	// <p> elements with class="demo"
\$(p:first)	// the first <p> element
\$(p span)	// span, descendant of p
\$(p > span)	// span, direct child of p
\$(p + span)	// span immediately preceding p
\$(p ~ span)	// strong element preceding p
\$(ul li:first)	// the first element in the list
\$(ul li:first-child)	// the first element in the list
\$(ul li:nth-child(3))	// third child
\$([href])	// any element with an href attribute
\$(a[target='_blank'])	// <a> elements with a target attribute
\$(a[target!='_blank'])	// <a> elements with a target attribute not equal to '_blank'
(:input)	// all form elements
(:button)	// <button> and <input>
(\$tr:even)	// even <tr> elements
(\$tr:odd)	// odd <tr> elements
(\$span:parent)	// element which has children
(\$span:contains('demo'))	// element containing the string "demo"

Actions

\$(selector).action()	
\$(this).hide()	// the current element
(\$div).hide()	// all <div> elements
(\$(".demo").hide())	// all elements with class="demo"



JS Promise Syntax

```
let prom = new Promise(function(resolve, reject){  
  if(condition){  
    resolve();  
  } else{  
    reject();  
  }  
});
```

```
let onfulfilment = (result) => {  
  console.log(result);  
}  
  
let onRejection = (error) => {  
  console.log(error);  
}
```

JS Fetch() Syntax

```
fetch();
```

```
fetch(file / URL);
```

Promise

```
fetch(file / URL).then();
```

Promise

```
fetch(file / URL).then(function(response){  
    return response.data;  
});
```

text()

json()

JS Different type of Variable :

ES6

Var

Let

ES6

Const

```
var x = "Hello";
```

```
var x = "World";
```

```
x = "WoW";
```

```
let x = "Hello";
```

```
let x = "World"; ✗
```

```
x = "WoW";
```

```
const x = "Hello";
```

```
const x = "World"; ✗
```

```
x = "WoW"; ✗
```

JS What is Fetch() Method ?

AJAX

JavaScript ES6

jQuery

```
$ajax();  
$.get();  
$.post();
```

Fetch()

- Insert
- Update
- Read
- Delete

JavaScript

XMLHttpRequest

Yahoo Baba

www.yahooibaba.net

Yahoo
Baba

JS What is Modules ?

USE

Variables

Functions

Classes

File1.js

File2.js

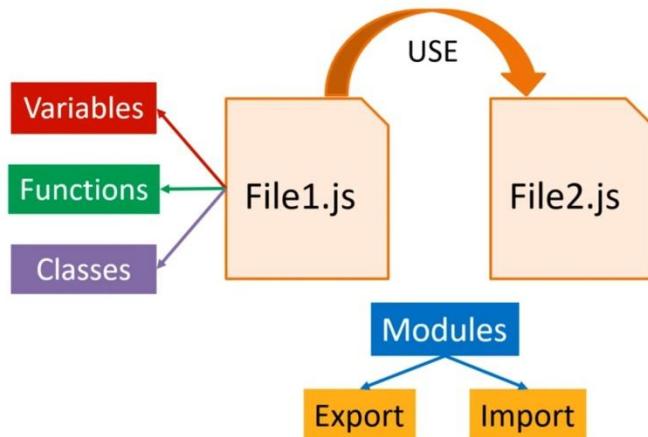
Modules

Yahoo Baba

www.yahooibaba.net

Yahoo
Baba

JS What is Modules ?



Yahoo Baba

www.yahooibaba.net

Yahoo
Baba

JS How to work with Modules ?

File1.js

```
export let name = "Yahoo Baba";
export function hello{
}
export class user{
}
```

File2.js

```
import { name } from './File1.js'
import { hello } from './File1.js'
import { user } from './File1.js'
console.log(name);
hello();
let a = new user();
```

HTML File

```
<script type="module" src="./File2.js"></script>
```

Yahoo Baba

www.yahooibaba.net

Yahoo
Baba

JS JavaScript : Data Types

```
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
```

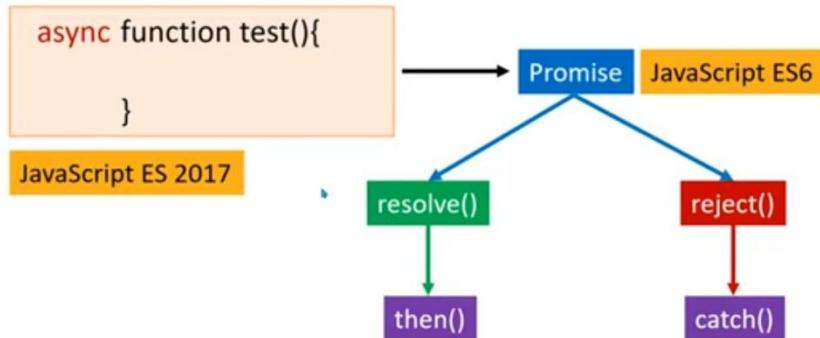
JS Symbol Data Type

```
var x = Symbol();  
  
var x = Symbol("Hello"); → Unique Value  
  
var y = Symbol("Hello");  
  
console.log(x === y) → False
```

JS JavaScript : Generators

```
function *test() {  
    Pause ←———— yield "First";  
    yield "Second";  
    yield "Third";  
}  
  
let g = test();  
g.next(); → First  
  
g.next();  
  
g.next();
```

JS Async Function

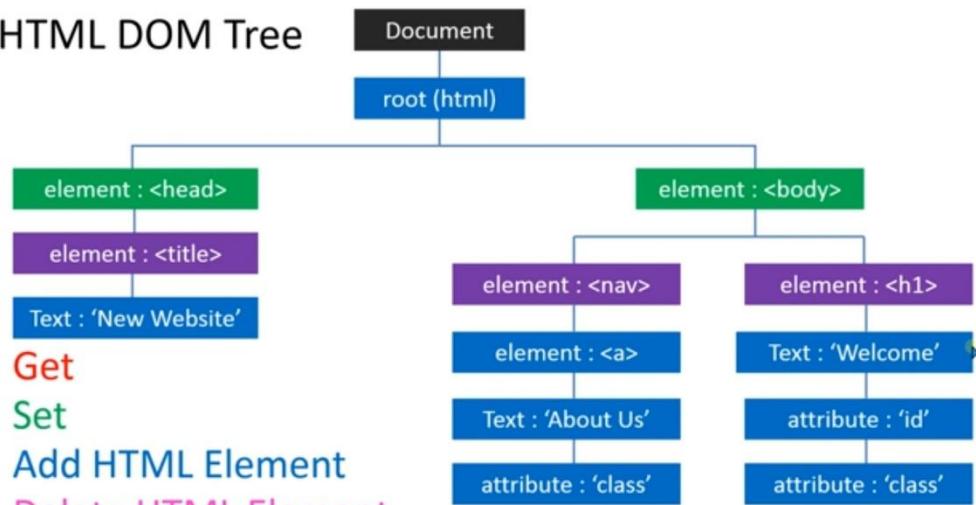


JS Await Method

```
async function test(){
    console.log("A");
    await console.log("B");
    console.log("C");
}

test();
console.log("D");
console.log("E");
```

HTML DOM Tree



DOM Get Methods :

- innerText
- innerHTML
- getAttribute
- getAttributeNode
- Attributes

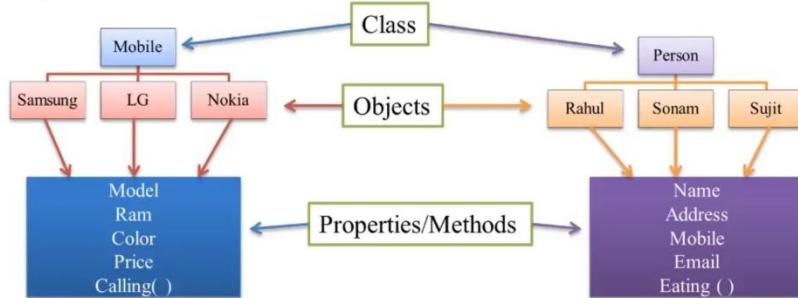
DOM Set Methods :

- innerText
- innerHTML
- setAttribute
- Attribute
- removeAttribute

Class

A specific category can be defined as class.

Example:-



www.geekyshows.com

Defining a Class

We define class in JavaScripts using custom constructor.

```
var Mobile = function(model_no, sprice) {
    this.model = model_no;
    this.color = 'white';
    this.price = 3000;
    this.sp = sprice;
    this.sellingprice = function() {
        return (this.price + this.sp);
    };
};

var samsung = new Mobile('Galaxy', 2000);
```

www.geekyshows.com

Defining a Class

We define class in JavaScripts using custom constructor.

```
var Mobile = function(model_no, sprice) {  
    this.model = model_no;  
    this.color = 'white';  
    this.price = 3000;  
    this.sp = sprice;  
    this.sellingprice = function() {  
        return (this.price + this.sp);  
    };  
};  
  
var samsung = new Mobile('Galaxy', 2000);  
var nokia = new Mobile('3310', 1000);  
↳
```

www.geekyshows.com

Defining a Class

We define class in JavaScripts using custom constructor.

```
var Mobile = function(model_no, sprice) {  
    this.model = model_no;  
    this.color = 'white';  
    this.price = 3000;  
    this.sp = sprice;  
    this.sellingprice = function() {  
        return (this.price + this.sp);  
    };  
};  
  
var samsung = new Mobile('Galaxy', 2000);  
var nokia = new Mobile('3310', 1000);
```

ES6 Class

```
class Mobile {  
    constructor () {  
        this.model = 'Galaxy';  
    }  
    show() { return this.model +  
              "Price 3000";  
    }  
}  
var nokia = new Mobile();
```

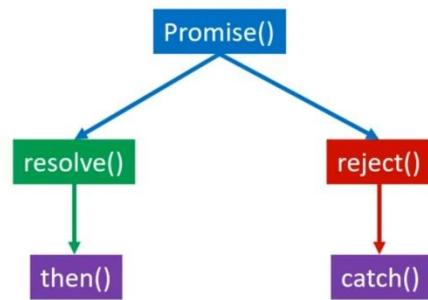
www.geekyshows.com

A screenshot of Notepad++ showing a JavaScript file named 'S.html'. The code defines a function 'add' that takes an array of numbers, initializes a sum to 0, iterates through the array using a for loop, and logs the sum to the console. The code is as follows:

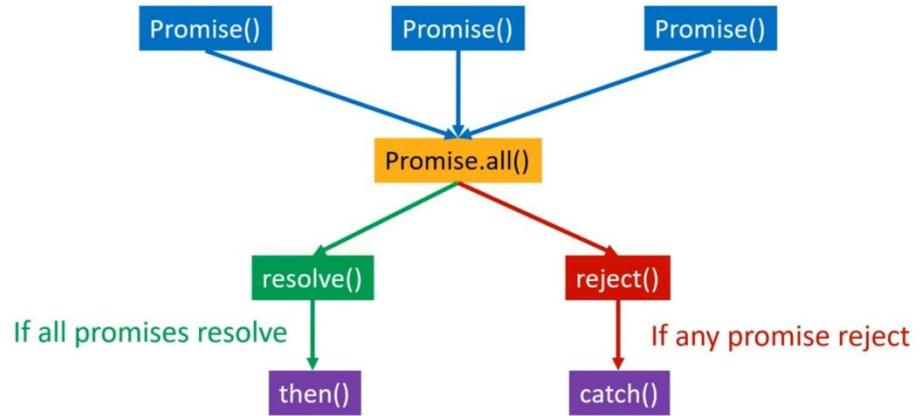
```
<script>
function add(...x) {
    let sum=0;
    for(i=0;i<x.length;i++) {
        sum=sum+x[i];
    }
    console.log(sum);
}
add(10,20,30,40,50);
</script>
```

The Notepad++ interface shows the file path 'D:\yob\es6\S.html', the file type 'Hyper Text Markup Language file', and various status bars at the bottom.

JS | What is Promise ?



JS | What is Promise.all() ?



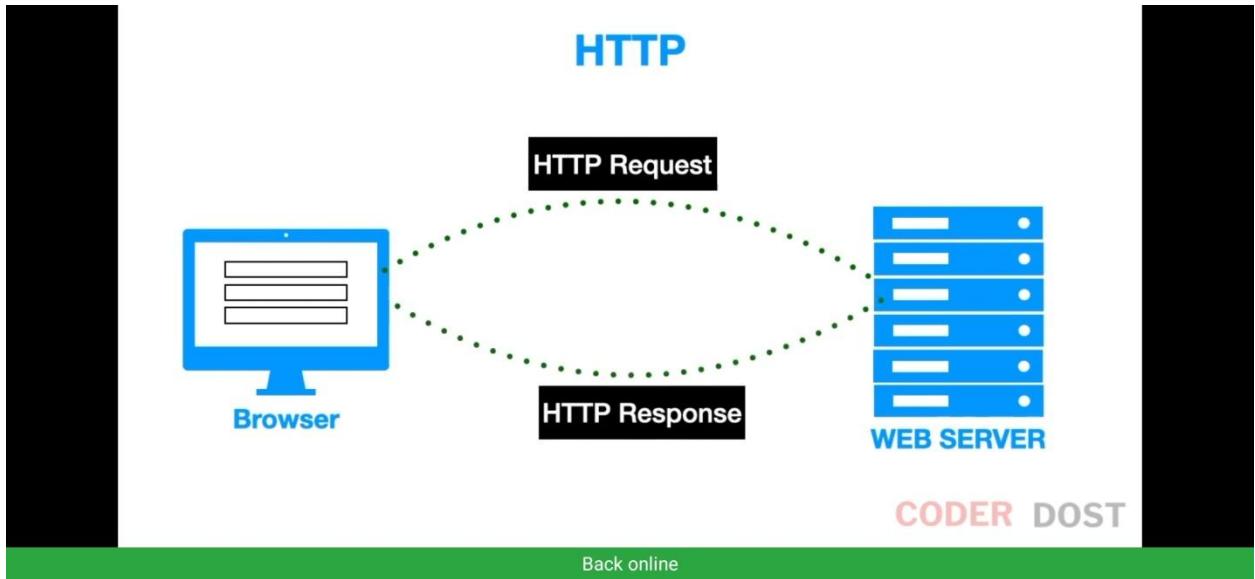
Yahoo Baba

www.yahooibaba.net

Promise : .then() & .catch()

```
const p = new Promise(function(resolve, reject){  
})  
p.then(function(data){  
})  
→ p.catch(function(data){  
})
```

CODER DOST

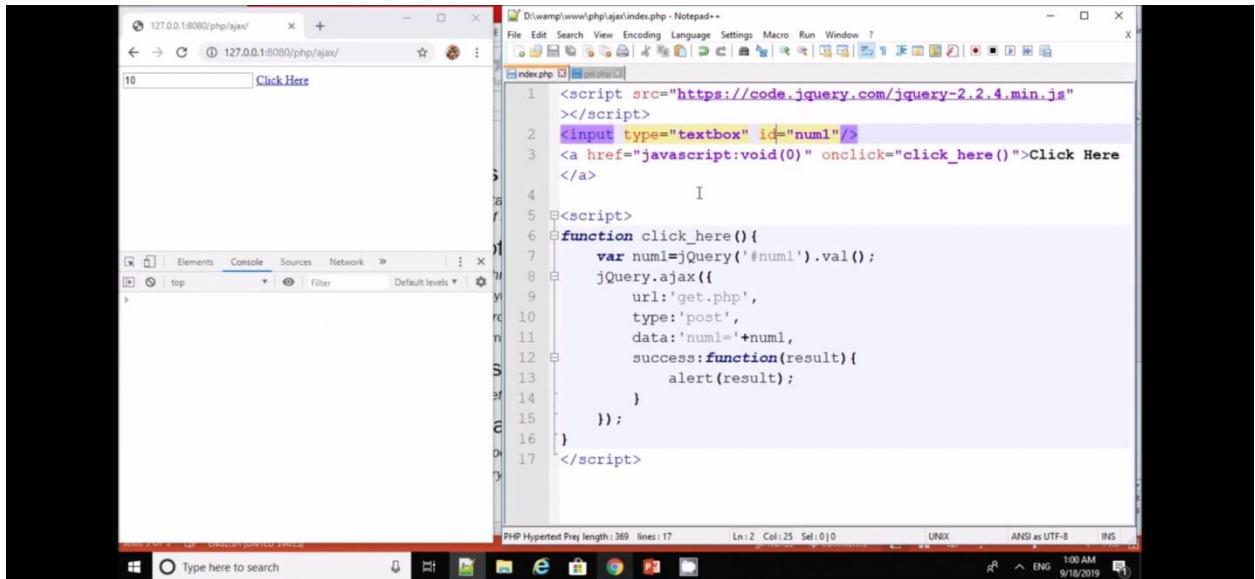


Recursion in Function

It is the process in which a function calls itself again and again to perform any repetitive task.

For Example

```
function fact(a)
{
    If(a==1)
        return 1;
    else
        return a*fact(a-1);
}
```



The screenshot shows a Windows desktop environment. In the center is a Notepad++ window titled "D:\wamp\www\php\ajax\index.php - Notepad++". The code in the editor is:

```
1 <script src="https://code.jquery.com/jquery-2.2.4.min.js">
2 </script>
3 <input type="textbox" id="num1"/>
4 <a href="javascript:void(0)" onclick="click_here()">Click Here
5 </a>
6 <script>
7   function click_here(){
8     var num1=$('#num1').val();
9     jQuery.ajax({
10       url:'get.php',
11       type:'post',
12       data:'num1='+num1,
13       success:function(result){
14         alert(result);
15       }
16     });
17 </script>
```

Below the Notepad++ window, the taskbar is visible with various icons. The status bar at the bottom of the screen displays "PHP Hypertext Preprocessor length: 369 lines: 17" and "Ln: 2 Col: 25 Sel: 0 | 0 UNIX ANSI as UTF-8 INS".

Callback Functions

```
var calc = function(fx,a,b){
    return fx(a,b);
}
```

```
var sum = function(x,y){
    return a+b;
}
```

```
calc(sum,4,5)
```

CODER DOST

The screenshot shows a browser's developer tools interface. On the left, there is a code editor window titled "JS app.js" containing the following JavaScript code:

```
1 function sum(a,b){  
2     return a+b;  
3 }  
4  
5 function calc(fx,a,b){  
6     return fx(a,b);  
7 }  
8  
9 console.log(calc(sum,4,5));
```

On the right, there is a "Console" tab showing the output of the code execution. The output is:

```
9  
Live reload enabled. (index):38  
>
```

A watermark "CODER DOST" is visible at the bottom right of the screenshot.

The screenshot shows a browser's developer tools interface. On the left, there is a code editor window titled "JS app.js" containing the following JavaScript code:

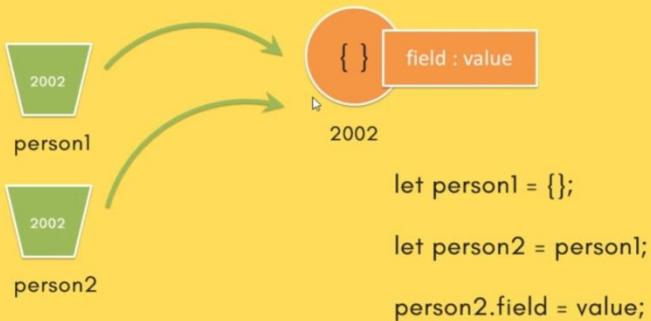
```
1 function sum(a,b){  
2     return a+b;  
3 }  
4 function diff(a,b){  
5     return a-b;  
6 }  
7  
8 function calc(fx,a,b){  
9     return fx(a,b);  
10 }  
11  
12 console.log(calc(diff,4,5));
```

On the right, there is a "Console" tab showing the output of the code execution. The output is:

```
-1  
Live reload enabled. (index):38  
>
```

A watermark "CODER DOST" is visible at the bottom right of the screenshot.

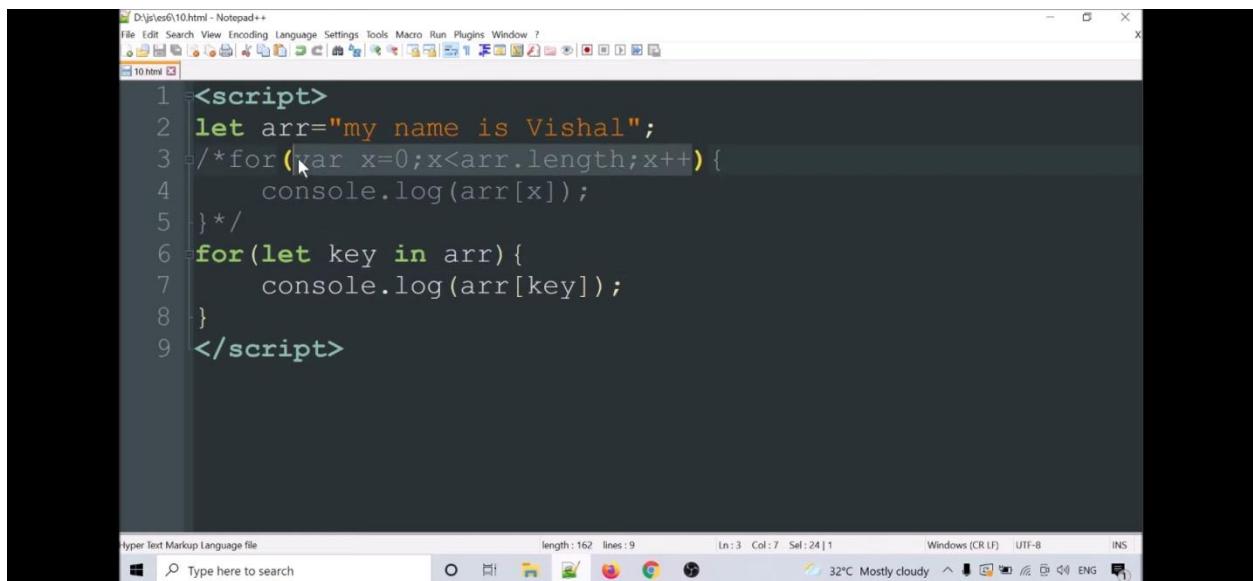
REFERENCE TYPES



CHEEZYCODE

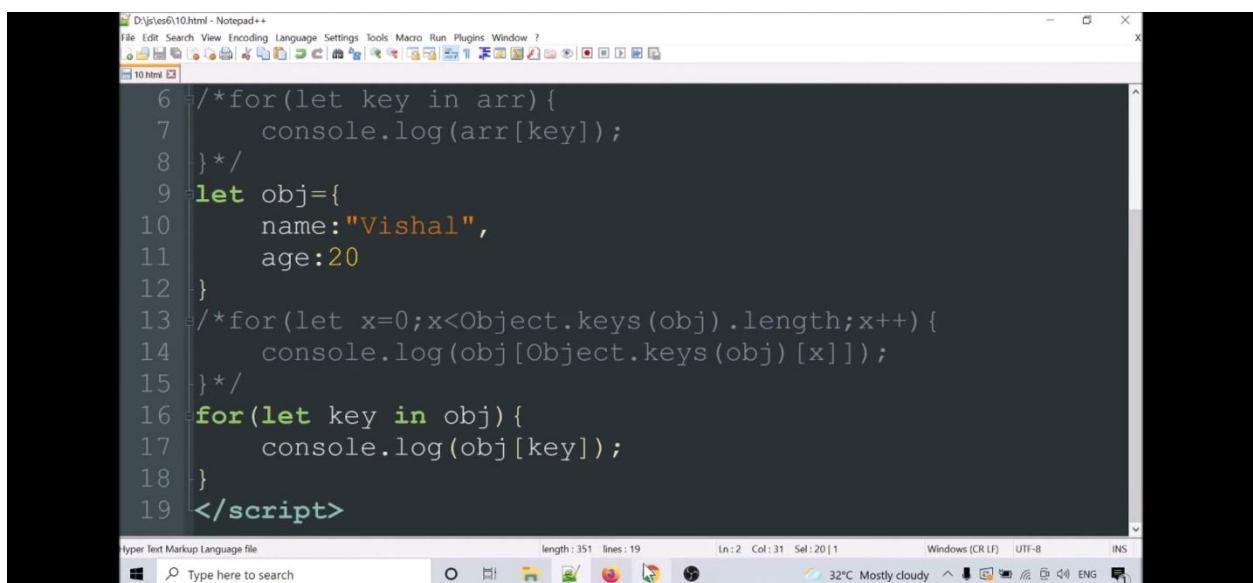
A screenshot of Sublime Text showing a file named `function.html`. The code is a simple JavaScript function that adds two numbers and prints the result to the console.

```
<script type="text/javascript">
var stored = sum(10, 20);
document.write("The sum of two no. is "+stored+"<br>");
function sum(a , b){
    var add = a + b ;
    return add;
}
</script>
```



```
D:\vishal\10.html - Notepad+
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
1 <script>
2 let arr="my name is Vishal";
3 /*for(var x=0;x<arr.length;x++){
4     console.log(arr[x]);
5 }*/
6 for(let key in arr){
7     console.log(arr[key]);
8 }
9 </script>

Hyper Text Markup Language file length : 162 lines : 9 Ln : 3 Col : 7 Sel : 24 | 1 Windows (CR LF) UTF-8 INS
Type here to search 32°C Mostly cloudy ENG
```



```
D:\vishal\10.html - Notepad+
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
10 /*for(let key in arr){
11     console.log(arr[key]);
12 }*/
13 let obj={
14     name:"Vishal",
15     age:20
16 }
17 /*for(let x=0;x<Object.keys(obj).length;x++) {
18     console.log(obj[Object.keys(obj)[x]]);
19 }*/
20 for(let key in obj){
21     console.log(obj[key]);
22 }
23 </script>

Hyper Text Markup Language file length : 351 lines : 19 Ln : 2 Col : 31 Sel : 20 | 1 Windows (CR LF) UTF-8 INS
Type here to search 32°C Mostly cloudy ENG
```

1. Encapsulation: -

Encapsulation means wrapping up data and member function (Method) together into a single unit i.e. class.



2. Abstraction: -

Abstraction is the process of showing only essential/necessary features of an entity/object to the outside world and hide the other irrelevant information. For example to open your TV we only have a power button, It is not required to understand how infra-red waves are getting generated in TV remote control.



3. Inheritance: -

Inheritance allows a class (subclass) to acquire the properties and behavior of another class (super-class). It helps to reuse, customize and enhance the existing code. So it helps to write a code accurately and reduce the development time.



4. Polymorphism:-

So polymorphism means "many forms". A subclass can define its own unique behavior and still share the same functionalities or behavior of its parent/base class.

```
class square(){
    area()
}

Var S1 = new square();
S1->area();
```

```
class circle(){
    area()
}

Var C1 = new circle();
C1->area();
```



File Edit Selection View Go Run Terminal Help index.html - js tutorial - Visual Studio Code

index.html X

index.html > html > body > script

```
7 <title>Document</title>
8 </head>
9 <body>
10 <h1>JavaScript Tutorial</h1>
11
12 <script>
13     let score = [80,39,70,54];
14
15     for(let x of score){
16         console.log(x);
17     }
18 </script>
19 </body>
20 </html>
```

Ln 16, Col 28 Spaces: 4 UTF-8 CRLF HTML Go Live ⚡ Q



A screenshot of Visual Studio Code showing the file `index.html`. The code contains a simple JavaScript script that logs "hello world" to the console. The code editor has syntax highlighting and line numbers. The status bar at the bottom shows the file path, line count (Ln 13, Col 35), and other settings.

```
<title>Document</title>
<body>
    <h1>JavaScript Tutorial</h1>
    <script>
        let score = "hello world";
        for(let x of score){
            console.log(x);
        }
    </script>
</body>
</html>
```

A screenshot of Microsoft Word showing a document titled JavaScript Type Casting. The document content includes:

Converting one type of data value to another type is called as type casting.

There are **two** types of type casting:

- Implicit type casting (Coercion): Done by JavaScript Engine
- Explicit type casting (Conversion): Done by programmers

Implicit type casting:

If required JavaScript engine automatically converts one type of value to another type.
This is known as implicit type casting.

Ex:

```
document.write(2+ 4.3); // 6.3
document.write("2" + 2 ); // "22"
document.write(2 + "2"); // "22"
```

V:\jsamedev\foreachs.html • (jsamedev) - Sublime Text (UNREGISTERED)

FOLDERS

- jsamedev
 - JSGameCarRacingThe...
 - darkmode.html
 - foreachs.html
 - foreachs.html
 - form.html
 - JSGameCarRacingThe...
 - jsameday2.html
 - mainindex.html
 - prachtest

foreachs.html

foreachs.html

```
1 <script>
2
3 const myFavProg = ['JavaScript', 'PHP', 'Java', 'c', 'Python'];
4
5 // for of iterable objects...
6 iterate
7
8 arrays, strings
9
10 for(items of myFavProg){
11     console.log(items);
12 }
13
14 // console.log(myFavProg[0]);
15 // console.log(myFavProg[1]);
16 // console.log(myFavProg[2]);
17
18 // for (let x=0; x<myFavProg.length; x++){
19 //     console.log(myFavProg[x]);
20 }
```

INSERT MODE, 7 characters selected

Tab Size: 4 HTML

This screenshot shows a Sublime Text window with a dark theme. The file being edited is 'foreachs.html' located in the 'jsamedev' folder. The code demonstrates the use of the 'for...of' loop to iterate over an array of programming languages. The code is color-coded, with 'arrays' and 'strings' highlighted in blue.

File Edit Selection View Go Run Terminal Help

index.html - js tutorial - Visual Studio Code

index.html

index.html > html > body > script

```
7 <title>Document</title>
8 </head>
9 <body>
10 <h1>JavaScript Tutorial</h1>
11
12 <script>
13     let score = [80,39,70,54];
14
15     for(let x of score){
16         console.log(x);
17     }
18 </script>
19 </body>
20 </html>
```

Ln 16, Col 28 Spaces: 4 UTF-8 CRLF HTML Go Live

This screenshot shows a Visual Studio Code window with a dark theme. The file being edited is 'index.html'. The code is identical to the one in Sublime Text, demonstrating the 'for...of' loop. The interface includes a sidebar with file navigation icons and a status bar at the bottom.

The `forEach()` method calls a function once for each element in an array, in order.

forEach() Method

Syntax: `arr.forEach(callback(currentValue [, index [, array]])[, thisArg])`

Argument	Description
<code>currentValue</code>	Required. The value of the current element
<code>index</code>	Optional. The array index of the current element
<code>arr</code>	Optional. The array object the current element belongs to

```
V:\jgamedev\foreach.html • (jgamedev) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  jgamedev
    JSGameCarRacingThe
      darkmode.html
      foreach.html
      foreachs.html
    JSGameCarRacingThe
      jgameday2.html
      mainindex.html
      prach.html
  4
  5 // console.log(myFavProg[0]);
  6 // console.log(myFavProg[1]);
  7 // console.log(myFavProg[2]);
  8
  9 // for (let x=0; x<myFavProg.length; x++){
 10 //   console.log(myFavProg[x]);
 11 // }
 12
 13 myFavProg.forEach(function(currValue,){
 14   console.log(currValue);
 15 })
 16
 17 </script>
```

INSERT MODE, Line 13, Column 38 Tab Size: 4 HTML

V:\jgamedev\foreachs.html (jgamedev) - Sublime Text (UNREGISTERED)

```
4
5 // console.log(myFavProg[0]);
6 // console.log(myFavProg[1]);
7 // console.log(myFavProg[2]);
8
9 // for (let x=0; x<myFavProg.length; x++){
10 //   console.log(myFavProg[x]);
11 // }
12
13 myFavProg.forEach(function(currValue){
14   console.log(currValue);
15 })
16
17 </script>
```

INSERT MODE, Line 13, Column 37

Tab Size: 4 HTML

V:\jgamedev\foreachs.html (jgamedev) - Sublime Text (UNREGISTERED)

```
1 <script>
2
3 const myFavProg = ['JavaScript', 'PHP', 'Java', 'c', 'Python'];
4
5 // console.log(myFavProg[0]);
6 // console.log(myFavProg[1]);
7 // console.log(myFavProg[2]);
8
9 // for (let x=0; x<myFavProg.length; x++){
10 //   console.log(myFavProg[x]);
11 // }
12
13 myFavProg.forEach(function(arrvalue, index ){
14   console.log(index + " -- " + arrvalue);
15 })
16
17 </script>
```

INSERT MODE, 4 characters selected

Tab Size: 4 HTML

In programming languages we often say “An object is an instance of a class”.

This means that, using a class, I can create many objects and they all share methods and properties.

Since objects can be enhanced, there are many ways to create objects sharing methods and properties. But we want the simplest one.

Bind `this`

For methods in React, the `this` keyword should represent the component that owns the method.

That is why you should use arrow functions. With arrow functions, `this` will always represent the object that defined the arrow function.

Why Arrow Functions?

In class components, the `this` keyword is not defined by default, so with regular functions the `this` keyword represents the object that called the method, which can be the global window object, a HTML button, or whatever.

Read more about binding `this` in our [React ES6 'What About this?'](#) chapter.

If you *must* use regular functions instead of arrow functions you have to bind `this` to the component instance using the `bind()` method:

bind() Method

by this method, we can bind an object to a common function, so that the function gives different result when its need.



```
<script>
    function Employee(firstName,lastName)
    {
        this.firstName=firstName;
        this.lastName=lastName;
    }

    Employee.prototype.company="Javatpoint"

    var employee1=new Employee("Martin","Roy");
    var employee2=new Employee("Duke", "William");
    document.writeln(employee1.firstName+" "+employee1.lastName+" "+employee1.company+"
    r>");
    document.writeln(employee2.firstName+" "+employee2.lastName+" "+employee2.company);
</script>
```

Let's see an example to add a new property to the constructor function.

```
<script>
    function Employee(firstName,lastName)
    {
        this.firstName=firstName;
        this.lastName=lastName;
    }

    Employee.prototype.company="Javatpoint"

    var employee1=new Employee("Martin","Roy");
    var employee2=new Employee("Duke", "William");
    document.writeln(employee1.firstName+" "+employee1.lastName+" "+employee1.company+"<b>" +
    "<r>");
    document.writeln(employee2.firstName+" "+employee2.lastName+" "+employee2.company);
</script>
```

The screenshot shows a Visual Studio Code interface with two tabs: `index.html` and `index.js`. The `index.js` tab contains the following code:

```
function Student(firstName, lastName, subject) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.subject = subject;
    this.greet = function(){
        console.log(`${this.firstName} ${this.lastName}`);
    }
}

let student1 = new Student("John", "Doe", "CS");
student1.greet();

function Employee(firstName, lastName, department) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.department = department;
    this.greet = function(){
        console.log(`${this.firstName} ${this.lastName}`);
    }
}

let emp1 = new Employee("Jason", "Smith");
emp1.greet();
```

To the right, a browser window displays the output of the console logs:

```
John Doe
index.js:6
Jason Smith
index.js:18
```

At the bottom right of the browser window, the text "CHEEZYCODE" is visible.

```
const person = {
  firstName:"John",
  lastName: "Doe",
  fullName: function () {
    return this.firstName + " " + this.lastName;
  }
}

console.log( person.fullName() ); // John Doe
```

JS



```
const person = {
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}

const person1 = {
  firstName:"John",
  lastName: "Doe"
}
const person2 = {
  firstName:"Mary",
  lastName: "Doe"
}

console.log( person.fullName.call(person1) ); // John Doe
console.log( person.fullName.call(person2) ); // Mary Doe
```

JS



```
const obj = { name: "John" };

let greeting = function(a,b){
    return `${a} ${this.name}. ${b}`;
};

console.log( greeting.call(obj, "Hello", "How are you?") );
// returns: Hello John. How are you?
```

JS



```
const obj = { name: "John" };

let greeting = function(a,b){
    return `${a} ${this.name}. ${b}`;
};

console.log( greeting.apply(obj, ["Hello", "How are you?"]) );
// returns: Hello John. How are you?
```

JS



```
const obj = { name: "John" };

let greeting = function(a,b){
    return `${a} ${this.name}. ${b}`;
}

let bound = greeting.bind(obj);

console.log( bound("Hello", "How are you?") );
// returns: Hello John. How are you?
```



Array – `findIndex()` Function :



```
var ages = [10 , 23 , 19 , 20];
var adultAge = 18;
ages >= adultAge
```

`findIndex(Function Name)`

`findIndex()` method returns the index of the first element in an array that pass a test



Array – find() Function :



```
var ages = [10 , 23 , 19 , 20];  
var adultAge = 18;  
ages >= adultAge
```

find(Function Name)

find() method returns the value of the first element in an array that pass a test



The screenshot shows a code editor on the left and a browser window on the right. The code editor displays a file named 'map.html' containing the following JavaScript code:

```
<!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>  
<body>
```

The browser window on the right shows the output of the code, which is the string "110,40,90,160".



Access Modifiers

	Class Itself	Outside Class	Derived Class
Public	✓	✓	✓
Protected	✓	✗	✓
Private	✓	✗	✗



Namespace

First.php

```
namespace first;  
class test{  
}
```

Second.php

```
namespace second;  
class test{  
}
```

Third.php

```
require First.php;  
require second.php;
```

```
$obj = new first\test();
```

Web notes

Class is a blue print of object.

memory is allocated for an object by Constructor.

No need of define variable in constructor.

Constructor function is a same as class.

Constructor is a method which is use to initialize object state.

this keyword ka use kewal object ya constructor function ke under hota hai

History of JavaScript

- JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images etc.
- Written by Brendan Eich in 1995.
- ECMAScript or ES released in 1997
- ES2 & ES3 released in 1998, 1999 resp.
- ES5 was released in 2009
- ES6 (or ES2015) was released in 2015
Anything greater than ES5 is called ESNext or ES2015+

14



Prerequisites

- JavaScript
- ES6
- JSX
- Babel
- Webpack
- Node
- PHP

15



Why do we use ESNext, JSX, Babel & Webpack?

- ESNext makes for simpler code that is easier to read and write.
- Convert ESNext & JSX syntax to code browser can understand
- Using transformation tools help in optimizing our code for widest variety of browsers.
- Organize your code into smaller modules and then bundle them together using Webpack into a single download

16



Why do we use ESNext, JSX, Babel & Webpack?

- Both webpack and Babel are tools written in JavaScript and run using Node.js (node)
- Node.js is a runtime environment for JavaScript outside of a browser
- node allows you to run JavaScript code on the command-line..

17





Cyber Technophile



What are COOKIES ?

- Cookies are small files, that a website stores in your browser which contain information pertaining to that particular website.
- Each time you return, the data, or 'cookie' is sent back to the site.
- After certain interval of time it gets deleted automatically.

Cookies may contain:

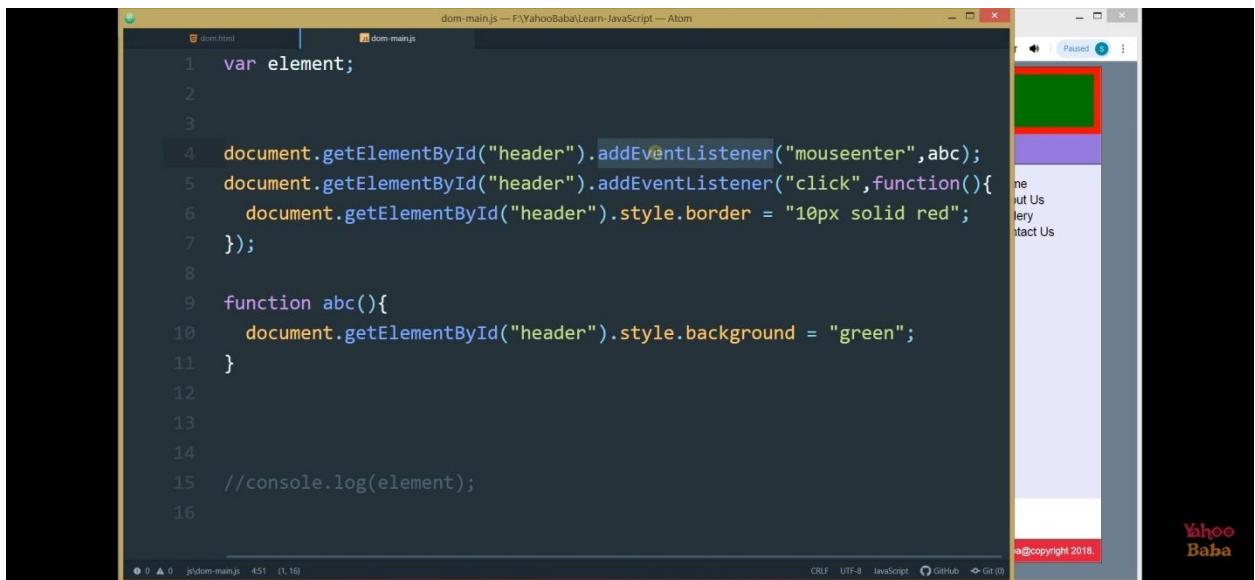
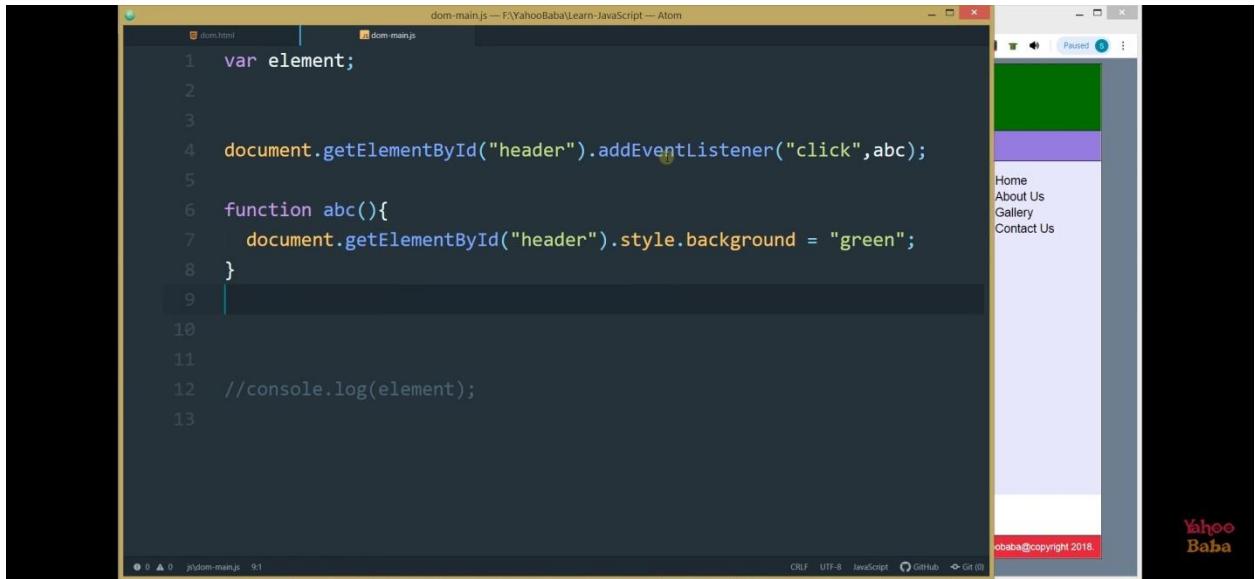
- Username and password
- links that you have clicked on
- Visitor tracking information
- Your cart information for buying online
- Your computer's general location in the world
- Videos you have watched



```
dom-main.js — F:\YahooBaba\Learn-JavaScript — Atom
1 var element;
2
3
4 document.getElementById("header").onclick = abc;
5
6 function abc(){
7   document.getElementById("header").style.background = "green";
8 }
9
10
11
12 //console.log(element);
13
```

CRFLF UTF-8 JavaScript GitHub Git (0)

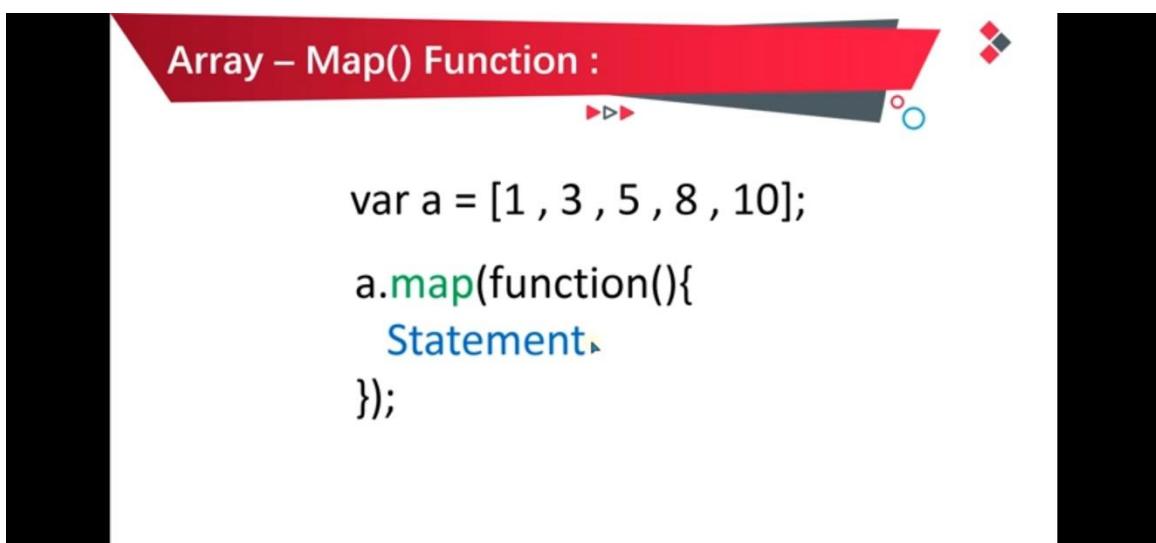
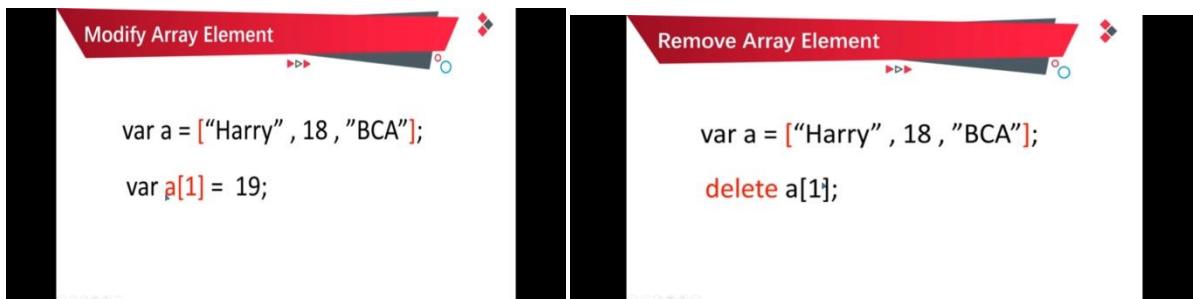
Yahoo Baba



The image shows a screenshot of a computer screen. On the left is a code editor window titled "array-modify.html" containing the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>
<script>
var a = ["Harry", 18, "Male", "BCA"];
</script>
</head>
<body>
</body>
</html>
```

On the right is a browser window titled "JavaScript" showing the output of the code, which is a list of four items: "Harry", 18, "Male", and "BCA".



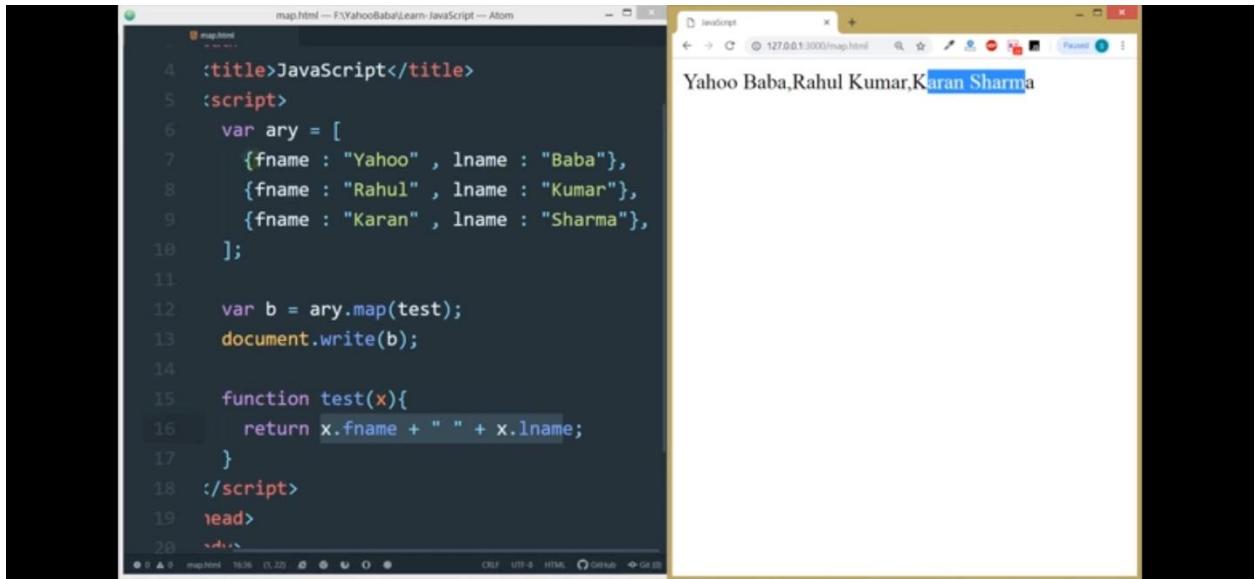
Array – Map() Function :

```
var a = [1 , 3 , 5 , 8 , 10];  
a.map(function(){  
    Statement  
});
```

The screenshot shows two windows side-by-side. On the left is a code editor window titled 'map.html' in 'Atom'. It contains the following JavaScript code:

```
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4     <title>JavaScript</title>  
5     <script>  
6         var ary = [11,4,9,1];  
7  
8         var b = ary.map(test);  
9         document.write(b);  
10  
11         function test(x){  
12             return x * 10;  
13         }  
14     </script>  
15 </head>  
16 <body>  
17
```

On the right is a browser window titled 'JavaScript' with the URL '127.0.0.1:3000/map.html'. The page displays the output of the script: '110,40,90,160'.

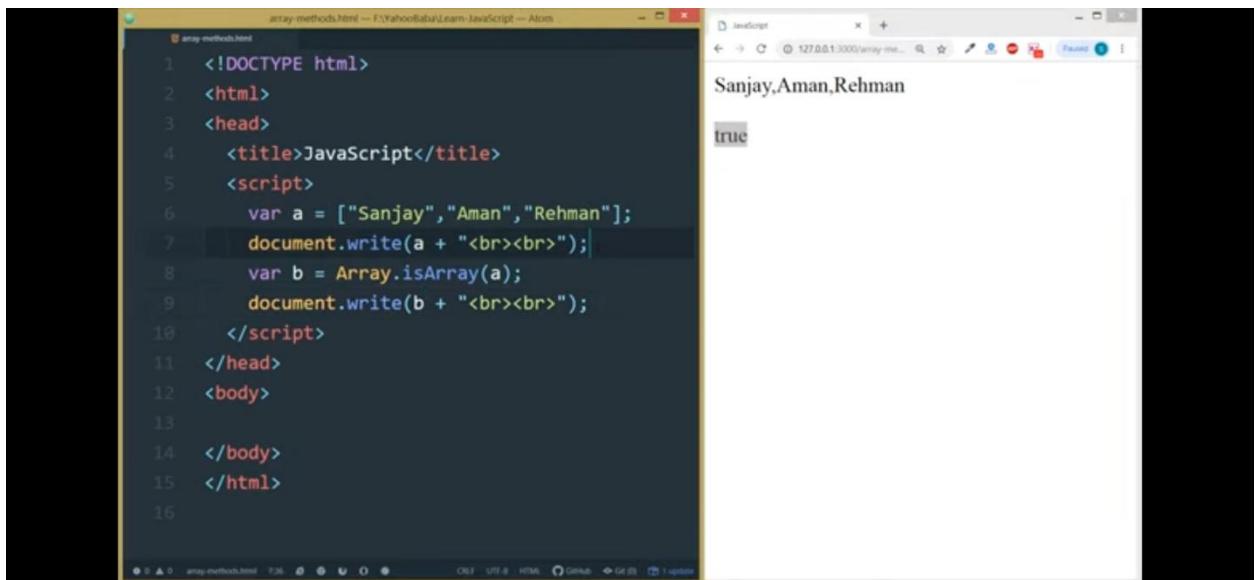


```
map.html — F:\YahooBaba\Learn-JavaScript — Atom
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>JavaScript</title>
5     <script>
6       var ary = [
7         {fname : "Yahoo" , lname : "Baba"}, 
8         {fname : "Rahul" , lname : "Kumar"}, 
9         {fname : "Karan" , lname : "Sharma"}, 
10        ];
11
12        var b = ary.map(test);
13        document.write(b);
14
15        function test(x){
16          return x.fname + " " + x.lname;
17        }
18      </script>
19    </head>
20  </html>
```

JavaScript

127.0.0.1:3000/map.html

Yahoo Baba,Rahul Kumar,Karan Sharma



```
array-methods.html — F:\YahooBaba\Learn-JavaScript — Atom
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>JavaScript</title>
5     <script>
6       var a = ["Sanjay","Aman","Rehman"];
7       document.write(a + "<br><br>");
8       var b = Array.isArray(a);
9       document.write(b + "<br><br>");
10      </script>
11    </head>
12    <body>
13
14    </body>
15  </html>
16
```

JavaScript

127.0.0.1:3000/array-me...

Sanjay,Aman,Rehman

true

The screenshot shows a code editor window titled "array-methods.html" and a browser window titled "JavaScript".

Code Editor Content:

```
1 <head>
2   <title>JavaScript</title>
3   <script>
4     //var a = ["Sanjay", "Aman", "Rehman"];
5     var a = "Rahul";
6     document.write(a + "<br><br>");
7
8
9
10    if(Array.isArray(a) == "true"){
11      document.write("This is an Array");
12    }else{
13      document.write("This is not an Array");
14    }
15
16  </script>
17 </head>
18 <body>
19
```

Browser Output:

Rahul
This is not an Array

Array – filter() Function :

```
var ages = [10 , 23 , 9 , 20];
var adultAge = 18;
ages >= adultAge —> [23 , 20]
```

filter(Function Name)

filter() method creates an array filled with all array elements that pass a test

The image shows a screenshot of a computer desktop. On the left is a code editor window titled "array-methods.html" in Atom. The code is as follows:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript</title>
    <script>
        var ages = [10,12,19,20];
        document.write(ages + "<br><br>");

        var b = ages.filter(checkAdult);
        document.write(b + "<br><br>");

        function checkAdult(age){
            return age >= 18;
        }
    </script>
</head>
<body>
```

On the right is a browser window titled "JavaScript" showing the output of the code. The output is:

10,12,19,20
19,20

Array – Join() Function :

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

Aman – Rehman – Karan

join()



The screenshot shows a code editor window for the file "array-methods.html". The code uses the `includes` method to check if the string "Aman" is present in the array `a`. The browser output window shows the result as "true".

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript</title>
    <script>
        var a = ["Sanjay", "Aman", "Rehman", "Rahul"];
        document.write(a + "<br><br>");

        var b = a.includes("Aman");
        document.write(b + "<br><br>");

    </script>
</head>
<body>

</body>
</html>
```

The screenshot shows a code editor window for the file "array-methods.html". The code uses the `every` method with a custom function `checkAdult` to check if all ages in the array are 18 or older. The browser output window shows the result as "false" because there are ages below 18 in the array.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript</title>
    <script>
        var ages = [10, 13, 15, 2];
        document.write(ages + "<br><br>");

        var b = ages.every(checkAdult);
        document.write(b + "<br><br>");

        function checkAdult(age){
            return age >= 18;
        }
    </script>
</head>
<body>

</body>
</html>
```

Array – indexOf() Function :

0 1 2 3

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

Aman → 1

indexOf("Search item", Start)



The image shows a dual-pane view. On the left, an Atom code editor displays the file 'array-methods.html' with the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript</title>
    <script>
        var a = ["Sanjay", "Aman", "Rehman"];
        var b = a.concat("Rahul", "Karan");
        document.write(b);
    </script>
</head>
<body>
</body>
</html>
```

On the right, a Microsoft Edge browser window shows the output of the script: "Sanjay,Aman,Rehman,Rahul,Karan".

The screenshot shows a Windows desktop environment. On the left, there is a code editor window titled "array-methods.html" in Atom. The code is a simple HTML file with a script that concatenates two arrays and prints the result to the console. On the right, there is a web browser window titled "JavaScript" showing the output of the code: "Sanjay,Aman,Rehman,Rahul,Karan". The browser's address bar shows the URL "127.0.0.1:3000/array-me...". In the bottom right corner of the screen, there is a small "Yahoo Baba" logo.

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript</title>
<script>
var a = ["Sanjay", "Aman", "Rehman"];
var b = ["Rahul", "Karan"];
var c = a.concat(b);
document.write(c);
</script>
</head>
<body>
</body>
</html>
```

The screenshot shows a web page with a yellow sidebar on the left containing a "JS" logo. The main content area has a white background and features a blue header "Converting an Arrays to Strings". Below the header, there is a list item with a blue circular icon followed by the text: "The JavaScript method `toString()` converts an array to a string of (comma separated) array values. JavaScript automatically converts an array to a comma separated string when a primitive value is expected." At the bottom right of the page, there is a watermark that says "Activate Windows Go to Settings to activate Windows".

Converting an Arrays to Strings

- The JavaScript method `toString()` converts an array to a string of (comma separated) array values. JavaScript automatically converts an array to a comma separated string when a primitive value is expected.

Activate Windows
Go to Settings to activate Windows.

```
97 <script>
98 // array.reduce(function(total, currentValue,
  currentIndex, arr), initialValue)
99 let array = [200,100,200,500];
100 var sum = 0;
101 array.forEach(function(item){
102   sum+=item;
103 })|
104 console.log(sum);
105 </script>
106 </body>
107 </html>
```

<dsc />

JavaScript 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



JavaScript Number Methods :

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



JavaScript Math Methods :

- toDateString()
- getDate()
- getFullYear()
- getMonth()
- getDay()
- getHours()
- getMinutes()
- getSeconds()
- getMilliseconds()
- setDate()
- setFullYear()
- setHours()
- setMilliseconds()
- setMinutes()
- setMonth()
- setSeconds()

DOM Get Methods :

- innerText
- innerHTML
- getAttribute
- getAttributeNode
- Attributes

Yahoo
Baba

The screenshot shows the Atom code editor with two files open: `dom.html` and `dom-main.js`. The `dom.html` file contains the following HTML:

```
<html>
<head>
    <title>My Website</title>
</head>
<body>
    <div id="content">
        <p>Aenean lacinia bibendum nulla sed consectetur.</p>
        <ul>
            <li>Home</li>
            <li>About Us</li>
            <li>Gallery</li>
            <li>Contact Us</li>
        </ul>
    </div>
</body>
</html>
```

The `dom-main.js` file contains the following JavaScript code:

```
var element;

element = document.getElementById("content").innerText;

console.log(element);
```

To the right of the editor, a browser window is displayed showing the rendered HTML. The page has a red header, a purple sidebar with a navigation menu, and a light blue footer. The main content area displays the text "Aenean lacinia bibendum nulla sed consectetur." and the navigation menu.

A screenshot of the Atom code editor interface. On the left, there are two tabs: 'dom.html' and 'dom-main.js'. The 'dom-main.js' tab contains the following JavaScript code:

```
1 var element;
2
3 element = document.getElementById("content").innerHTML;
4
5 console.log(element);
6
```

The code editor has a status bar at the bottom with icons for file operations, encoding (UTF-8), GitHub integration, and Git status.

On the right, a browser window displays a web page with a red header, a purple sidebar containing a navigation menu with items like 'Home', 'About Us', 'Gallery', and 'Contact Us', and a light blue footer. A developer tools panel is open, showing the DOM tree. The 'Elements' tab shows the 'content' element with its innerHTML. The 'Memory' tab shows memory usage. The 'Sources' tab lists 'dom-main.js:5' and 'dom.html:76'. The status bar at the bottom of the browser window also shows file operations, encoding, GitHub, and Git status.

A screenshot of the Atom code editor interface, similar to the one above. The 'dom-main.js' tab now contains this code:

```
1 ement;
2
3 t = document.getElementById("header").getAttribute("class");
4
5 e.log(element);
6
```

The code editor status bar shows file operations, encoding (UTF-8), GitHub, and Git status.

The browser window on the right shows the same web page structure. The developer tools panel is still open, showing the DOM tree. The 'Elements' tab shows the 'header' element with its class attribute. The 'Memory' tab shows memory usage. The 'Sources' tab lists 'dom-main.js:5' and 'dom.html:76'. The status bar at the bottom of the browser window shows file operations, encoding, GitHub, and Git status.

A screenshot of the Atom code editor showing a file named 'dom-main.js'. The code is as follows:

```
1  element;
2
3  element = document.getElementById("header").attributes[2].name;
4
5  console.log(element);
6
```

The code highlights the line `element = document.getElementById("header").attributes[2].name;` with a yellow background. To the right of the editor is a browser window displaying a simple website with a red header, a purple sidebar menu containing 'Home', 'About Us', 'Gallery', and 'Contact Us', and a light blue footer.

DOM Set Methods :

- `innerText`
- `innerHTML`
- `setAttribute`
- `Attribute`
- `removeAttribute`



The screenshot shows the Atom code editor with two tabs open: 'dom.html' and 'dom-main.js'. The 'dom-main.js' tab contains the following code:

```
1 var element;
2
3 document.getElementById("header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.getElementById("header").innerHTML; ⓘ
6
7 console.log(element);
8
```

The browser preview panel on the right shows a red header bar with a purple navigation menu containing links: Home, About Us, Gallery, and Contact Us.

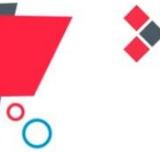
At the bottom of the Atom interface, the status bar displays: js\dom-main.js 7:3, CRLF, UTF-8, JavaScript, GitHub, Git (0), and a YahooBaba watermark.

The screenshot shows a browser window with developer tools open. The sidebar on the right contains a navigation menu with links to Home, About Us, Gallery, and Contact Us. The main area displays a piece of JavaScript code:

```
1 var element;  
2  
3 document.getElementById("header").innerHTML = "<h1>Wow</h1>";  
4  
5 document.getElementById("header").setAttribute("class","xyz");  
6  
7 element = document.getElementById("header").getAttribute("class");  
8  
9 console.log(element);  
10
```

The status bar at the bottom shows the video player controls, the current time (15:46 / 21:52), and the file path (dom-main.js — F:\YahooBaba\Learn-JavaScript — Atom).

New DOM Targeting Methods :



- **querySelector** → `document.querySelector(CSS Selector)`
- **querySelectorAll** → `document.querySelectorAll(CSS Selector)`



Yahoo
Baba

JavaScript DOM querySelector & querySelectorAll Tutorial in Hindi

```
1 var element;
2
3 document.querySelector("#header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.querySelector("#header").getAttribute("class");
6
7 console.log(element);
8
```

The screenshot shows a code editor window titled "dom-main.js" in the "F:\YahooBaba\Learn-JavaScript" directory using the Atom editor. The code demonstrates how to change the innerHTML of an element and retrieve its class attribute. On the right side of the screen, a browser developer tools panel is visible, showing the DOM tree with a red header element containing the text "Wow". The browser interface includes a toolbar with icons for refresh, search, and other developer tools, and a status bar at the bottom.

Screenshot of the Atom IDE showing a code completion dropdown for the `document.querySelectorAll` method. The code in the editor is:

```
1 var element;
2
3 document.querySelector("#header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.querySelectorAll(".list");
6 console.log(element);
7
```

The cursor is on the word `querySelectorAll`. A code actions dropdown is open with the following options:

- Select an action
- Extract to Function in global scope
- Extract to constant in enclosing scope

The right panel shows a preview of the DOM structure with a red header and a purple list item.

Screenshot of the Atom IDE showing a code completion dropdown for the `innerHTML` property of the first element in the array. The code in the editor is:

```
1 var element;
2
3 document.querySelector("#header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.querySelectorAll(".list")[1].innerHTML;
6 console.log(element);
7
```

The cursor is on the word `innerHTML`. A code actions dropdown is open with the following options:

- Select an action
- Extract to Function in global scope
- Extract to constant in enclosing scope

The right panel shows a preview of the DOM structure with a red header and a purple list item. The preview also includes some Latin text: "Ipsa, quam."

DOM CSS Styling Methods :



- **Style**
- **className**
- **classList**



Yahoo
Baba

The screenshot shows the Atom code editor interface. On the left, the code file `dom-main.js` contains the following JavaScript code:

```
1 var element;
2
3 document.querySelector("#header").style.backgroundColor = "tan";
4 document.querySelector("#header").style.color = "blue";
5 element = document.querySelector("#header").style.color;
6
7 console.log(element);
8
```

On the right, the `main.css` file contains the following CSS code:

```
header main.css:42
{
  padding: ▶ 30px 0
    0 20px;
  margin: ▶ 0; Yahoo
    Baba
```

The browser preview window shows a vertical bar with four colored segments: tan, blue, purple, and dark blue.

A screenshot of the Atom code editor. The main pane displays the following JavaScript code:

```
1 var element;
2
3 document.querySelector("#header").className = "abc";
4
5 element = document.querySelector("#header").className;
6
7 console.log(element);
8
```

The code editor interface includes tabs for 'dom.html', 'dom-main.js' (which is the active tab), 'color.css', and 'main.css'. A status bar at the bottom shows file statistics (0 0 ▲ 0 js\dom-main.js 5:54) and system information (CRLF, UTF-8, JavaScript, GitHub, Git (0)). On the right side, there's a vertical toolbar with icons for file operations and a color palette.

JavaScript Basic Events

- 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)

HTML Event Attributes :

```
<button onClick="abc()"></button>
```

```
<img src.." onmouseenter="xyz()">
```

YahOO
Baba

Assign Events Using the HTML DOM :

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

YahOO
Baba



The screenshot shows the Atom code editor with two tabs: 'dom.html' and 'dom-main.js'. The 'dom-main.js' tab contains the following code:

```
1 var element;
2
3
4 document.getElementById("header").onclick = abc;
5
6 function abc(){
7     document.getElementById("header").style.background = "green";
8 }
9
10
11 //console.log(element);
12
13
```

The right side of the screen shows a browser window with a header bar containing 'Home', 'About Us', 'Gallery', and 'Contact Us'. The background of the header bar is green, indicating that the JavaScript code has been executed.

DOM addEventListener() Method :

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



DOM addEventListener() Method :



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

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



Yahoo
Baba

The screenshot shows the Atom code editor with two tabs: 'dom.html' and 'dom-main.js'. The code in 'dom-main.js' is as follows:

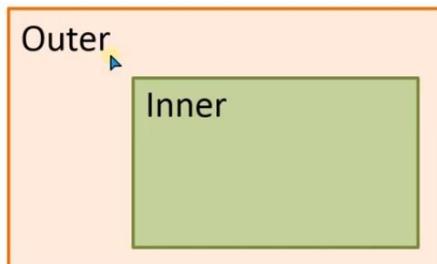
```
var element;  
  
document.getElementById("header").addEventListener("mouseenter",abc);  
document.getElementById("header").addEventListener("click",function(){  
    document.getElementById("header").style.border = "10px solid red";  
});  
  
function abc(){  
    document.getElementById("header").style.background = "green";  
}  
  
//console.log(element);
```

To the right of the editor is a browser window showing a header bar with a green background and a red border. The browser status bar at the bottom right says 'a@copyright 2018. Yahoo Baba'.

```
1 var element;
2
3
4 document.getElementById("header").addEventListener("click",abc);
5 document.getElementById("header").addEventListener("click",function(){
6     this.style.border = "10px solid red";
7 });
8
9 function abc(){
10     document.getElementById("header").style.background = "green";
11 }
12
13
14
15 //console.log(element);
16
```

UseCapture:

`addEventListener(event, function, useCapture);`



True

False

usecapture.js — P:\YahooBaba\Learn-JavaScript — Atom

JavaScript addEventListener Method Tutorial in Hindi / Urdu

```
1
2 document.querySelector("#inner").addEventListener('click',function(){
3     alert('Inner Div');
4 },true);
5
6 document.querySelector("#outer").addEventListener('click',function(){
7     alert('Outer Div');
8 },true)|①
9
```



12:53

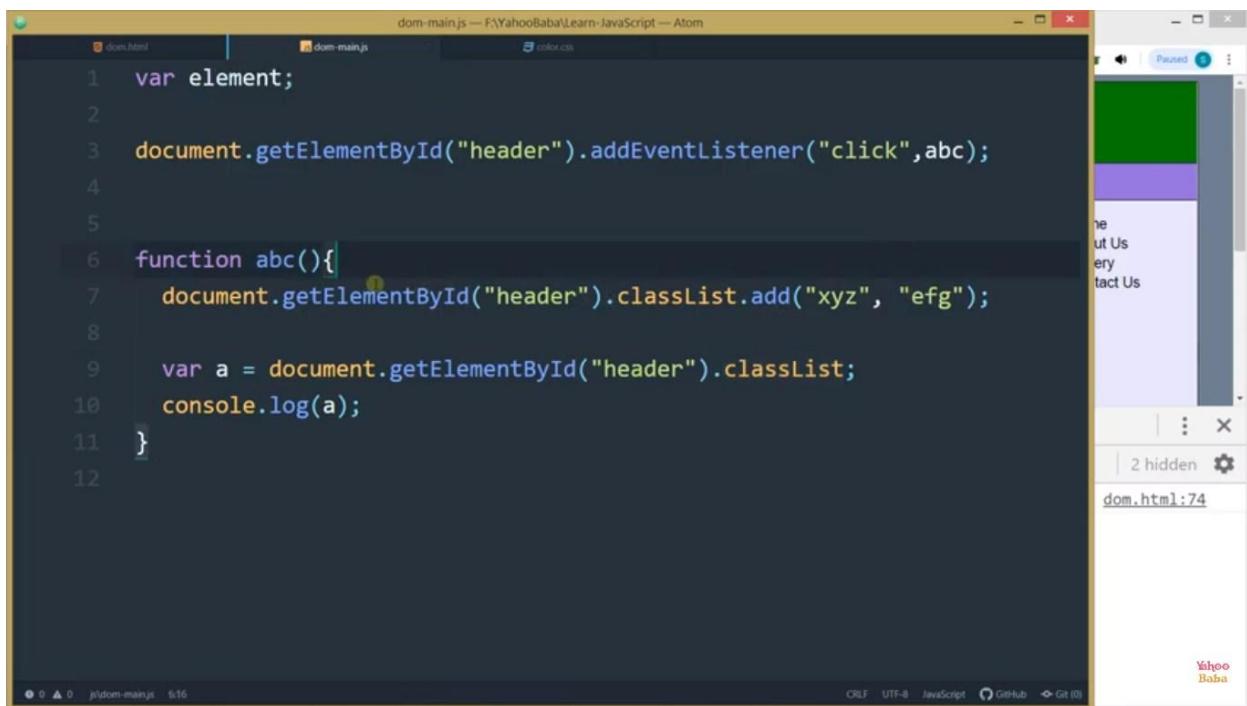
Yahoo Baba

0 Capture 13:08 / 18:04 CRLF UTF-8 JavaScript GitHub Git (0)

DOM removeEventListener() Method :

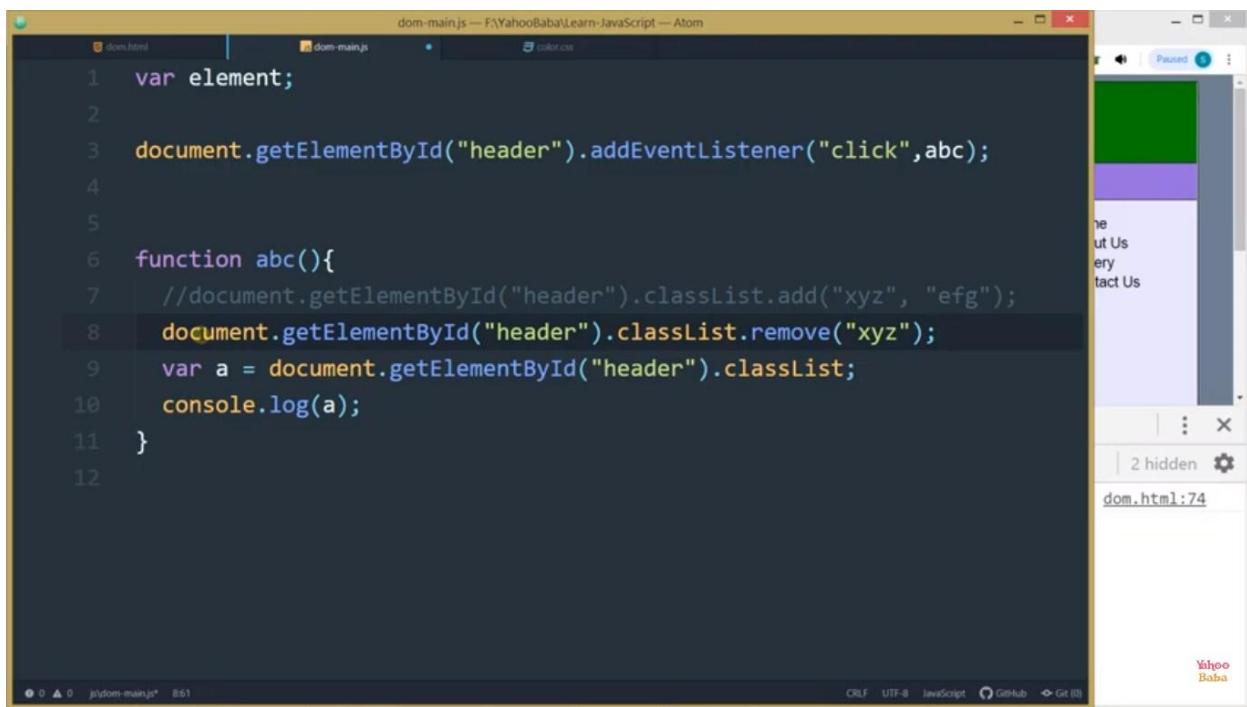
```
element.removeEventListener("ondblclick",functionName);
```





```
var element;
document.getElementById("header").addEventListener("click",abc);
function abc(){
    document.getElementById("header").classList.add("xyz", "efg");
    var a = document.getElementById("header").classList;
    console.log(a);
}
```

The screenshot shows the Atom code editor with a JavaScript file named 'dom-main.js'. The code defines a variable 'element', adds a click event listener to an element with ID 'header', and then defines a function 'abc' that adds the classes 'xyz' and 'efg' to the same element's class list. It also logs the modified class list to the console. A debugger sidebar on the right indicates the code is paused at line 12. The browser preview shows the element with the new classes applied.



```
var element;
document.getElementById("header").addEventListener("click",abc);
function abc(){
    //document.getElementById("header").classList.add("xyz", "efg");
    document.getElementById("header").classList.remove("xyz");
    var a = document.getElementById("header").classList;
    console.log(a);
}
```

This screenshot shows the same setup as the first one, but with a comment added to the line that adds classes. The code now removes the 'xyz' class from the element's class list instead. The browser preview shows the element with the 'xyz' class removed.

JavaScript classList Methods :



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

```
var element;

document.getElementById("header").addEventListener("mouseleave",abc);
document.getElementById("header").addEventListener('click', xyz);

function abc(){
    document.getElementById("header").style.background = "green";
}

function xyz(){
    document.getElementById("header").removeEventListener('mouseleave',abc);
}

//console.log(element);
```

The screenshot shows a browser window with a red header bar containing the text "Yahoo Baba". To the right of the header is a sidebar with several navigation links: Home, About Us, Gallery, and Contact Us. The main content area of the browser displays the code from the Atom editor. The code defines a variable 'element', sets up event listeners for 'mouseleave' and 'click' on an element with ID 'header', and contains two functions, 'abc' and 'xyz', which handle these events by changing the background color of the header element.

HTML Event Attributes :

```
<button onClick="abc()></button>
```

```
<img src.." onmouseenter="xyz()>
```



The screenshot shows a web browser window with a header menu containing "Home", "About Us", "Gallery", and "Contact Us". The background of the header is green when the mouse cursor is over the image element.

```
dom-main.js — F:\YahooBaba\Learn-JavaScript — Atom
JavaScript addEventListener Method Tutorial in Hindi / Urdu
1 var element;
2
3
4 document.getElementById("header").onclick = abc;
5
6 function abc(){
7   document.getElementById("header").style.background = "green";
8 }
9
10
11 //console.log(element);
12
13
```

DOM addEventListener() Method :



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

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



The screenshot shows a browser window with a green header bar containing the text "Yahoo Baba". Below the header, there is a purple rectangular area with some white text that is partially visible. The main content area of the browser shows a code editor for a file named "dom.html". The code contains JavaScript code that adds event listeners to an element with ID "header". When the mouse enters the header, its background turns green. When it is clicked, its border turns red.

```
var element;  
  
document.getElementById("header").addEventListener("mouseenter",abc);  
document.getElementById("header").addEventListener("click",function(){  
    document.getElementById("header").style.border = "10px solid red";  
});  
  
function abc(){  
    document.getElementById("header").style.background = "green";  
}  
  
//console.log(element);
```

dom-main.js — F:\YahooBaba\Learn-JavaScript — Atom

```
1 var element;
2
3
4 document.getElementById("header").addEventListener("click",abc);
5 document.getElementById("header").addEventListener("click",function(){
6     this.style.border = "10px solid red";
7 });
8
9 function abc(){
10     document.getElementById("header").style.background = "green";
11 }
12
13
14
15 //console.log(element);
16
```

usecapture.js — F:\YahooBaba\Learn-JavaScript — Atom

```
1
2 document.querySelector("#inner").addEventListener('click',function(){
3     alert('Inner Div');
4 },true);
5
6 document.querySelector("#outer").addEventListener('click',function(){
7     alert('Outer Div');
8 },true);
9
```

DOM removeEventListener() Method :



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



DOM Get Methods :

- innerText
- innerHTML
- getAttribute
- getAttributeNode
- Attributes

Yahoo
Baba

The screenshot shows the Atom code editor with two files open: `dom.html` and `dom-main.js`. The `dom.html` file contains the following HTML:

```
<html>
<head>
    <title>DOM Get Methods</title>
</head>
<body>
    <div id="content">
        <p>Aenean lacinia bibendum nulla sed consectetur.</p>
        <ul>
            <li>• Home</li>
            <li>• About Us</li>
            <li>• Gallery</li>
            <li>• Contact Us</li>
        </ul>
    </div>
</body>
</html>
```

The `dom-main.js` file contains the following JavaScript code:

```
var element;

element = document.getElementById("content").innerText;

console.log(element);
```

To the right of the editor, a browser window is displayed showing the rendered HTML. The page has a red header, a purple sidebar with a navigation menu, and a light blue footer. The main content area displays the text "Aenean lacinia bibendum nulla sed consectetur." and the navigation menu.

A screenshot of the Atom code editor interface. On the left, there are two tabs: 'dom.html' and 'dom-main.js'. The 'dom-main.js' tab contains the following JavaScript code:

```
1 var element;
2
3 element = document.getElementById("content").innerHTML;
4
5 console.log(element);
6
```

The code editor has a status bar at the bottom showing 'js\dom-main.js 6:1'. On the right side of the screen, there is a browser window displaying a webpage with a red header, a purple sidebar menu, and a light blue main content area. The sidebar menu includes links for Home, About Us, Gallery, and Contact Us. Below the browser window is a devtools panel titled 'Memory' which shows '2 hidden' items. The bottom right corner of the screen features the 'Yahoo Baba' logo.

A screenshot of the Atom code editor interface. On the left, there are two tabs: 'dom.html' and 'dom-main.js'. The 'dom-main.js' tab contains the following JavaScript code:

```
1 ement;
2
3 t = document.getElementById("header").getAttribute("class");
4
5 e.log(element);
6
```

The code editor has a status bar at the bottom showing 'js\dom-main.js 3:64'. On the right side of the screen, there is a browser window displaying a webpage with a red header, a purple sidebar menu, and a light blue main content area. The sidebar menu includes links for Home, About Us, Gallery, and Contact Us. Below the browser window is a devtools panel titled 'Memory' which shows '2 hidden' items. The bottom right corner of the screen features the 'Yahoo Baba' logo.

A screenshot of the Atom code editor showing a file named 'dom-main.js'. The code is as follows:

```
1  element;
2
3  element = document.getElementById("header").attributes[2].name;
4
5  console.log(element);
6
```

The code highlights the line `element = document.getElementById("header").attributes[2].name;` with a yellow background. To the right of the editor is a browser window displaying a simple website with a red header, a purple sidebar menu containing 'Home', 'About Us', 'Gallery', and 'Contact Us', and some placeholder text.

DOM Set Methods :

- innerText
- innerHTML
- setAttribute
- Attribute
- removeAttribute



The screenshot shows the Atom code editor with two tabs open: 'dom.html' and 'dom-main.js'. The 'dom-main.js' tab contains the following code:

```
1 var element;
2
3 document.getElementById("header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.getElementById("header").innerHTML; ⓘ
6
7 console.log(element);
8
```

The browser preview panel on the right shows a red header bar with a purple navigation menu containing links: Home, About Us, Gallery, and Contact Us.

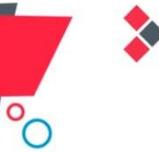
At the bottom of the Atom interface, the status bar displays: js\dom-main.js 7:3, CRLF, UTF-8, JavaScript, GitHub, Git (0), and a YahooBaba watermark.

The screenshot shows a browser window with developer tools open. The sidebar on the right contains a navigation menu with links to Home, About Us, Gallery, and Contact Us. The main content area displays a block of JavaScript code:

```
1 var element;  
2  
3 document.getElementById("header").innerHTML = "<h1>Wow</h1>";  
4  
5 document.getElementById("header").setAttribute("class","xyz");  
6  
7 element = document.getElementById("header").getAttribute("class");  
8  
9 console.log(element);  
10
```

The status bar at the bottom shows the video player controls, the current time (15:46 / 21:52), and the file path (dom-main.js — F:\YahooBaba\Learn-JavaScript — Atom).

New DOM Targeting Methods :



- **querySelector** → `document.querySelector(CSS Selector)`
- **querySelectorAll** → `document.querySelectorAll(CSS Selector)`



Yahoo
Baba

JavaScript DOM querySelector & querySelectorAll Tutorial in Hindi

```
1 var element;
2
3 document.querySelector("#header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.querySelector("#header").getAttribute("class");
6
7 console.log(element);
8
```

The screenshot shows a code editor window titled "dom-main.js" in the "F:\YahooBaba\Learn-JavaScript" directory using the Atom editor. The code demonstrates how to change the innerHTML of an element and retrieve its class attribute. To the right of the editor is a browser developer tools panel showing the DOM tree. The header element has been modified to contain the text "Wow". The browser's status bar at the bottom indicates the video is at 3:57 / 9:45.

Screenshot of the Atom IDE interface. The left pane shows two files: `dom.html` and `dom-main.js`. The `dom-main.js` file contains the following code:`1 var element;
2
3 document.querySelector("#header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.querySelectorAll(".list");
6 console.log(element);
7`A code completion dropdown menu titled "Code Actions" is open over the line `element = document.querySelectorAll(".list");`. The menu options are: "Select an action", "Extract to Function in global scope", and "Extract to constant in enclosing scope". The right pane shows a preview of the DOM structure with a red header bar and a purple sidebar. The bottom status bar shows "js\dom-main.js* 5:43 (1, 5)".

Screenshot of the Atom IDE interface. The left pane shows the same `dom-main.js` file with the following code:`1 var element;
2
3 document.querySelector("#header").innerHTML = "<h1>Wow</h1>";
4
5 element = document.querySelectorAll(".list")[1].innerHTML;
6 console.log(element);
7`The code completion dropdown is now open over the line `element = document.querySelectorAll(".list")[1].innerHTML;`, showing the option "Select an action". The right pane shows the DOM preview with the "Wow" text in the first list item. The bottom status bar shows "js\dom-main.js* 5:58" and "dom.html:74".

DOM CSS Styling Methods :



- **Style**
- **className**
- **classList**



Yahoo
Baba

The screenshot shows the Atom code editor interface. On the left, the code file `dom-main.js` contains the following JavaScript code:

```
1 var element;
2
3 document.querySelector("#header").style.backgroundColor = "tan";
4 document.querySelector("#header").style.color = "blue";
5 element = document.querySelector("#header").style.color;
6
7 console.log(element);
8
```

On the right, the `main.css` file contains the following CSS code:

```
header main.css:42
{
  padding: ▶ 30px 0
    0 20px;
  margin: ▶ 0; Yahoo
    Baba
```

The browser preview window shows a vertical bar with four colored segments: tan, blue, purple, and dark blue.

A screenshot of the Atom code editor. The main pane displays the following JavaScript code:

```
1 var element;
2
3 document.querySelector("#header").className = "abc";
4
5 element = document.querySelector("#header").className;
6
7 console.log(element);
8
```

The code editor interface includes tabs for 'dom.html', 'dom-main.js' (which is the active tab), 'color.css', and 'main.css'. A status bar at the bottom shows file statistics (0 0 ▲ 0 js\dom-main.js 5:54) and system information (CRLF, UTF-8, JavaScript, GitHub, Git (0)). On the right side, there's a vertical toolbar with icons for file operations and a color palette.

JavaScript Basic Events

- 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)

HTML Event Attributes :

```
<button onClick="abc()"></button>
```

```
<img src.." onmouseenter="xyz()">
```

YahOO
Baba

Assign Events Using the HTML DOM :

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

YahOO
Baba

The screenshot shows the Atom code editor with two tabs: 'dom.html' and 'dom-main.js'. The 'dom-main.js' tab contains the following code:

```
1 var element;
2
3
4 document.getElementById("header").onclick = abc;
5
6 function abc(){
7     document.getElementById("header").style.background = "green";
8 }
9
10
11 //console.log(element);
12
13
```

The right side of the screen shows a browser window with a header bar containing 'Home', 'About Us', 'Gallery', and 'Contact Us'. The background of the header bar is green, indicating that the JavaScript code has been executed.

DOM addEventListener() Method :

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



DOM addEventListener() Method :



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

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



Yahoo
Baba

The screenshot shows the Atom code editor with two tabs: 'dom.html' and 'dom-main.js'. The code in 'dom-main.js' is as follows:

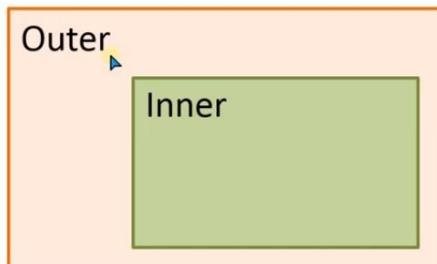
```
var element;  
  
document.getElementById("header").addEventListener("mouseenter",abc);  
document.getElementById("header").addEventListener("click",function(){  
    document.getElementById("header").style.border = "10px solid red";  
});  
  
function abc(){  
    document.getElementById("header").style.background = "green";  
}  
  
//console.log(element);
```

To the right of the editor is a browser window showing a header bar with 'Home', 'About Us', 'Gallery', and 'Contact Us' links. The 'About Us' link is highlighted with a green background. A status bar at the bottom of the browser window displays 'a@copyright 2018. Yahoo Baba'.

```
1 var element;
2
3
4 document.getElementById("header").addEventListener("click",abc);
5 document.getElementById("header").addEventListener("click",function(){
6     this.style.border = "10px solid red";
7 });
8
9 function abc(){
10     document.getElementById("header").style.background = "green";
11 }
12
13
14
15 //console.log(element);
16
```

UseCapture:

`addEventListener(event, function, useCapture);`



True

False

usecapture.js — P:\YahooBaba\Learn-JavaScript — Atom

JavaScript addEventListener Method Tutorial in Hindi / Urdu

```
1
2 document.querySelector("#inner").addEventListener('click',function(){
3     alert('Inner Div');
4 },true);
5
6 document.querySelector("#outer").addEventListener('click',function(){
7     alert('Outer Div');
8 },true)|①
9
```



12:53

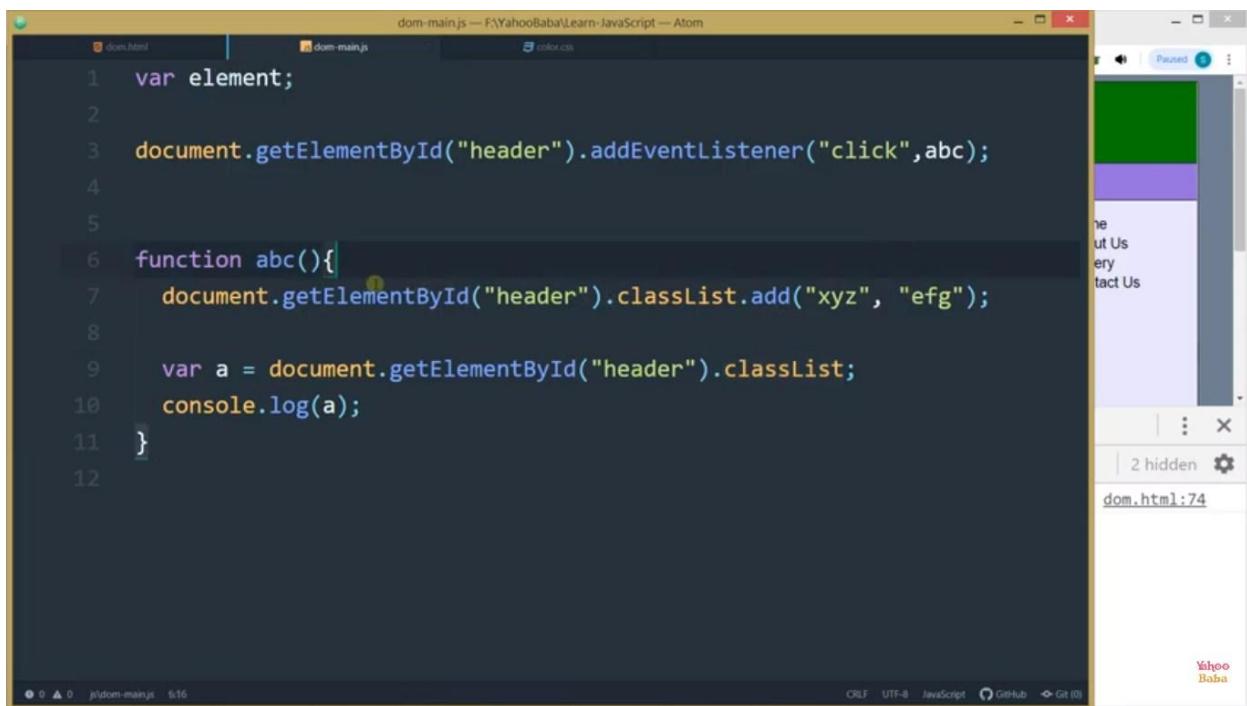
Yahoo Baba

0 Capture 13:08 / 18:04 CRLF UTF-8 JavaScript GitHub Git (0)

DOM removeEventListener() Method :

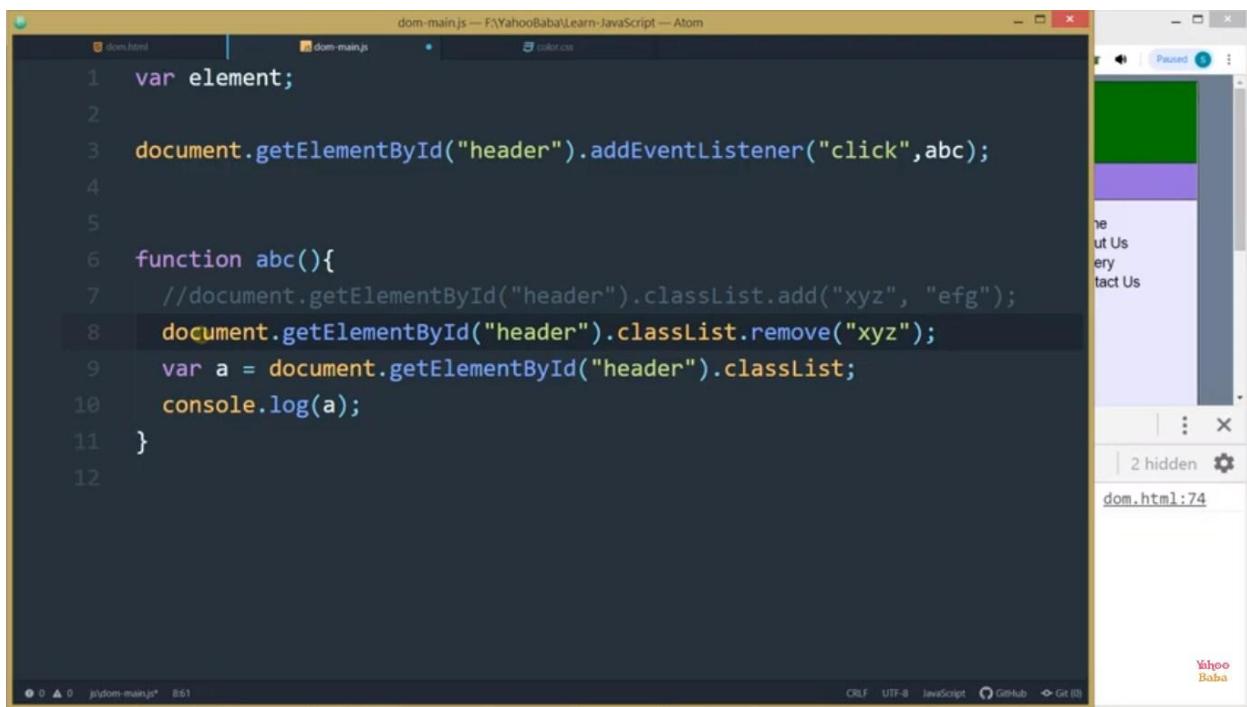
```
element.removeEventListener("ondblclick",functionName);
```





```
var element;
document.getElementById("header").addEventListener("click",abc);
function abc(){
    document.getElementById("header").classList.add("xyz", "efg");
    var a = document.getElementById("header").classList;
    console.log(a);
}
```

The screenshot shows the Atom code editor with a JavaScript file named 'dom-main.js'. The code defines a variable 'element', adds a click event listener to an element with ID 'header', and then defines a function 'abc' that adds the classes 'xyz' and 'efg' to the same element. It also logs the current classList to the console. A debugger sidebar on the right shows a stack trace with three frames: 'dom.html:74' (highlighted), 'dom-main.js:12', and 'color.css:1'. The status bar at the bottom indicates the file is 6:16.



```
var element;
document.getElementById("header").addEventListener("click",abc);
function abc(){
    //document.getElementById("header").classList.add("xyz", "efg");
    document.getElementById("header").classList.remove("xyz");
    var a = document.getElementById("header").classList;
    console.log(a);
}
```

This screenshot shows the same code as the first one, but with a change: the line 'document.getElementById("header").classList.add("xyz", "efg");' has been commented out with a double slash ('//'). The rest of the code remains the same. The debugger sidebar and status bar are identical to the first screenshot.

JavaScript classList Methods :



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

A screenshot of a browser window showing developer tools. The main content area has a red background and displays a purple sidebar with menu items: Home, About Us, Gallery, and Contact Us. The developer tools interface shows several tabs: 'dom.html' (active), 'dom-main.js', 'usecapture.html', and 'usecapture.js'. The code editor contains the following JavaScript code:

```
1 var element;
2
3
4 document.getElementById("header").addEventListener("mouseleave",abc);
5
6 document.getElementById("header").addEventListener('click', xyz);
7
8 function abc(){
9     document.getElementById("header").style.background = "green";
10 }
11
12 function xyz(){
13     document.getElementById("header").removeEventListener('mouseleave',ab
14 }
15
16
17 //console.log(element);
```

The status bar at the bottom of the Atom editor shows: 0 0 0 JS\dom-main.js 9:14 CRLF UTF-8 JavaScript GitHub Git {0}

HTML Event Attributes :

```
<button onClick="abc()></button>
```

```
<img src.." onmouseenter="xyz()>
```



The screenshot shows a web browser window with a header menu containing "Home", "About Us", "Gallery", and "Contact Us". The background of the header is green when the mouse cursor is over the image element. The browser interface includes a title bar, tabs, and a status bar at the bottom.

```
dom-main.js — F:\YahooBaba\Learn-JavaScript — Atom
JavaScript addEventListener Method Tutorial in Hindi / Urdu

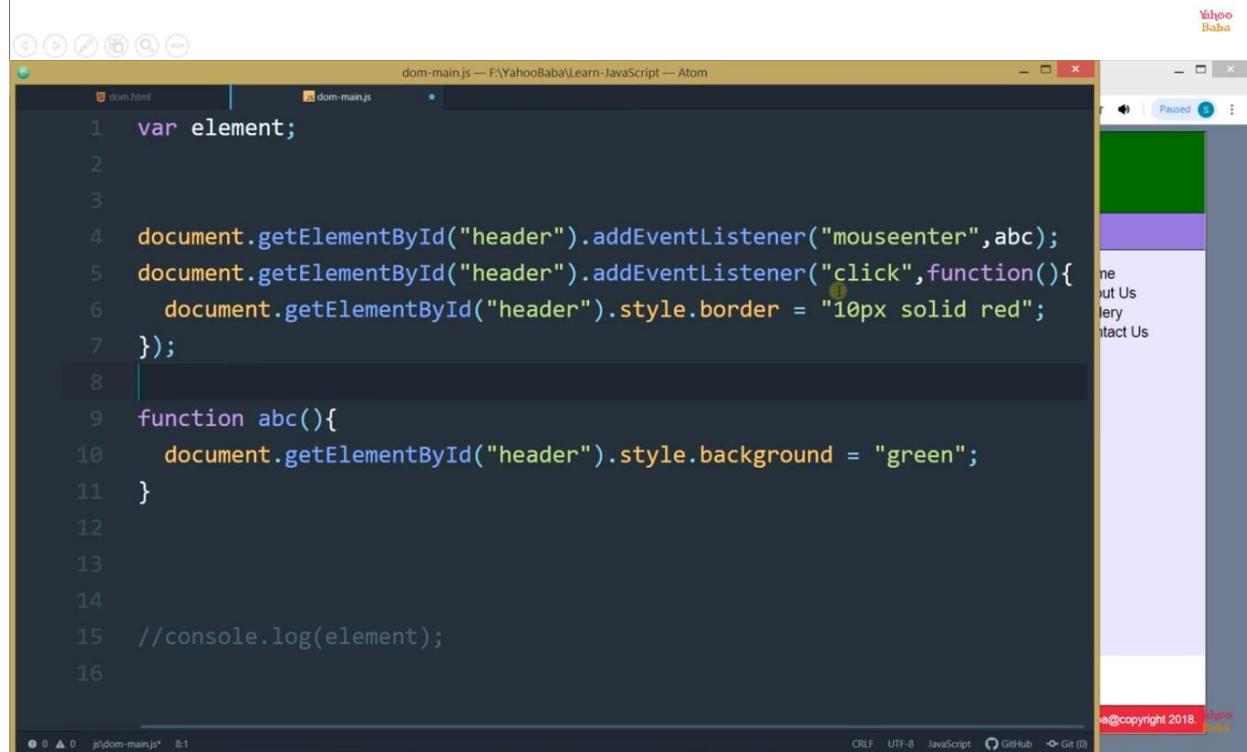
1 var element;
2
3
4 document.getElementById("header").onclick = abc;
5
6 function abc(){
7   document.getElementById("header").style.background = "green";
8 }
9
10
11 //console.log(element);
12
13
```

DOM addEventListener() Method :



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

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



A screenshot of a web browser window titled "Yahoo Baba". The header bar is styled with a solid red border. The page content shows a navigation menu with items like Home, About Us, Gallery, and Contact Us.

```
var element;  
  
document.getElementById("header").addEventListener("mouseenter",abc);  
document.getElementById("header").addEventListener("click",function(){  
    document.getElementById("header").style.border = "10px solid red";  
});  
  
function abc(){  
    document.getElementById("header").style.background = "green";  
}  
  
//console.log(element);
```

dom-main.js — F:\YahooBaba\Learn-JavaScript — Atom

```
1 var element;
2
3
4 document.getElementById("header").addEventListener("click",abc);
5 document.getElementById("header").addEventListener("click",function(){
6     this.style.border = "10px solid red";
7 });
8
9 function abc(){
10     document.getElementById("header").style.background = "green";
11 }
12
13
14
15 //console.log(element);
16
```

usecapture.js — F:\YahooBaba\Learn-JavaScript — Atom

```
1
2 document.querySelector("#inner").addEventListener('click',function(){
3     alert('Inner Div');
4 },true);
5
6 document.querySelector("#outer").addEventListener('click',function(){
7     alert('Outer Div');
8 },true);
9
```

DOM removeEventListener() Method :



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



The image shows three separate windows of the Notepad+ application, each displaying a different snippet of JavaScript code.

Top Window:

```
1 <script>
2 let arr="my name is Vishal";
3 /*for(var x=0;x<arr.length;x++){
4     console.log(arr[x]);
5 }*/
6 for(let key in arr){
7     console.log(arr[key]);
8 }
9 </script>
```

Middle Window:

```
6 /*for(let key in arr){
7     console.log(arr[key]);
8 }*/
9 let obj={
10     name:"Vishal",
11     age:20
12 }
13 /*for(let x=0;x<Object.keys(obj).length;x++) {
14     console.log(obj[Object.keys(obj)[x]]);
15 }*/
16 for(let key in obj){
17     console.log(obj[key]);
18 }
19 </script>
```

Bottom Window:

```
1 <script>
2 let arr="my name is Vishal";
3 for(let x=0;x<arr.length;x++){
4     console.log(arr[x]);
5 }
6 for(let key in arr){
7     console.log(arr[key]);
8 }
9 </script>
```

1. Encapsulation:-

Encapsulation means wrapping up data and member function (Method) together into a single unit i.e. class.

2. Abstraction: -

Abstraction is the process of showing only essential/necessary features of an entity/object to the outside world and hide the other irrelevant information. For example to open your TV we only have a power button, It is not required to understand how infra-red waves are getting generated in TV remote control.



3. Inheritance: -

Inheritance allows a class (subclass) to acquire the properties and behavior of another class (super-class). It helps to reuse, customize and enhance the existing code. So it helps to write a code accurately and reduce the development time.



4. Polymorphism: -

So polymorphism means "many forms". A subclass can define its own unique behavior and still share the same functionalities or behavior of its parent/base class.

```
class square(){  
    area()  
}
```

```
Var S1 = new square();  
S1->area();
```

```
class circle(){  
    area()  
}
```

```
Var C1 = new circle();  
C1->area();
```



The image shows three vertically stacked Microsoft Edge browser windows, each displaying a different version of the same HTML file ('index.html') in Visual Studio Code. The code in all three versions is identical, demonstrating a basic JavaScript loop that logs four numbers to the console.

```
<title>Document</title>
<body>
<h1>JavaScript Tutorial</h1>
<script>
let score = [80,39,70,54];
for(let x of score){
    console.log(x);
}
</script>
</body>
</html>
```

The browser windows are titled 'index.html - js tutorial - Visual Studio Code'. The status bar at the bottom of each window shows the line and column numbers (e.g., 'Ln 16, Col 28'), the number of spaces (e.g., 'Spaces: 4'), and the encoding (e.g., 'UTF-8'). The bottom-most window also shows the 'CRLF' option. The right side of the bottom-most window features the TechG logo.

JavaScript Type Casting

Converting one type of data value to another type is called as type casting.

There are **two** types of type casting:

- Implicit type casting (Coercion): Done by JavaScript Engine
- Explicit type casting (Conversion): Done by programmers

Implicit type casting:

If required JavaScript engine automatically converts one type of value to another type.
This is known as implicit type casting.

Ex:

```
document.write(2+4.3); // 6.3
document.write("2" + 2 ); // "22"
document.write(2 + "2"); // "22"
```

The image shows a dual-monitor setup. The left monitor displays Sublime Text with an open file named 'foreach.html'. The code in the editor is:

```
1 <script>
2
3 const myFavProg = ['JavaScript', 'PHP', 'Java', 'c', 'Python'];
4
5 // for of iterable objects...
6 iterate
7
8 arrays, strings
9
10 for(items of myFavProg){
11     console.log(items);
12 }
13
14 // console.log(myFavProg[0]);
15 // console.log(myFavProg[1]);
16 // console.log(myFavProg[2]);
17
18 // for (let x=0; x<myFavProg.length; x++) {
```

The right monitor displays Visual Studio Code with an open file named 'index.html'. The code in the editor is:

```
7 <title>Document</title>
8 </head>
9 <body>
10 <h1>JavaScript Tutorial</h1>
11
12 <script>
13     let score = [80,39,70,54];
14
15     for(let x of score){
16         console.log(x);
17     }
18 </script>
19 </body>
20 </html>
```

A callout box from the Sublime Text window points to the 'foreach.html' tab in VS Code, containing the text: "The forEach() method calls a function once for each element in an array, in order."

forEach() Method

Syntax: arr.forEach(callback(currentValue [, index [, array]])[, thisArg])

Argument	Description
currentValue	Required. The value of the current element
index	Optional. The array index of the current element
arr	Optional. The array object the current element belongs to

The image shows three vertically stacked instances of the Sublime Text editor, all displaying the same file content: `foreachs.html`. The code is a simple JavaScript snippet demonstrating a `foreach` loop.

```
4
5 // console.log(myFavProg[0]);
6 // console.log(myFavProg[1]);
7 // console.log(myFavProg[2]);
8
9 // for (let x=0; x<myFavProg.length; x++){
10 //   console.log(myFavProg[x]);
11 // }
12
13 myFavProg.forEach(function(currValue){
14   console.log(currValue);
15 })
16
17 </script>
```

The code is in `INSERT MODE` at Line 13, Column 38 in the first window, Line 13, Column 38 in the second window, and Line 13, Column 37 in the third window. The status bar in each window indicates `Tab Size: 4` and `HTML`.

In the bottom-most window, the status bar also shows `4 characters selected`.

In programming languages we often say “An object is an instance of a class”.

This means that, using a class, I can create many objects and they all share methods and properties.

Since objects can be enhanced, there are many ways to create objects sharing methods and properties. But we want the simplest one.

Bind `this`

For methods in React, the `this` keyword should represent the component that owns the method.

That is why you should use arrow functions. With arrow functions, `this` will always represent the object that defined the arrow function.

Why Arrow Functions?

In class components, the `this` keyword is not defined by default, so with regular functions the `this` keyword represents the object that called the method, which can be the global window object, a HTML button, or whatever.

Read more about binding `this` in our [React ES6 'What About this?'](#) chapter.

If you *must* use regular functions instead of arrow functions you have to bind `this` to the component instance using the `bind()` method:

bind() Method

by this method, we can bind an object to a common function, so that the function gives different result when its need.

Welcome to master CSS Class

```
margin: 0; padding: 0;
box-sizing: border-box;
}
.parent_div{
width: 100vw;
height: 100vh;
background: #2980b9;
display: flex;
justify-content: center;
align-items: center;
}

.child_div{
width: 60vw;
height: 60vh;
background:#1abc9c;
/*position: absolute;
left: 50%;
```

```

}
.parent_div{
width: 100vw;
height: 100vh;
background: #2980b9;
/*display: flex;
justify-content: center;
align-items: center;*/
display: grid;
place-items:center;
}

.child_div{
width: 60vw;
height: 60vh;
background:#1abc9c;
/*position: absolute;
left: 50%;
```

```

<script>
    function Employee(firstName,lastName)
    {
        this.firstName=firstName;
        this.lastName=lastName;
    }

    Employee.prototype.company="Javatpoint"

    var employee1=new Employee("Martin","Roy");
    var employee2=new Employee("Duke", "William");
    document.writeln(employee1.firstName+" "+employee1.lastName+" "+employee1.company+"<b>r"+");
    document.writeln(employee2.firstName+" "+employee2.lastName+" "+employee2.company);
</script>

```

Let's see an example to add a new property to the constructor function.

```

<script>
    function Employee(firstName,lastName)
    {
        this.firstName=firstName;
        this.lastName=lastName;
    }

    Employee.prototype.company="Javatpoint"

    var employee1=new Employee("Martin","Roy");
    var employee2=new Employee("Duke", "William");
    document.writeln(employee1.firstName+" "+employee1.lastName+" "+employee1.company+"<b>r"+");
    document.writeln(employee2.firstName+" "+employee2.lastName+" "+employee2.company);
</script>

```

The screenshot shows a Visual Studio Code interface with two tabs: 'index.html' and 'index.js'. The 'index.js' tab contains the following code:

```

function Student(firstName, lastName, subject) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.subject = subject;
    this.greet = function(){
        console.log(`${this.firstName} ${this.lastName}`);
    }
}

let student1 = new Student("John", "Doe", "CS");
student1.greet();

function Employee(firstName, lastName, department) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.department = department;
    this.greet = function(){
        console.log(`${this.firstName} ${this.lastName}`);
    }
}

let emp1 = new Employee("Jason", "Smith");
emp1.greet();

```

To the right, a browser window displays the output of the script. The console log entries are:

Output	Line Number
John Doe	index.js:6
Jason Smith	index.js:18

At the bottom right of the browser window, it says 'CHEEZYCODE'.

```
const person = {  
    firstName:"John",  
    lastName: "Doe",  
    fullName: function () {  
        return this.firstName + " " + this.lastName;  
    }  
}  
  
console.log( person.fullName() ); // John Doe
```

JS



```
const person = {  
    fullName: function() {  
        return this.firstName + " " + this.lastName;  
    }  
}  
  
const person1 = {  
    firstName:"John",  
    lastName: "Doe"  
}  
const person2 = {  
    firstName:"Mary",  
    lastName: "Doe"  
}  
  
console.log( person.fullName.call(person1) ); // John Doe  
console.log( person.fullName.call(person2) ); // Mary Doe
```

JS



```
const obj = { name: "John" };  
  
let greeting = function(a,b){  
    return `${a} ${this.name}. ${b}`;  
};  
  
greeting();  
  
console.log( greeting.call(obj, "Hello", "How are you?") );  
// returns: Hello John. How are you?
```

JS



```
const obj = { name: "John" };

let greeting = function(a,b){
    return `${a} ${this.name}. ${b}`;
};

console.log( greeting.apply(obj, ["Hello", "How are you?"]) );
// returns: Hello John. How are you?
```

JS



```
const obj = { name: "John" };

let greeting = function(a,b){
    return `${a} ${this.name}. ${b}`;
};

let bound = greeting.bind(obj);

console.log( bound("Hello", "How are you?") );
// returns: Hello John. How are you?
```

JS

Array – findIndex() Function :



```
var ages = [10 , 23 , 19 , 20];
```

```
var adultAge = 18;
```

```
ages >= adultAge
```

findIndex(Function Name)

findIndex() method returns the index of the first element in an array that pass a test



Array – find() Function :



```
var ages = [10 , 23 , 19 , 20];  
var adultAge = 18;  
ages >= adultAge
```

find(Function Name)

find() method returns the value of the first element in an array that pass a test

The screenshot displays a development environment with two windows. On the left, an Atom code editor shows the file 'map.html' with the following code:

```
<!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>  
<body>
```

On the right, a browser window titled 'JavaScript' shows the output of the code: "110,40,90,160". A watermark for "Yahoo Baba" is visible in the bottom right corner of the browser window.