



# PART 1

Brandon Eich



Version ES2015

## PROGRAMMING IN JAVASCRIPT-1995

# INTRODUCTION

2

HaarisInfotech – M.H. Shoiab

1. Objectives
2. Java Vs JavaScript
3. Elements of JavaScript
  1. Variables
  2. Expressions
  3. Control Statements
  4. Functions
  5. Objects and Arrays

At the end of this session, you will be able to:

- ❑ Write JavaScript code using all the basic elements of JavaScript Programs
- ❑ Create Windows & Dialog Boxes
- ❑ Use the JavaScript's in-built objects in a web page
- ❑ Write code that does Event Handling in HTML pages
- ❑ Manipulate HTML Forms through JavaScript dynamically

# INTRODUCTION

- An easy to use object scripting language
- Designed for creating live online applications
- Code is included as part of a HTML document
- Scripts are run by the Web browser
- JavaScript is a scripting language that is object-based but not completely object-oriented.
- It helps in creation of very dynamic web applications.
- Since it is an interpreted language, it can be directly embedded into HTML documents and it is executed by the built-in interpreter in the browsers.

# INTRODUCTION

## JavaScript Versus Java

- JavaScript can be combined directly with HTML
- The JavaScript language structure is simpler than that of Java
- JavaScript is a purely interpreted language
- The JavaScript interpreter is built into a Web browser

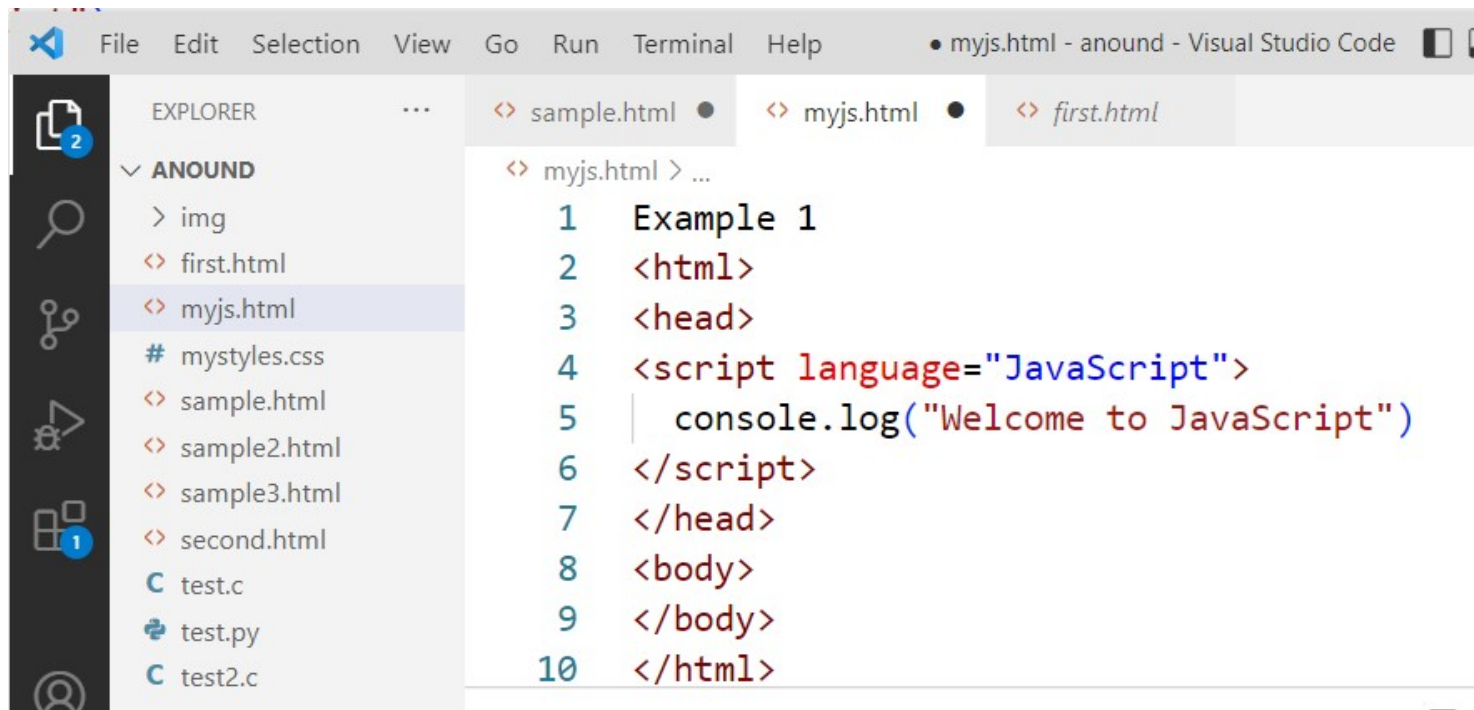
# INTRODUCTION

```
<HTML>
  <HEAD>
    <TITLE>Simple JavaScript Example </TITLE>
  </HEAD>
  <BODY>
    HTML Text goes here.
    <SCRIPT LANGUAGE="JavaScript">
      console.log("Welcome to JavaScript.")
    </SCRIPT>
  </BODY>
</HTML>
```

# How to Run the JS Program

6

HaarisInfotech – M.H. Shoiab



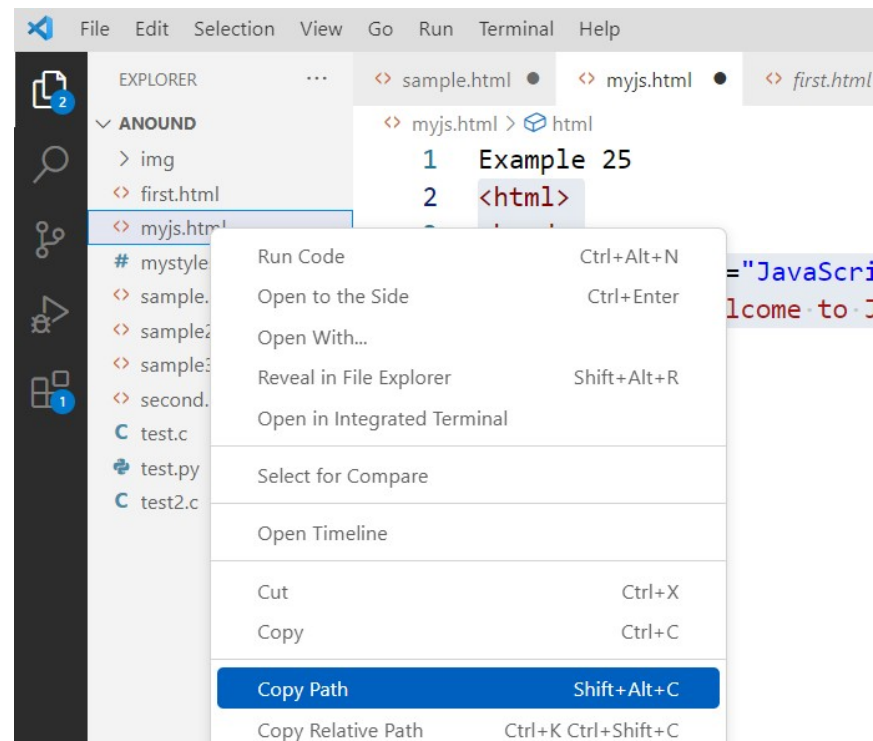
The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays a file tree for a project named 'ANOUND'. The file 'myjs.html' is selected. The main editor area shows the content of 'myjs.html', which is an HTML document with a JavaScript script that logs a welcome message to the console. The code is as follows:

```
<? myjs.html > ...  
1 Example 1  
2 <html>  
3 <head>  
4 <script language="JavaScript">  
5 | console.log("Welcome to JavaScript")  
6 </script>  
7 </head>  
8 <body>  
9 </body>  
10 </html>
```

# How to Run the JS Program

7

HaarisInfotech – M.H. Shoiab



# How to Run the JS Program

8

HaarisInfotech – M.H. Shoiab

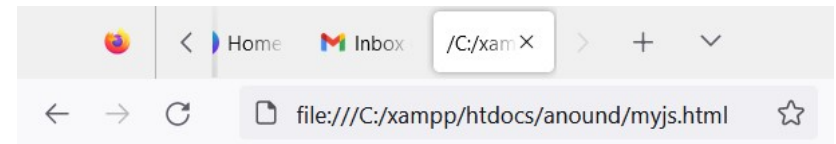
## Using Console Tab of Web Browsers

All the popular web browsers have built-in JavaScript engines. Hence, you can run JavaScript on a browser. To run JavaScript on a browser,

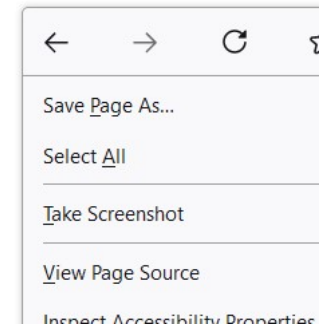
Open your favorite browser (here we will use Google Chrome).

Open the developer tools by right clicking on an empty area and select

**Inspect. Shortcut:**

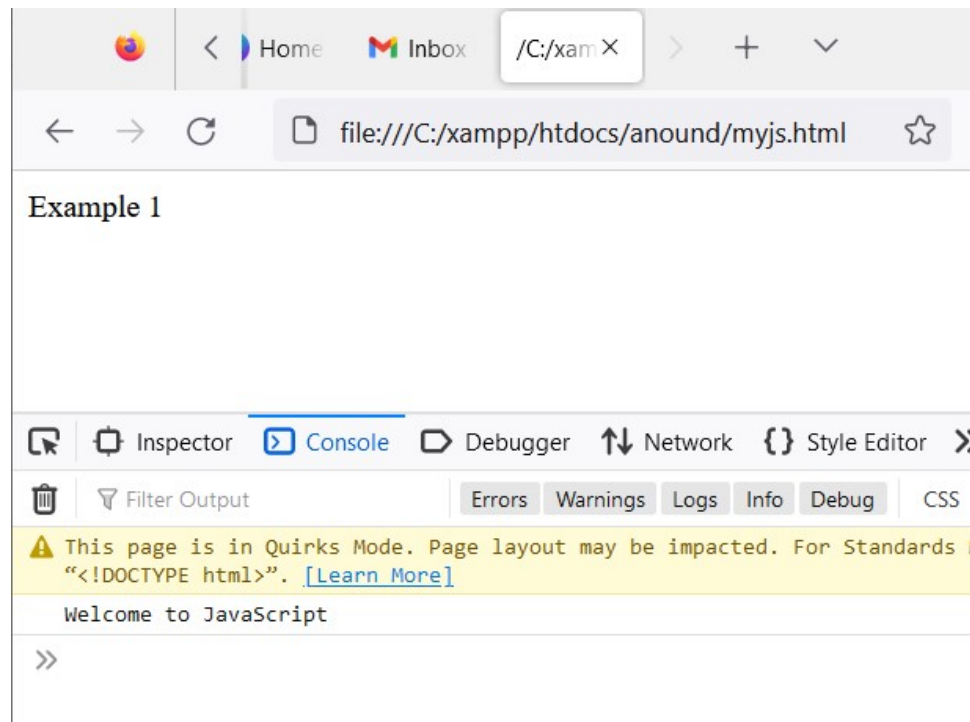


Example 1





# How to Run the JS Program



# How to Run the JS Program in Node.JS

10

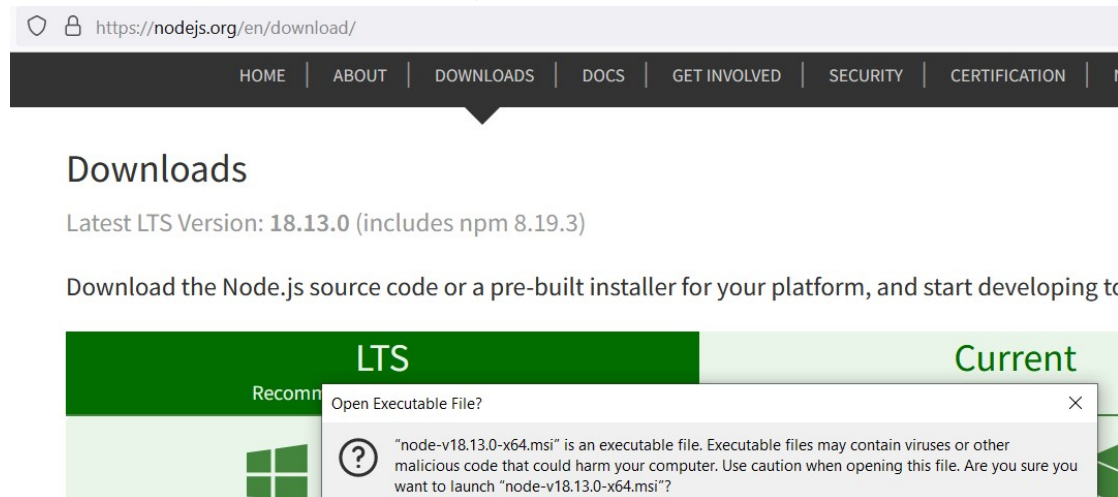
HaarisInfotech – M.H. Shoiab

Node is a back-end run-time environment for executing JavaScript code. To run JS using Node.js, follow these steps:

Install the latest version of [Download | Node.js \(nodejs.org\)](https://nodejs.org/en/download/)

<https://nodejs.org/en/download/>

Install an IDE/Text Editor like [Visual Studio Code](#). In VS code, **create a file > write JS code > save it with .js extension**

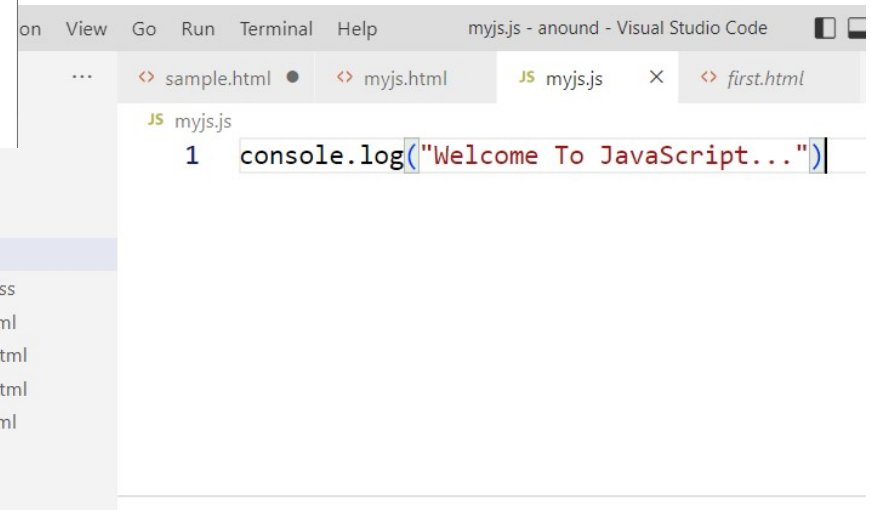
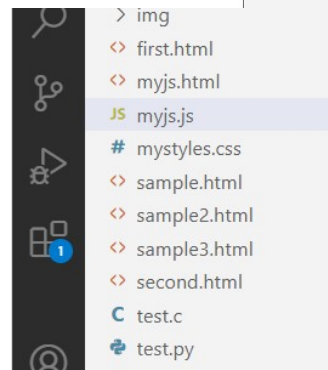
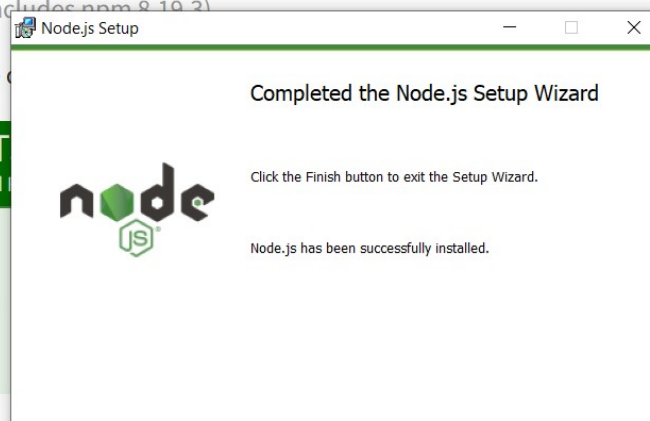
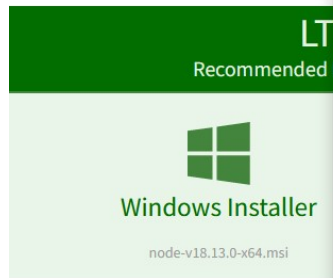


# How to Run the JS Program in Node.JS

## Downloads

Latest LTS Version: 18.13.0 (includes npm 8.19.2)

Download the Node.js source code



# How to Run the JS Program in Node.JS

12

Command Prompt

```
C:\xampp\htdocs\anound>node myjs.js  
Welcome To JavaScript...  
  
C:\xampp\htdocs\anound>
```

**Open up the terminal/command prompt > navigate to the file location > type node hello.js > hit enter.**

# How To Include .js file in a HTML

13

HaarisInfotech – M.H. Shoiab

Myjs.js

```
const number = parseInt(prompt("Enter a number: "));
```

```
// check if number is greater than 0
```

```
if (number % 2==0) {
```

```
    // the body of the if statement
```

```
    console.log("The number is Even Number..");
```

```
}
```

```
else{
```

```
    console.log("The number is Odd Number..")
```

```
}
```

Usejs.html

```
<html>
```

```
    <script src="myjs.js">
```

```
    </script>
```

```
    <script>
```

```
    </script>
```

```
</html>
```

# How To Include .js file in a HTML

14

HaarisInfotech – M.H. Shoiab

Programming in JavaScript-PART1.pptx - Microsoft P...

Design Animations Slide Show Review View Format

Slide Show Slide Master Handout Master Notes Master

Ruler Gridlines Message Bar

Zoom Fit to Window

Color Grayscale Pure Black and White

New Window Switch Windows

Macros

HAARIS Infotech Where JAVA Meets You

## How To Include .js file in a HTML

13 HaarisInfotech – M.H. Shoiab

Myjs.js

```
const number = parseInt(prompt("Enter a number: "));

// check if number is greater than 0
if (number % 2==0) {
    // the body of the if statement
```

EXPLORER

ANOUND

- img
- ajax.html
- JS demo\_worker.js
- draganddrop.html
- first.html
- geolocation.html
- index.html
- jsdemo.html
- localStorage.html
- mydata.txt
- myjquery.html
- myjs.html

JS myjs.js

```
<? usejs.html >
1 <ht
2
3
4
5
6
7
8 </h
```

file:///C:/xampp/htdocs/anound/usejs.html

file://

# INTRODUCTION

## Elements Of JavaScript Program

Elements of JavaScript Program can be divided into five categories, as follows:

- Variables
- Expressions
- Control Structures
- Functions
- Objects and Arrays

# INTRODUCTION

**Variables** are named containers that can store data.

Rules for variable names:

- Can include alphabets, digits and the underscore ( \_ )
- The first character can be either an alphabet or the underscore
- Are case-sensitive

There is no official limit on the length of variable names, but they must fit within one line.

**Expressions** are a combination of variables, constants and operators that can be evaluated to a single value.



# INTRODUCTION

**Control Structures** are structures of Java script code used to control the flow of execution based on the evaluation of certain conditional expressions.

**Function** is a group of JavaScript statements that can be referred to, using a function name and arguments.

**Object** is a type of variable that can store multiple values, called properties and functions called methods.

**Array** is a set of variables that can be referred to with the same name and a number, called an index.

# VARIABLES

Data Types

Rules for variable names

Type Casting

Variable Declaration and Scope

# VARIABLES

## JavaScript Declare Variables

In JavaScript, we use either var or let keyword to declare variables.

Both var and let are used to declare variables. However, there are some differences between them.

Var is used in older version and it is function scoped

Let is used in newer version (Es6) and it is block scoped

```
Var x=5
```

```
Let x=5
```

## JavaScript Constants

The const keyword was also introduced in the **ES6(ES2015)** version to create constants. For example,  
`const x = 5;`

# VARIABLES

## Data Types and Variables

- You can use data in two ways:
  - As a literal or constant value
  - As a variable
- The four fundamental data types used are:
  - Numbers
  - Boolean, or logical values
  - Strings
  - The null value/ the object

## Rules for variable names

- Can include alphabets, digits and the underscore (\_)
- The first character can be either an alphabet or the underscore
- Are case-sensitive
- There is no official limit on the length of variable names, but they must fit within one line.

# VARIABLES

The **undefined** data type represents **value that is not assigned**. If a variable is declared but the value is not assigned, then the value of that variable will be undefined.

For example,

```
let name;
```

```
console.log(name); // undefined
```

## JavaScript **null**

In JavaScript, null is a special value that represents **empty** or **unknown value**.

For example,

```
const number = null;
```

# VARIABLES

## •Type Casting

- A variable's type depends on the kind of information it contains
- JavaScript is loosely typed
- The type is automatically assigned depending on the literal assigned to the variable
- By the same token, the type can change depending on the operation

## •Variable Declaration and Scope

Two types of scope are possible:

- Local - variables declared inside a function
  - Global - variables declared inside a <SCRIPT> block, but not inside any functions
- You can access the contents of the Global variable anywhere inside the current web page.

# VARIABLES

23

HaarisInfotech – M.H. Shoiab

## JavaScript Implicit Conversion

In certain situations, JavaScript automatically converts one data type to another (to the right type). This is known as implicit conversion.

```
result = Number('1 23');  
console.log(result); // 123
```

```
result = Number('123e-1')  
console.log(result); // 12.3
```

```
// boolean to number  
result = Number(true);  
console.log(result); // 1
```

```
result = Number(false);  
console.log(result); // 0
```

```
let result;  
result = parseInt('10.01');  
console.log(result); // 10  
result = parseFloat('10.01');  
console.log(result); // 10.01
```

```
result = String(123);  
console.log(result); // "123"
```

# JavaScript Operators

## Types of Python Operators

Here's a list of different types of Python operators that we will learn in this tutorial.

[Arithmetic operators](#)

[Assignment Operators](#)

[Comparison Operators](#)

[Logical Operators](#)

[Bitwise Operators](#)

[Special Operators](#)



# JavaScript Operators

25

HaarisInfotech – M.H. Shoiab

## Arithmetic Operators

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo

## Javascript Assignment Operators

Operator	Name	Example
=	Assignment Operator	a = 7
+=	Addition Assignment	a += 1 # a = a
-=	Subtraction Assignment	a -= 3 # a = a
*=	Multiplication Assignment	a *= 4 # a = a
/=	Division Assignment	a /= 3 # a = a

# JavaScript Operators

26

HaarisInfotech – M.H. Shoiab

## JavaScript Comparison/Relational Operators

Operator	Description
<code>==</code>	<b>Equal to:</b> returns <code>true</code> if the operands are equal
<code>!=</code>	<b>Not equal to:</b> returns <code>true</code> if the operands are not equal
<code>===</code>	<b>Strict equal to:</b> <code>true</code> if the operands are equal and of the same type
<code>!==</code>	<b>Strict not equal to:</b> <code>true</code> if the operands are equal but of different type or not equal at all
<code>&gt;</code>	<b>Greater than:</b> <code>true</code> if left operand is greater than the right operand
<code>&gt;=</code>	<b>Greater than or equal to:</b> <code>true</code> if left operand is greater than or equal to the right operand
<code>&lt;</code>	<b>Less than:</b> <code>true</code> if the left operand is less than the right

## JavaScript Logical Operators

Operator	Description
<code>&amp;&amp;</code>	<b>Logical AND:</b> <code>true</code> if both the operands are <code>true</code> , else returns <code>false</code>
<code>  </code>	<b>Logical OR:</b> <code>true</code> if either of the operands is <code>true</code> ; returns <code>false</code> if both are <code>false</code>

# JavaScript Operators

27

HaarisInfotech – M.H. Shoiab

## JavaScript Bitwise operators

Operator	Meaning	Example
&	Bitwise AND	<code>x &amp; y = 0</code> (
	Bitwise OR	<code>x   y = 14</code> (
~	Bitwise NOT	<code>~x = -11</code> ( 1'
^	Bitwise XOR	<code>x ^ y = 14</code> (
>>	Bitwise right shift	<code>x &gt;&gt; 2 = 2</code> (

# Other Operators of JavaScript

28

HaarisInfotech – M.H. Shoiab

Operator	Description	Example
,	evaluates multiple operands and returns the value of the last operand.	<code>let a = (1,</code>
<code>?:</code>	returns value based on the condition	<code>(5 &gt; 3) ? ' 'error'; //</code>
<code>delete</code>	deletes an object's property, or an element of an array	<code>delete x</code>
<code>typeof</code>	returns a string indicating the data type	<code>typeof 3; /</code>
<code>void</code>	discards the expression's return value	<code>void(x)</code>
<code>in</code>	returns <code>true</code> if the specified property	<code>prop in obj</code>

# Some Exercises on Operators

```
let x = 6;  
let y = 5;
```

```
console.log('x + y = ', x + y);
```

```
console.log('x - y = ', x - y);
```

```
console.log('x * y = ', x * y);
```

```
console.log('x / y = ', x / y);
```

```
console.log('x % y = ', x % y);
```

```
// increment
```

```
console.log('++x = ', ++x); // x is now 7
```

```
console.log('x++ = ', x++); // prints 7 and then increased to 8
```

```
console.log('x = ', x); // 8
```

```
// decrement
```

```
X=6
```

```
console.log('--x = ', --x); // x is now 5
```

```
console.log('x-- = ', x--); // prints 5 and then decreased to 4
```

```
console.log('x = ', x); // 4
```

```
//exponentiation
```

```
console.log('x ** y = ', x ** y);
```

# Some Exercises on Operators

30

HaarisInfotech – M.H. Shoiab

```
// equal operator
console.log(2 == 2); // true
console.log(2 == '2'); // true
```

```
// not equal operator
console.log(3 != 2); // true
console.log('hello' != 'Hello'); // true
```

```
// strict equal operator
console.log(2 === 2); // true
console.log(2 === '2'); // false
```

```
// strict not equal operator
console.log(2 !== '2'); // true
console.log(2 !== 2); // false
```

```
// logical AND
console.log(true && true); // true
console.log(true && false); // false
```

```
// logical OR
console.log(true || false); // true
```

```
// logical NOT
console.log(!true); // false
```

# Statements

Conditional statements: if...else and switch

Loop Statements: for, while, do...while, break and continue

Object Manipulation Statements and Operators: for...in, new, this and with

Comments: single-line (//) and multi-line (/\*...\*/ )

# Statements

- if...else Statement

Use the if statement to perform certain statements if a logical condition is true; use the optional else clause to perform other statements if the condition is false. An if statement looks as follows:

```
if (condition) {  
    statements1  
}  
else {  
    statements2  
}
```

-The condition can be any JavaScript expression that evaluates to true or false. The statements to be executed can be any JavaScript statements, including further nested if statements. If you want to use more than one statement after an if or else statement, you must enclose the statements in curly braces: {}.



# Statements

```
const number = parseInt(prompt("Enter a number: "));
```

```
// check if number is greater than 0
if (number % 2==0) {
    // the body of the if statement
    console.log("The number is Even Number..");
}
else{
    console.log("The number is Odd Number..")
}
```

# Statements-Nested If's and else if

```
const day = prompt("Enter a day number: ");
const holiday=prompt("Enter YES or NO for
Holiday..:")
if (day == 1) {
    console.log("The day is sunday");
    if(holiday=='YES'){
        console.log("Yearly holiday wasted.....")
    }
    else{
        console.log("Normal sunday holiday...")
    }
}
```

```
else if (day == 2) {
    console.log("The day is monday");
    if(holiday=='YES'){
        console.log("Yearly holiday not
        wasted..its holiday today...")
    }
    else{
        console.log("Normal working day...")
    }
}
else {
    console.log("unknown day...");
}
```

# Statements-If's with Logical Operators

```
const day = prompt("Enter a day number: ");
const holiday=prompt("Enter YES or NO for
Holiday..:")
if (day == 1 && holiday=='YES') {
    console.log("Yearly holiday
wasted.....")
}
else if (day == 1 && holiday=='NO') {
    console.log("Yearly holiday not
wasted..its sunday today...")
}
```

```
else if (day == 2 && holiday=='YES') {
    console.log("Yearly holiday not
wasted..its holiday today...")
}
else if (day == 2 && holiday=='NO') {
    console.log("its working day today")
}
else {
    console.log("unknown day...");
}
```

# Statements-Ternary Operator

```
console.log(10%2==0)?"Even number...":"Odd Number...";
```

```
const day = prompt("Enter a day number: ");  
let holiday=prompt("Enter YES or NO for Holiday..")
```

```
s=(day==1)?(holiday=='YES')?"holiday wasted...":"normal sunday..":  
  (day==2)?(holiday=='YES')?"monday holiday":"working day"  
  : "unknown day";  
console.log(s)
```

# Statements-SwitchCase

A switch statement allows a program to evaluate an expression and attempt to match the expression's value to a case label. If a match is found, the program executes the associated statement. A 'switch' statement looks as follows:

- `switch (expression){`
- `case label :`
- `statement;`
- `break;`
- `default : statement;`
- `}`

-The program first looks for a label matching the value of expression and then executes the associated statement.

-The optional break statement associated with each case label ensures that the program breaks.

# Statements-SwitchCase

38

HaarisInfotech – M.H. Shoiab

```
const day = prompt("Enter a day number: ");
let holiday=prompt("Enter YES or NO for Holiday..")
switch(day){
  case '1':{
    switch(holiday){
      case 'YES':{
        console.log("holiday wasted....");
        break;
      }
      case 'NO':{
        console.log("Normal sunday..");
        break;
      }
    }
  }
}
```

```
case '2':{
  switch(holiday){
    case 'YES':{
      console.log("holiday utilised....");
      break;
    }
    case 'NO':{
      console.log("Normal working day..");
      break;
    }
  }
}
default:{
  console.log("unknown....") }}
```

# Statements

A for loop repeats until a specified condition evaluates to false. A 'for' statement looks as follows:

- for ([initial-expression]; [condition]; [increment-expression]) {
- statements
- }

When a for loop executes, the following occurs:

- Step 1: The initializing expression 'initial-expression', if any, is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.
- Step 2: The 'condition' expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the for loop terminates.
- Step 3: Assuming that the condition is true, the statements execute.
- Step 4: Finally, the update expression 'increment-expression' executes and control returns to step 2.

# Statements-Simple For-Loop

```
for(i=0;i<10;i++){  
    console.log(i)  
}
```

```
for(i=0;i<10;i++){  
    for(j=10;j>0;j--){  
        console.log(i+":"+j)  
    }  
}
```

```
var a = [[1,2,3],[4,5,6]];
```

```
for(i=0;i<a.length;i++){  
    for(j=0;j<a[i].length;j++){  
        console.log(a[i][j])  
    }  
}
```



# Statements

The do...while statement repeats until a specified condition evaluates to false. A 'do...while' statement looks as follows:

- do {
- statement
- } while (condition)

A while statement executes its statements as long as a specified condition evaluates to true. A 'while' statement looks as follows:

- while (condition) {
- statements
- }

# Statements

```
movieticket=false
trainticket=false
do{
    console.log("allowed to enter train....")
}while(trainticket);

while (movieticket) {
    console.log("allowed to enter..theater..");
}
```

# Functions

Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure: a set of statements that performs a specific task. To use a function, you must first define it, then your script can call it. A function definition looks as follows:

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

What are the parts that make up a function?

- The function statement.
- The name of the function, in this case sum.
- Expected parameters (arguments), in this case a and b. A function can accept
- zero or more arguments, separated by commas.
- A code block, also called the body of the function.
- The return statement. A function always returns a value. If it doesn't return
- value explicitly, it implicitly returns the value undefined.

# Functions

44

HaarisInfotech – M.H. Shoiab

Self-Invoking Function:

Used for initialization task or one time use ..... It can't be repeatedly called.

```
(  
function(variable){  
console.log('Hello ' + variable + '!');  
}  
)('shoiab')
```

# Inner Functions or Private Functions

45

HaarisInfotech – M.H. Shoiab

The benefit of using private functions are as follows:

You keep the global namespace clean (smaller chance of naming collisions).

Privacy—you expose only the functions you decide to the "outside world", keeping to yourself functionality that is not meant to be consumed by the rest of the application.

```
function outer(param) {  
  function inner(theinput) {  
    return theinput * 2;  
  }  
  return 'The result is ' + inner(param);  
}
```

```
console.log(outer(2)); //The result is 4
```

```
console.log(inner(2)); //Uncaught ReferenceError: inner is not defined
```

# Closure

46

HaarisInfotech – M.H. Shoiab

What is a Closure ?

The concept that a variable will continue to exist, and can be referenced after the function that created it has ceased executing is known as CLOSURE.

It is proved through inner functions...

The inner function can be accessed only from statements in the outer function.

## **Rule 1:**

Closures have access to the outer function's variable even after the outer function returns:  
Because its stack-reference are not destroyed like other pro. Languages

## **Rule 2:**

All global function inside function having local variable will have global scope within the closure.

# Closure

## Proof for Rule 1

```
function showName (firstName, lastName) {  
  // Local variable that ends up within closure  
  var nameIntro = "Your name is ";  
  // this inner function has access to the outer function's variables, including the parameter  
  makeFullName=function () {  
    return nameIntro + firstName + " " + lastName;  
  }  
  return makeFullName ();  
}  
res=showName("shoiab","sho")  
console.log(res)
```

# Closure – Rule 2

48

HaarisInfotech – M.H. Shoiab

```
function setupSomeGlobals() {  
    // Local variable that ends up within closure  
    var num = 666;  
    // Store some references to functions as global variables  
    gAlertNumber = function() { console.log(num); }  
    gIncreaseNumber = function() { num++; }  
    gSetNumber = function(x) { num = x; }  
}
```

```
setupSomeGlobals();  
gAlertNumber();  
gSetNumber(10);  
gAlertNumber();
```

The inner functions can be accessed by global functions.



# Objects

An object is a self-contained unit of code having the following characteristics:

- Properties
- Methods
- Identity

# window

50

HaarisInfotech – M.H. Shoiab

All global variables become properties of the window object.

```
>>>window.somevar = 1;
```

Also, all of the core JavaScript functions are methods of the window object.

```
>>>parseInt('123a456')
```

## **window.navigator:**

Used to find information on Browser and its function.

```
>>>window.navigator.userAgent //Used to find Name and version
```

of browser

```
"Mozilla/5.0 (Windows NT 6.1; rv:25.0) Gecko/20100101 Firefox/25.0"
```

## Feature Sniffing:

Its good to not to use above syntax since it hard to keep track all available browser and its version .so It's much easier to simply check if the feature you intend to use is indeed available in the user's browse.

# window

## Window.location:

The location property points to an object that contains information about the URL of the currently loaded page. There are also three methods that location provides—`reload()`, `assign()`, and `replace()`.

```
>>>window.location.href = 'http://www.packtpub.com'
```

```
>>>location.href = 'http://www.packtpub.com'
```

```
>>>location.assign('http://www.packtpub.com')
```

`replace()` is almost the same as `assign()`. The difference is that it doesn't create an entry in the browser's history list:

```
>>>location.replace('http://www.yahoo.com')
```

To reload a page you can use:

```
>>>location.reload()
```

Alternatively, you can use `location.href` to point it to itself, like so:

```
>>>window.location.href = window.location.href
```

# window

52

HaarisInfotech – M.H. Shoiab

## window.history

window.history allows limited access to the previously-visited pages in the same browser session. For example, you can see how many pages the user has visited before coming to your page:

```
>>>window.history.length  
5
```

You cannot see the actual URLs though. For privacy reasons this doesn't work: You can, however, navigate back and forth through the user's session as if the user had clicked the Back/Forward browser buttons:

```
>>>history.forward()  
>>>history.back()
```

You can also skip pages back and forth with history.go(). This is same as calling history.back():>>>history.go(-1);Two pages back:>>>history.go(-2);Reload the current page:>>>history.go(0);

# Creating Windows

53

HaarisInfotech – M.H. Shoiab

Opening new browser windows is a great feature of JavaScript

You can either load a new document (for example a HTML-document) to the new window or you can create new documents (on-the-fly)

```
window.open("www.google.com");
```

Here is a list of the properties a window can have:

directories                      yes|no

height      *number of pixels*

location    yes|no

menubar    yes|no

resizable   yes|no

scrollbars   yes|no

status      yes|no

toolbar     yes|no

width        *number of pixels*

# document

Represents characteristics of the current HTML page.

Some of its properties are:

title

- lastModified

fgColor

- bgColor

Some of its methods are:

write()

writeln()

In the browser object hierarchy, the document object is contained in a window object (for a page without frames)

# math

The Math object can't be created, since it exists automatically in all Java Script Programs  
Its properties represent mathematical constants  
Its methods are mathematical functions

Three of the most useful methods of the Math object enable you to round decimal values up and down:

- `Math.ceil()` rounds a number up to the next integer.
- `Math.floor()` rounds a number down to the next integer.
- `Math.round()` rounds a number to the nearest integer.

All these take a single parameter: the number to be rounded. You might notice one thing missing: the ability to round to a decimal place, such as for dollar amounts. You can easily simulate this, though.

# string

Any String variable in JavaScript is a String object. It has a property  
Length and  
Many Methods

String object have a single property, length.

They have a variety of methods that you can use to perform string functions.

Methods enable you to convert strings, work with pieces of strings, search for values, control the string's appearance on the HTML page, and control HTML links and anchors.

String Conversions methods:

- toUpperCase() converts all characters in the string to uppercase.

- toLowerCase() converts all characters in the string to lowercase.



# date

Is built-in JavaScript object

Create using new keyword

- The Date object is a built-in JavaScript object that enables you to conveniently work with dates and times.
- You can create a Date object any time you need to store a date, and use the Date object's methods to work with the date.
- You can create a Date object using the new keyword.
- You can use any of the following formats:

```
birthday = new Date();  
birthday = new Date("June 20, 1996 08:00:00");  
birthday = new Date(6, 20, 96);  
birthday = new Date(6, 20, 96, 8, 0, 0);
```

# date

- Setting Date Values

- A variety of set methods enable you to set components of a Date object to values:

- setDate() sets the day of the month.

- setMonth() sets the month. JavaScript numbers the months from 0 to 11, starting with January as 0.

- setYear() sets the year.

- Getting Date Values

- You can use the get methods to get values from a Date object. This is the only way to obtain these values, because they are not available as properties:

- getDate() gets the day of the month.

- getMonth() gets the month.

- getYear() gets the year.

- getHours(), getMinutes() and getSeconds() get the time.

# Arrays

59

HaarisInfotech – M.H. Shoiab

JavaScript doesn't support array variables

Arrays need to be created using array object

- Once you define the array, you can access its elements by using brackets to indicate the index

- As an example, this statement creates an array called scores with 20 values:

- ```
Scores = new Array(20);
```

- Array object methods:

- `join()` quickly joins all the array's elements, resulting in a string. The elements are separated by commas by default.
  - `reverse()` returns a reversed version of the array: the last element becomes the first, and the first element becomes the last.
  - `sort()` returns a sorted version of the array.

# For Loop Array Object

60

HaarisInfotech – M.H. Shoiab

```
// Create one dimensional array
var twod = new Array(2);
console.log("Creating 2D array <br>");
// Loop to create 2D array using 1D array
for (var i = 0; i < twod.length; i++) {
    twod[i] = new Array(2);
}
var h = 0;
// Loop to initialize 2D array elements.
for (var i = 0; i < 2; i++) {
    for (var j = 0; j < 2; j++) {
        twod[i][j] = h++;
    }
}
```

```
// Loop to display the elements of 2D
array.
for (var i = 0; i < 2; i++) {
    for (var j = 0; j < 2; j++) {
        console.log(twod[i][j] + " ");
    }
    console.log()
}
```

# Looping through object properties

61

HaarisInfotech – M.H. Shoiab

The final statement used for object-oriented work in JavaScript is the for...in loop

This loop executes once for each property of an object, assigning the index variable to the property name

For example, you could make a simple function to list the properties of an object and their values

```
function list(object) {  
    for (var i in object) {  
        document.write("Property: ",i," Value: ",object[i], "<BR>");  
    }  
}
```

# Defining Custom Objects

62

HaarisInfotech – M.H. Shoiab

## Using Object Constructor

```
var person = new Object();  
person.name = "Diego";  
person.getName = function(){  
    return this.name;  
};
```

```
console.log(person.getName())
```

# Defining Custom Objects

63

HaarisInfotech – M.H. Shoiab

## Using Object Constructor and Literal notation

```
var person = new Object();  
person.name = "shoiab";  
person.getName = function(){  
    return this.name;  
};
```

```
console.log(person.getName())
```

```
var person = {  
    name:"shoiab",  
    getName:function(){  
        return this.name;  
    }  
}
```

```
console.log(person.getName())
```

# Defining Custom Objects

64

HaarisInfotech – M.H. Shoiab

## Using Factory Function

```
var Person=function(name){  
    var result = new Object();  
    result.name = name;  
    result.getName = function(){  
        return this.name;  
    };  
    return result;  
};  
var personOne = Person("shoiab");  
var personTwo =Person("habib");
```

## Another Way

```
var Person=function(name){  
    return {  
        name : name,  
        getName : function(){  
            return this.name;  
        }  
    };  
};
```

```
console.log(personOne.getName()); // prints shoiab  
console.log(personTwo.getName()); // prints habib()
```



# Defining Custom Objects

65

HaarisInfotech – M.H. Shoiab

The object definition is a simple function that accepts parameters to initialize a new object and assigns those to the corresponding properties.

```
function display()
{
    document.write("Name : "+ this.name + "< BR>")
    document.write("Address : "+ this.address + "< BR>")
}
function person(name,address)
{
    this.name = name
    this.address = address
    this.display = display
}
```

```
// creating instances
Ravi = new person("Ravi","Bangalore");
Mahesh = new person("Mahesh","Madras");
// print them
Ravi.display()    Mahesh.display()
```

# Constructor Property of Object

The constructor property contains a reference to a function, you might as well call this function to produce a new object. The following code is like saying, "I don't care how object h2 was created, but I want another one just like it".

```
function Hero(name)
{
    this.name = name;
}
var h2 = new Hero('NewCollege');
//create a deep copy of the object using constructor property
//without knowing the original function name
var h3=new h2.constructor("Dhaanish")
console.log(h3.constructor)
console.log(h3 instanceof Hero)
```

# Prototype Property of Object

67

HaarisInfotech – M.H. Shoiab

You can add **Methods and Properties Using the Prototype**

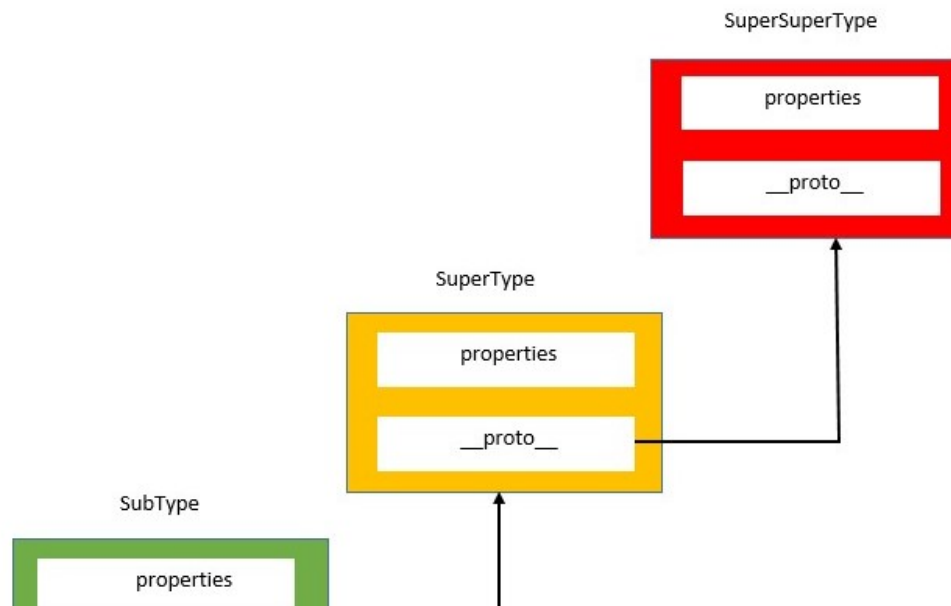
```
function Gadget(name, color) {  
  this.name = name;  
  this.color = color;  
  this.whatAreYou = function(){    return 'I am a ' + this.color + ' ' + this.name;    } }  
  
Gadget.prototype.price = 100;  
Gadget.prototype.rating = 3;  
Gadget.prototype.getInfo = function() {  
  return 'Rating: ' + this.rating + ', price: ' + this.price + ', name..:' + this.name + ', color..:' + this.color; };  
  
g=new Gadget("mobile","black")  
console.log(g.getInfo())
```

# Inheritance

68

HaarisInfotech – M.H. Shoiab

Prototype chaining is the default way to implement inheritance in JS.



# Inheritance

69

HaarisInfotech – M.H. Shoiab

```
function Shape(){
  this.name = 'shape';
  this.toString = function() {
    return this.name;}; }

function TwoDShape(){
  this.twodname = '2D shape'; }

function Triangle(side, height) {
  this.itemname = 'Triangle';
  this.side = side;
  this.height = height;
  this.getArea = function(){
    return this.side * this.height / 2;}; }

//The code that performs the inheritance magic is as follows:
TwoDShape.prototype = new Shape();
//inheriting shape to TwoDShape
TwoDShape.prototype.constructor = TwoDShape;
Triangle.prototype = new TwoDShape();
//inheriting TwoDShape to Triangle
Triangle.prototype.constructor = Triangle;

var res = new Triangle(2,8);
console.log(res.name+"...."+res.twodname+"...."+res.itemname);
console.log(res.getArea());
```

# Dynamic Object Creation

70

HaarisInfotech – M.H. Shoiab

//The code that performs the inheritance magic is as follows:

```
TwoDShape.prototype = new Shape();
```

```
//inheriting shape to TwoDShape
```

```
TwoDShape.prototype.constructor = TwoDShape;
```

```
Triangle.prototype = new TwoDShape();
```

```
//inheriting TwoDShape to Triangle
```

```
Triangle.prototype.constructor = Triangle;
```

```
var res = new Triangle(2,8);
```

```
console.log(res.name+"...."+res.twodname+"...."+res.itemname); console.log(res.getArea());
```

# Dynamic Dependency Injection and JSON

```
<html>
<head>
<script language="JavaScript">
myjson='{ "shoeshop": "RahimShop", "shoefactory": "BataShoeFactory" }';
function LeatherShoe(){ this.name="this is leather shoe"; }
function SportsShoe(){ this.name="this is sports shoe..."; }
function Shoe(){ this.getName=function(){ return this.name; }}
function ShoeFactory(name){ this.name=name; this.getName=function(){ return this.name; }}
function BataShoeFactory(){ this.makeShoe=function(){ return new SportsShoe(); }}
function LakhaniShoeFactory(){ this.makeShoe=function(){ return new LeatherShoe(); }
}
shoe=new Shoe(); LeatherShoe.prototype=shoe; SportsShoe.prototype=shoe;
contd....
```

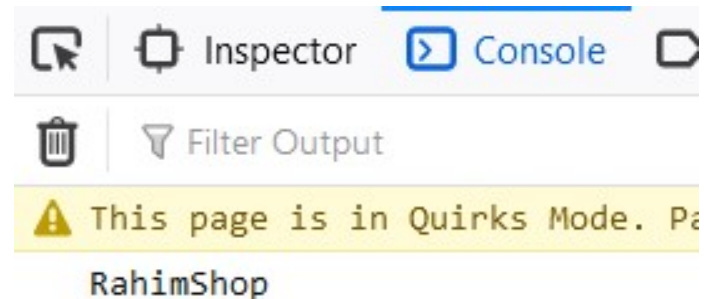
# Dynamic Dependency Injection and JSON

72

HaarisInfotech – M.H. Shoiab

```
ss=new ShoeShop(); RahimShop.prototype=ss; sf=new ShoeFactory();  
BataShoeFactory.prototype=sf; LakhaniShoeFactory.prototype=sf;  
function ShoeShop(){ this.setFactory=function(factory){ this.factory=factory; }  
  this.getFactory=function(){ return this.factory; }}  
function RahimShop(){ this.setShop=function(shop){ this.shop=shop; }  
  this.sellShoe=function(){ return this.getFactory().makeShoe(); }}
```

```
x=JSON.parse(myjson);  
shop=new window[x.shoeshop]();  
factory=new window[x.shoefactory]("factory")  
shop.setFactory(factory);  
console.log(shop.sellShoe().getName());  
</script></head><body></body></html>
```





# Events

Are things that happen to the browser

Used to trigger portions of program

Pertain to the web page containing the script

- When the user clicks on a link, selects or enters text, or even moves the mouse over part of the page, an event occurs.
- You can use JavaScript to respond to these events. For example, you can have custom messages displayed in the status line (or somewhere else on the page) as the user moves the mouse over links. You can also update fields in a form whenever another field changes.

# Events

-onBlur	Occurs when an object on the page loses focus
onChange	Occurs when a text field is changed by the user
onClick	Occurs when the user clicks on an item
onFocus	Occurs when an item gains focus
onLoad	Occurs when the page (or an image) finishes loading
onMouseOver	Occurs when the mouse pointer moves over an item
onMouseOut	Occurs when the mouse pointer moves off an item
onSelect	Occurs when the user selects text in a text area
onSubmit	Occurs when a submit button is pressed
onUnload	Occurs when the user leaves the document or exits

# Event Handling

75

HaarisInfotech – M.H. Shoiab

- Are embedded in html tags, typically as part of forms, but are also included as a part of some anchors and links
- Virtually anything a user can do to interact with a page is covered with the event handlers, from moving the mouse to leaving the current page

```
<SCRIPT LANGUAGE="JavaScript">  
    function hello()  
    {  
        window.alert("You clicked the heading");  
    }  
</SCRIPT>  
<P> Click <A onClick = "hello();" HREF = ""> here </A> you see a message  
</P>
```

# Timeout

76

HaarisInfotech – M.H. Shoiab

Statements that will be executed after a certain amount of time elapses

Handy for periodically updating a Web Page or for delaying the display of a message or execution of a function

`window.setTimeout()`

- You begin a timeout with the `setTimeout()` Method.
- Before a timeout has elapsed, you can stop it with the `clearTimeout()` method, specifying the identifier of the timeout to stop.
- For a continuous call use
- `window.setInterval()`

# Timeout

77

HaarisInfotech – M.H. Shoiab

```
// Update a page every 2 seconds
    var counter = 0;
    // call Update function in 2 seconds after first load
    ID=window.setTimeout("Update();",2000);
    function Update(){
counter++;
window.status="The counter is now at "+counter;
document.form1.input1.value="The counter is now at "+counter;
//set another timeout for the next count
ID=window.setTimeout("Update();",2000);}
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input1" SIZE="40"><BR>
<INPUT TYPE="button" VALUE="RESET" onClick="counter=0;">
<BR><INPUT TYPE="button" VALUE="STOP" onClick="window.clearTimeout(ID);"></form>
```

# LocalStorage

**LocalStorage** is a data storage type of web storage. This allows the JavaScript sites and apps to store and access the data without any expiration date. This means that the data will always be persisted and will not expire. So, data stored in the browser will be available even after closing the browser window.

In short, all we can say is that the localStorage holds the data with no expiry date, which is available to the user even after closing the browser window. It is useful in various ways, such as remembering the shopping cart data or user login on any website.

In the past days, cookies were the only option to remember this type of temporary and local information, but now we have localStorage as well. Local storage comes with a higher storage limit than cookies (5MB vs 4MB). It also does not get sent with every [HTTP](#) request. So, it is a better choice now for client-side storage.

# LocalStorage

79

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE html><html><body>
<div id="result"></div>
<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
    // Store
    localStorage.setItem("lastname", "new college");
    // Retrieve
    document.getElementById("result").innerHTML = localStorage.getItem("lastname");
} else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support Web
Storage...";
}
</script></body></html>
```

# Session Storage

## Window.sessionStorage

The read-only **sessionStorage** property accesses a session Storage object for the current origin. `sessionStorage` is similar to localStorage; the difference is that while data in `localStorage` doesn't expire, data in `sessionStorage` is cleared when the *page session* ends.



# Session Storage

81

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE html><html><body><div id="result"></div>
<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
// Store
sessionStorage.setItem("myname","shoiab");
// Retrieve
document.getElementById("result").innerHTML = sessionStorage.getItem("myname");
} else {
document.getElementById("result").innerHTML = "Sorry, your browser does not support Web
Storage...";
}
</script></body></html>
```

# Web Worker

## What are web workers in JavaScript?

A web worker is a piece of browser functionality. It is the real OS threads that can be spawned in the background of your current page so that it can perform complex and resource-intensive tasks.

Imagine that you have some large data to fetch from the server, or some complex rendering needs to be done on the UI. If you do this directly on your webpage then the page might get jankier and will impact the UI.

To mitigate this, you can simply create a thread – that is a web worker – and let the web worker take care of the complex stuff.

# Web Worker

83

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE html><html><body>
<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support Web
Workers.</p>
<script>
var w;
function startWorker() {
    if(typeof(Worker) !== "undefined") {
        if(typeof(w) == "undefined") {
            w = new Worker("demo_worker.js");
        }
        contd...
```

# Web Worker

84

HaarisInfotech – M.H. Shoiab

```
w.onmessage = function(event) {  
    document.getElementById("result").innerHTML = event.data;  
};  
} else {  
    document.getElementById("result").innerHTML = "Sorry, your browser does not support Web  
Workers...";  
}  
}  
function stopWorker() {  
    w.terminate();  
    w = undefined;  
}  
</script></body></html>
```

Demo\_worker.js

```
var i = 0;  
function timedCount() {  
    i = i + 1;  
    postMessage(i);  
    setTimeout("timedCount()",500);  
}timedCount();
```

# Drag and Drop

Drag and Drop is a very interactive and user-friendly concept that makes it easier to move an object to a different location by grabbing it. This allows the user to click and hold the mouse button over an element, drag it to another location, and release the mouse button to drop the element there. In HTML 5 Drag and Drop are much easier to code and any element in it is draggable.

**Drag and Drop Events:** There are various Drag and Drop events, some of them are listed below:

**ondrag:** It is used to use when the element or text selection is being dragged in HTML.

**ondragstart:** It is used to call a function, drag(event), that specifies what data to be dragged.

**ondragenter:** It is used to determine whether or not the drop target is to accept the drop. If the drop is to be accepted, then this event has to be canceled.

**ondragleave:** It occurs when the mouse leaves an element before a valid drop target while the drag is occurring.

**ondragover:** It specifies where the dragged data can be dropped.

**ondrop:** It specifies where the drop has occurred at the end of the drag operation.

**ondragend:** It occurs when the user has finished dragging an element.

# Drag and Drop

86

HaarisInfotech – M.H. Shoiab

## Step 1: Make an object draggable

First set the draggable attribute to true for that element to be draggable `<img draggable = "true">`

Then, specify what should happen when the element is dragged. The *ondragstart* attribute calls a function, `drag(event)`, that specifies what data to be dragged. The `dataTransfer.setData()` method sets the data type and the value of the dragged data. The event listener *ondragstart* will set the allowed effects (copy, move, link, or some combination).

## Step 2: Dropping the Object

The *ondragover* event specifies where the dragged data can be dropped. By default, data/elements cannot be dropped in other elements. To allow a drop, it must prevent the default handling of the element. This is done by calling the `event.preventDefault()` method. Finally, the drop event, which allows the actual drop to be performed.

# Drag and Drop

87

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE HTML><html><head><style>
#div1 {width:350px;height:70px;padding:10px;border:1px solid #aaaaaa;}
</style><script>
function allowDrop(ev) {
    ev.preventDefault();
}
function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}
function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
} contd...
```

# Drag and Drop

88

HaarisInfotech – M.H. Shoiab

```
</script></head><body>  
<p>Drag the image image into the rectangle:</p>  
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>  
<br>  
  
</body>  
</html>
```



# GeoLocation

## HTML5 Geolocation

The Geolocation is one of the best HTML5 API which is used to identify the user's geographic location for the web application.

This new feature of HTML5 allows you to navigate the latitude and longitude coordinates of the current website's visitor. These coordinates can be captured by JavaScript and send to the server which can show your current location on the website

Most of the geolocation services use Network routing addresses such as IP addresses, RFID, WIFI and MAC addresses or internal GPS devices to identify the user's location.

# GeoLocation

90

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE html><html><body>
<p>Click the button to get your coordinates.</p>
<button onclick="getLocation()">Try It</button>
<p id="demo"></p>
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showP
osition);
    } else {
        x.innerHTML = "Geolocation is not supported
by this browser.";
    }
}
```

```
function showPosition(position) {
    x.innerHTML = "Latitude: "
        + position.coords.latitude +
        "<br>Longitude: " +
        position.coords.longitude;
}
</script>

</body>
</html>
```

# JQuery

What is jQuery

jQuery is a small and lightweight JavaScript library.

jQuery is cross-platform.

jQuery means "write less do more".

jQuery simplifies AJAX call and DOM manipulation.

# JQuery

92

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE html>
<html>
<head>
  <title>First jQuery Example</title>
  <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
  </script>
  <script type="text/javascript" language="javascript">
$(document).ready(function() {
  $("p").css("background-color", "blue");
});
  </script>
</head>
<body>  <p>This is first paragraph.</p>  </body>  </html>
```

# Jquery-Event

93

HaarisInfotech – M.H. Shoiab

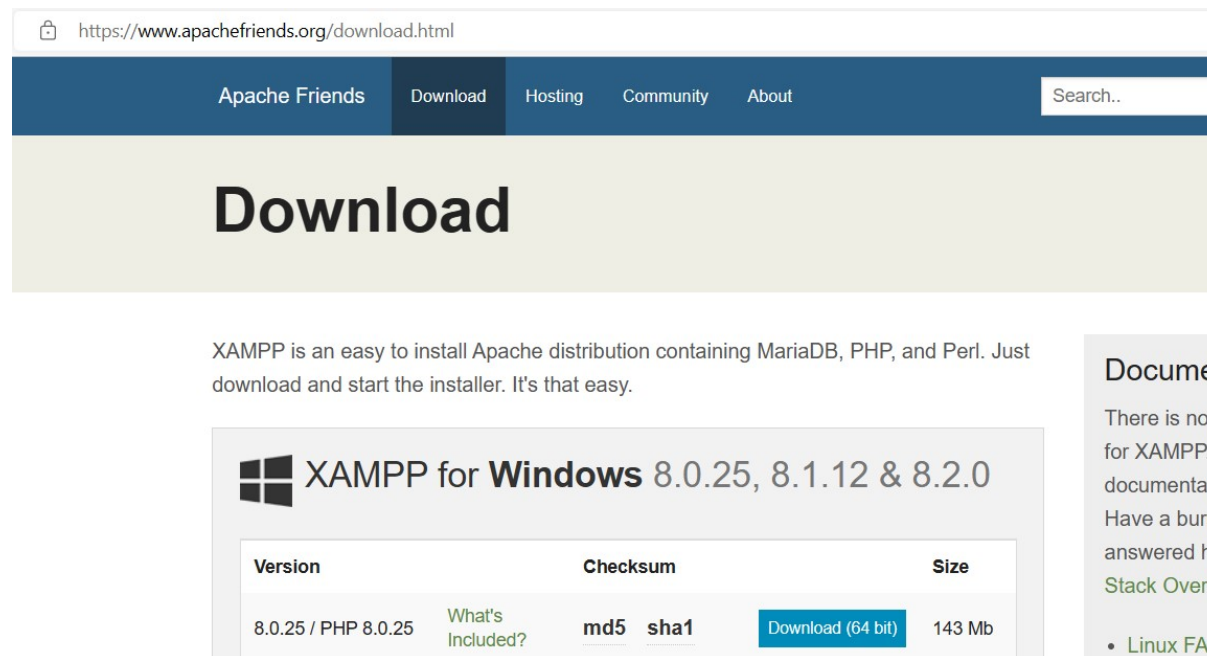
```
<!DOCTYPE html> <html> <head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").click(function(){
        alert("This paragraph was clicked.");
    });
});
</script>
</head>
<body>
<p>Click on the statement.</p>
</body>
</html>
```

# Server Side Scripting

94

HaarisInfotech – M.H. Shoiab

Before we start let us first download XAMPP HTTP Server and install the same.



The screenshot shows the Apache Friends website's download page. The URL in the browser is <https://www.apachefriends.org/download.html>. The navigation bar includes links for Apache Friends, Download, Hosting, Community, and About, along with a search bar. The main heading is "Download". Below this, a text block states: "XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy." The XAMPP for Windows section highlights versions 8.0.25, 8.1.12, and 8.2.0. A table lists the version, checksum, and size for the 8.0.25 / PHP 8.0.25 package, with a download button for the 64-bit version (143 Mb). A sidebar on the right contains a "Documentation" section with links to "Stack Overflow" and "Linux FAQ".

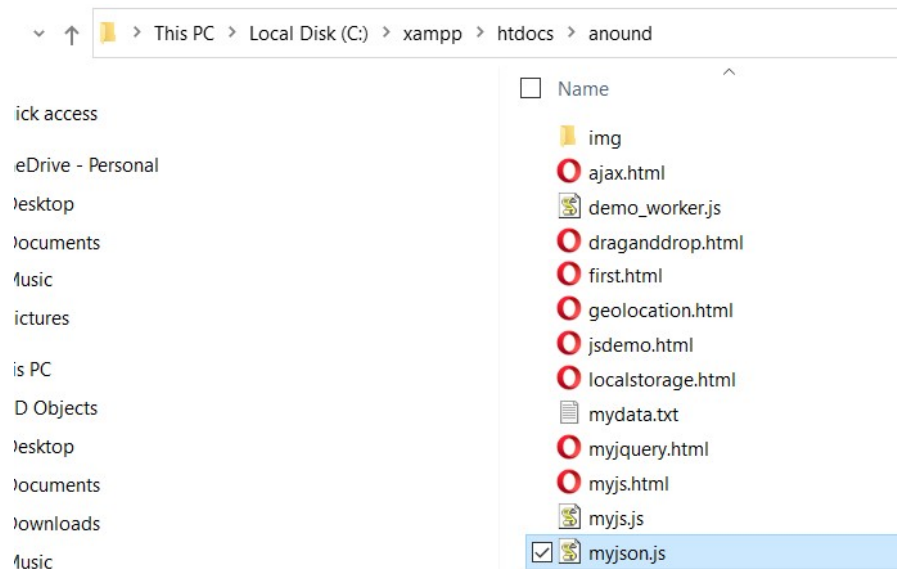
Version	Checksum	Size
8.0.25 / PHP 8.0.25	md5 sha1	143 Mb

<https://www.apachefriends.org/download.html>

# Server Side Scripting

95

HaarisInfotech – M.H. Shoiab



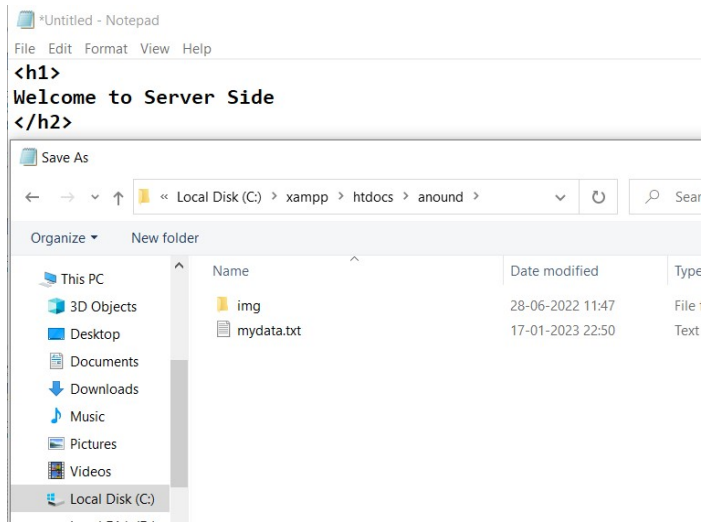
After installation, go to the folder where xampp is installed, then select htdocs folder.

You can now create a folder of your choice inside the htdocs folder and use that as your base folder, where you will keep all your web related programs.

# Server Side Scripting

96

HaarisInfotech – M.H. Shoiab



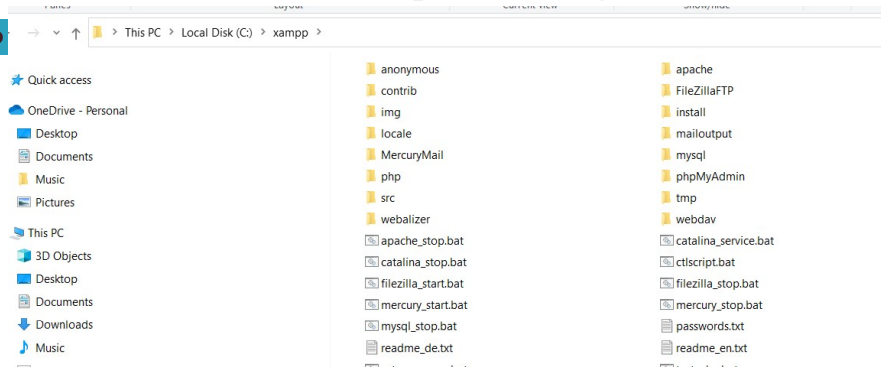
Now create a index.html file and place it in the folder you have created.



# Server Side Scripting

97

HaarisInfotech – M.H. Shoib



And then go to the xampp folder and double click xampp\_start.exe to start the server.

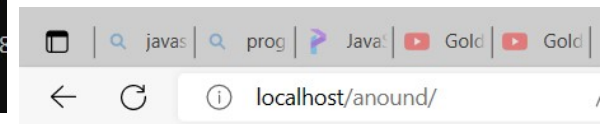
C:\xampp\xampp\_start.exe

XAMPP now starts as a console application.

Instead of pressing Control-C in this console window, please use xampp\_stop.exe to stop XAMPP, because it lets XAMPP end any current transactions and cleanup gracefully.

```
2023-01-17 22:50:30 0 [Note] Using unique option prefix 'key_buffer' is error-prone and can break
use the full name 'key_buffer_size' instead.
2023-01-17 22:50:30 0 [Note] mysql\bin\mysqld.exe (mysqld 10.4.21-MariaDB) starting as process 168
```

And then open a browser and type the url –  
<http://localhost/yourdirectoryname>



**Welcome to Server Side**

# Ajax – Server Side

**What is AJAX ?**

**Asynchronous Java Script and XML**

**With AJAX web applications finally start feeling like desktop applications**

**AJAX enables your web application to work behind the scenes, getting data as you need and displaying that data as you want.**

**When this happens, the browser doesn't get refreshed**

# Ajax – Server Side

99

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<script type="text/javascript">
    var XMLHttpRequestObject = false;
    if(window.XMLHttpRequest)
    {
        XMLHttpRequestObject =new XMLHttpRequest();
    }
    contd..
```

# Ajax – Server Side

100

HaarisInfotech – M.H. Shoiab

```
else if (window.ActiveXObject)
{
    XMLHttpRequestObject =
    new ActiveXObject("Microsoft.XMLHTTP");
}
function getData(dataSource, divID)
{
    if(XMLHttpRequestObject)
    {
        var obj=document.getElementById(divID);
        XMLHttpRequestObject.open("GET",dataSource);
```

Contd..

# Ajax – Server Side

101

HaarisInfotech – M.H. Shoiab

```
XMLHttpRequestObject.onreadystatechange=function()
{
    if(XMLHttpRequestObject.readyState==4 &&
XMLHttpRequestObject.status ==200)
    {
        var response=
XMLHttpRequestObject.responseText;
obj.innerHTML=response;
    }
}
XMLHttpRequestObject.send(null);
}
} contd..
```

# Ajax – Server Side

102

HaarisInfotech – M.H. Shoiab

```
</script>
```

```
<form>
```

```
    <input type="button" value="Display message"
    onclick=
    "getData('http://localhost/mydata.txt','targetDiv')">
```

```
<div id="targetDiv" style="color:red">
```

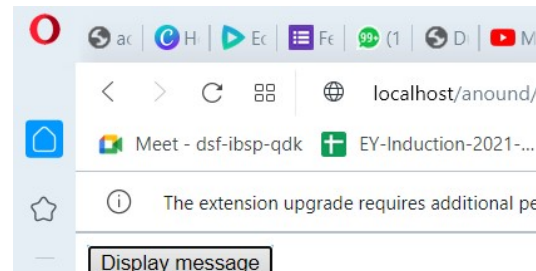
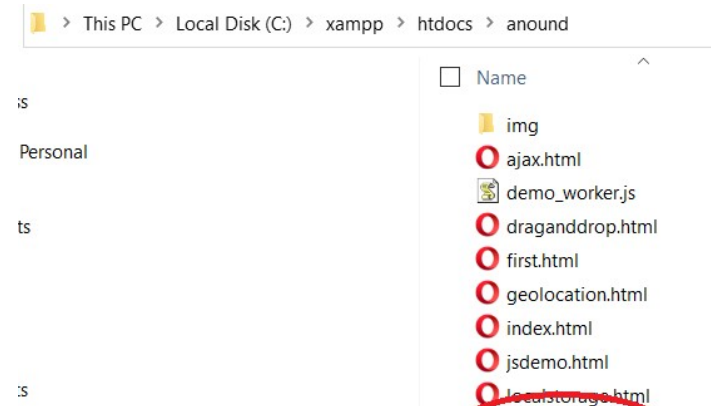
```
    <p>The fetched data will go here</p>
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```



# Jquery-Ajax- GET Client

103

HaarisInfotech – M.H. Shoiab

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script type="text/javascript"
src="http://cdnjs.cloudflare.com/ajax/libs/json2/20130526/json2.min.js"></script>
```

Contd..

# Jquery-Ajax

104

HaarisInfotech – M.H. Shoiab

```
<script>
```

```
$.ajax('http://localhost/anound/myjson.js',  
{
```

```
  dataType: 'json', // type of response data
```

```
  timeout: 500, // timeout milliseconds
```

```
  success: function (data,status,xhr) { // success callback function
```

```
    $('p').append(data[0].id + ' ' + data[0].name );
```

```
    let output = data.map(i => "<tr><td>" + i.id + "</td><td>" + i.name +  
"</td></tr>");
```

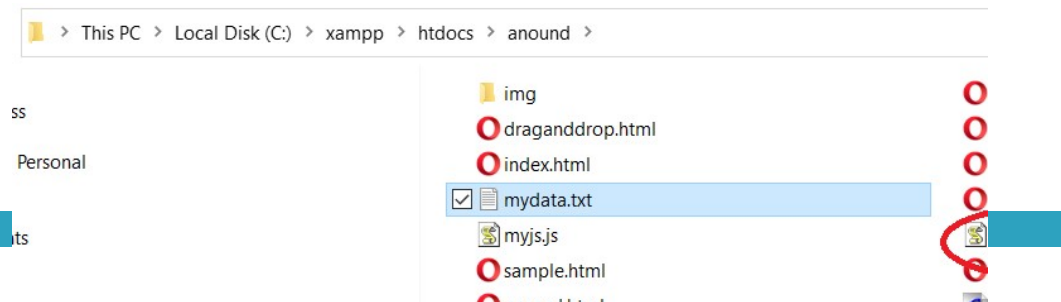
```
    $("#output").html(output);
```

```
    $("#mytable").show();
```

```
  },
```

```
  error: function (jqXHR, textStatus, errorMessage) { // error callback
```

```
    $('p').append('Error: ' + errorMessage);    } });
```





# Jquery-Ajax

105

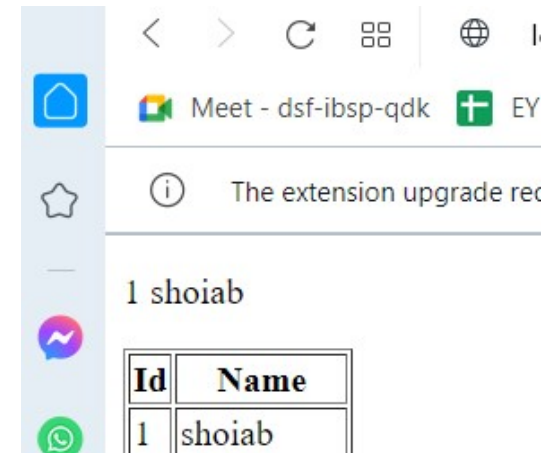
HaarisInfotech – M.H. Shoiab

```
</script>
<p></p>
<table style="display:none" id="mytable" class="table table-bordered" border="1">
<tr class="table-head">
<th>Id</th>
<th> Name</th>
</tr>
<tbody id="output">
</tbody>
</table>
</body>
</html>
```

myjson.js - Notepad

File Edit Format View Help

```
{"id": "RahimShop", "name": "BataShoeFac"}
```



# The Beginning....of Part2

106

HaarisInfotech – M.H. Shoiab



**Thankyou**

**All The Best**

**M.H. Shoiab**  
**9840135749**

