



## **MODULE-5**

# **HANDLING EVENTS WITH JAVASCRIPT**

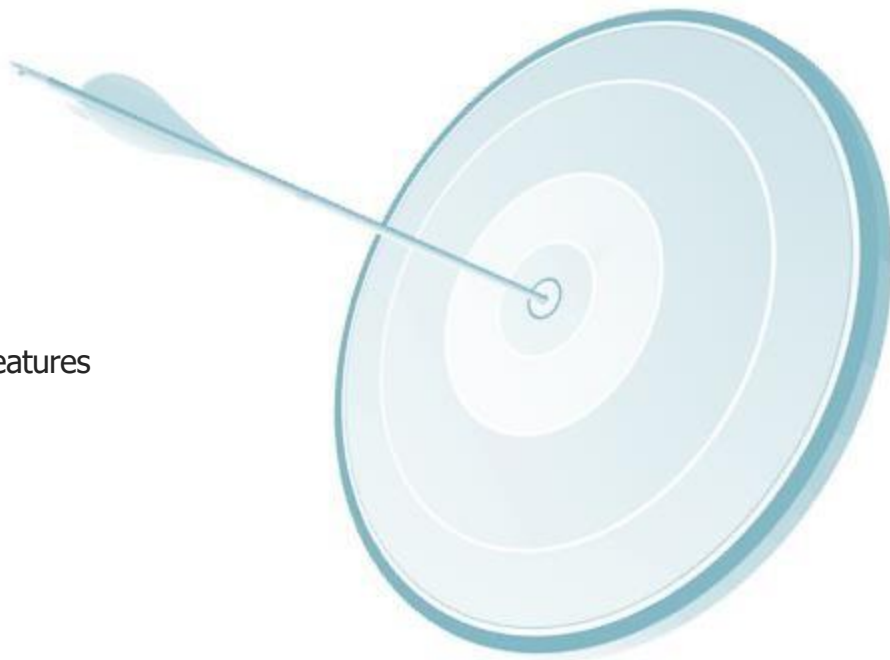
# Course Topics

- [Module 1](#)
  - » Deploying the first Website to Amazon S3
- [Module 2](#)
  - » Creating web pages with HTML5
- [Module 3](#)
  - » Styling web pages using CSS
- [Module 4](#)
  - » CSS3 effects and animations
- [Module 5](#)
  - » **Handling events with JavaScript**
- [Module 6](#)
  - » Twitter Bootstrap 3
- [Module 7](#)
  - » Twitter Bootstrap 3 Project
- [Module 8](#)
  - » Bootstrap ScrollSpy, jQuery and jQuery UI
- [Module 9](#)
  - » Ajax, Google APIs, Social Plugins
- [Module 10](#)
  - » Project - Building Website Tour

# Objectives

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

- Learn how to write code in JavaScript
- Know features of JavaScript
- Define different data types
- Handle events in JavaScript
- Manipulate DOM elements using JavaScript
- Understand JavaScript Variable Hoisting and Closure features



# Introduction to JavaScript

- Before getting into the insight of JavaScript, lets learn why should we learn JavaScript when a website can run even without enabling JavaScript?
- JavaScript is a **client-side programming language** that validates everything before sending requests to the server and hence, it helps in providing immediate feedback to the user and improves the performance of website



# Introduction to JavaScript

- Almost all the websites today use JavaScript in their webpages
- Using JavaScript the webpages become more interactive. You can use the JS to create menu, validate forms or swap two images
- JavaScript is a dynamic computer programming language used to handle the client side scripting associated with the browser
- JavaScript allows you to do certain things in the user's browser without sending messages back and forth to the server



It is not expected to know Java programming language to understand JavaScript

# Features of JavaScript w.r.t HTML and CSS

- JavaScript can modify the HTML data elements using [innerHTML property](#)
- It can also modify the HTML style elements
- One of the major use of JavaScript is to [validate the forms before sending data to the server](#)
- Similarly, without JavaScript , animations with round trips to the server is unlikely to work and it would be very slow.

# JavaScript Features

---

→ JavaScript features are same as most of the common programming languages. It supports:

- » Variables
- » Arrays
- » Objects
- » Operators
- » Functions
- » Conditional statements
- » Loops
- » Comments
- » Event handlers

# JavaScript – Variables

- JavaScript variables are the containers to store values. They are declared using the “var” keyword
- JavaScript code is written in between `<script>` `</script>` tags
- `document.getElementById(“num”)` will get the element with id “num” and its content is replaced by the value of variable “balance” using `innerHTML` property

## Example

```
<!DOCTYPE html>
<html>
<body>
<p id="num"/>
<p id="str"/>
<script>
var balance = 3000.54;
var person = "Jack";
document.getElementById("num").innerHTML = balance;
document.getElementById("str").innerHTML = person;
</script>
</body>
</html>
```

## Output



- Float data type is assigned
- String data type is assigned to the variable person

Previously type attribute was used : `<script type="text/javascript">`. But type attribute is not required. JavaScript is the default scripting language in HTML.




# JavaScript – Data Types

→ Below are the most commonly used JavaScript data types:

- » Number
- » Boolean
- » Null
- » Undefined
- » String
- » Object

```
var length = 16;  
var cost= 105.50  
var x = {firstName:"Andrew", lastName:"Erlchson"};  
var a = true;  
var lastName = "aString";  
var cars = ["India", "US", "UK"];
```



Previously type attribute was used : `<script type="text/javascript">`. But type attribute is not required. JavaScript is the default scripting language in HTML

```
// Number  
// Number  
// Object  
// Boolean  
// String  
// Array
```

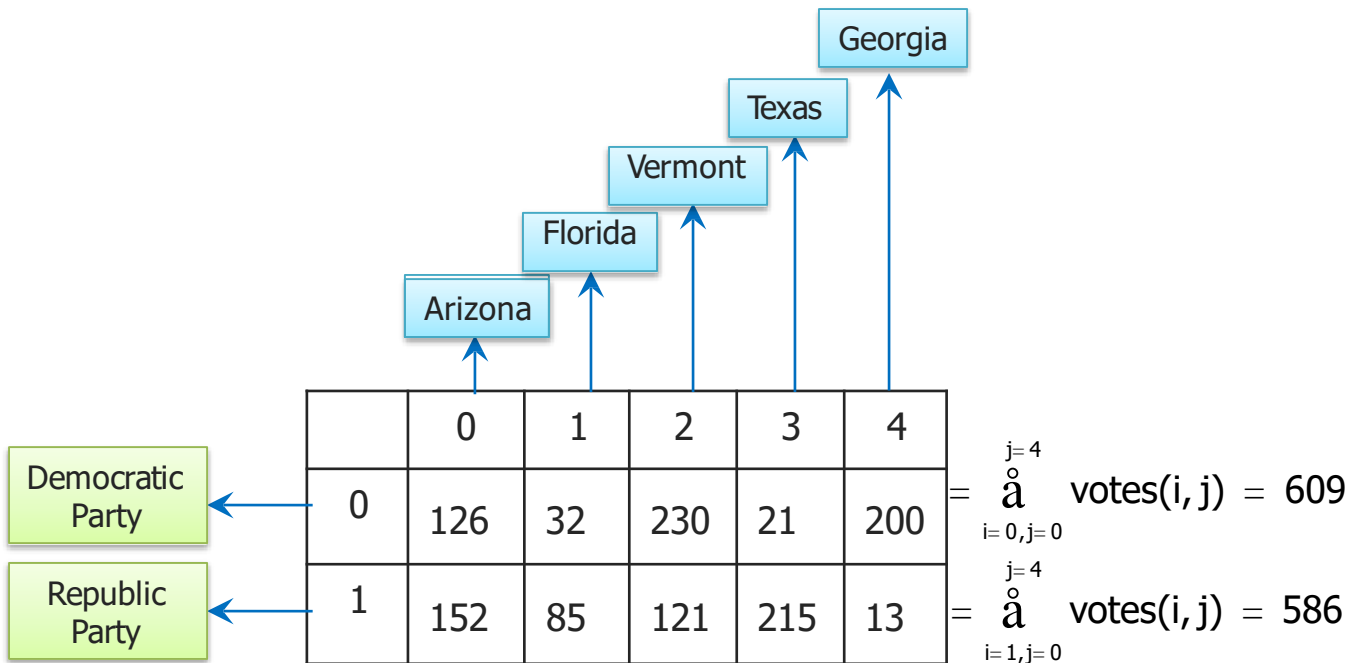
# Problem Statement

→ Given below is the data on the votes for different parties, collected during elections in US

Serial no.	Region	Democratic Party	Republican Party
1	Arizona	126	152
2	Florida	32	85
3	Vermont	230	121
4	Texas	21	215
5	Georgia	200	13

→ The data is huge and given for many regions. We can use JavaScript arrays to represent this data in an array format

# Solution – JavaScript Arrays



# JavaScript – Arrays

→ An **array** is a **sequence of homogeneous data types**. When you want to **store multiple values of same type**, **arrays can be used**

→ An array named "**cities**" is declared and the values "New York", "San Francisco" and "New Jersey" is assigned

→ The values stored in the array are as follows:

» cities[0] = New York, cities[1] = San Francisco, cities[2] = New Jersey

## Example

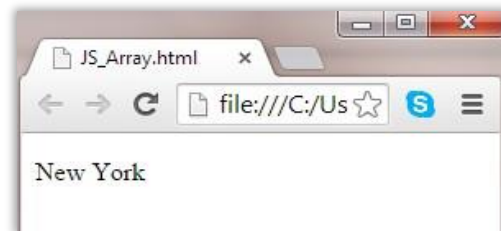
```
<script>
var cities = ["New York", "San Francisco", "New Jersey"];
document.getElementById("city").innerHTML = cities[0];
</script>
```

→ Here the value stored in cities[0] will be displayed

```
<script>
var cities = ["New York", "San Francisco", "New Jersey"];
document.getElementById("city").innerHTML = cities[2];
</script>
```

→ Here the value stored in cities[2] will be displayed

## Output



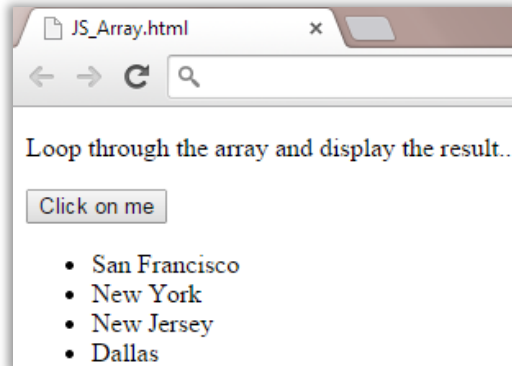
# JavaScript – Looping through Arrays

→ Now consider the example shown below:

## Example

```
<!DOCTYPE html>
<html>
<body>
<p>Loop through the array and display the result...</p>
<button onclick="ArrayDisplay()">Click on me</button>
<p id="ArrDispl"></p>
<script>
function ArrayDisplay() {
    var index;
    var text = "<ul>";
    var cities = ["San Francisco", "New York", "New Jersey", "Dallas"];
    for (index = 0; index < cities.length; index++) {
        text += "<li>" + cities[index] + "</li>";
    }
    text += "</ul>";
    document.getElementById("ArrDispl").innerHTML = text;
}
</script>
</body>
</html>
```

## Output



# JavaScript – Objects

→ This code defines an object by name car and sets the value for all the attributes and displays the value for the attributes

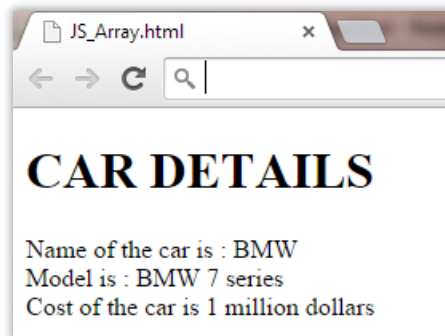
Variables are accessed using the dot(.) operator

Syntax: Object\_name.Variable\_name

## Example

```
<script>
var car = {
Name: "BMW",
Model: "BMW 7 series",
Cost: "1 million dollars"
};
document.getElementById("car details").innerHTML=
"Name of the car is : " + car.Name + "<br/>"
+ "Model is : " + car.model + "<br/>"
+ "Cost of the car is " + car.cost;
</script>
```

## Output



# JavaScript – Operators

→ When you want to perform any calculations, operators are used

→ Below we mention the JavaScript operators along with the operations that they perform:

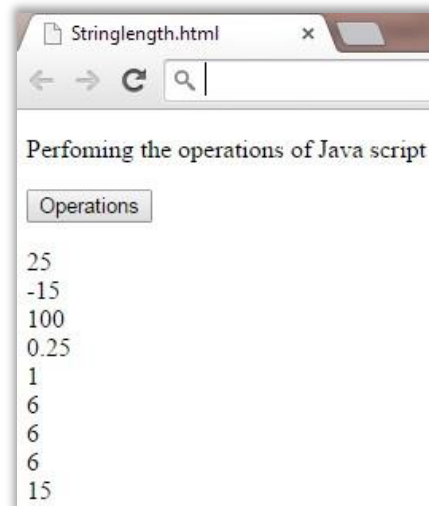
Operators	Operations
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo (returns remainder)
++	Increment the variable by one unit
--	Decrement the variable by one unit
+=, -=, *=, /=, %=	Expressions

# JavaScript – Operators (Contd.)

## Example

```
<script>
function Ops() {
var x = 5;
var y = 20;
var z1 = x + y;
var z2 = x - y;
var z3 = x * y;
var z4 = x / y;
var z5 = 21 % x;
var z6 = ++x;
var z7 = x++;
var z8 = --x;
y += 5;
y -= 10;
document.getElementById("result").innerHTML =
z1 + "<br/>" +
z2 + "<br/>" +
z3 + "<br/>" +
z4 + "<br/>" +
z5 + "<br/>" +
z6 + "<br/>" +
z8 + "<br/>" +
y + "<br/>";
}
</script>
```

## Output





# JavaScript – Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A function is executed when it is invoked
- A function can have zero or more arguments
- Biggest advantage of a function is [code reusability and modularity](#)

Syntax for function declaration:

```
function name_of_the_function (arguments)
{
    Statement - 1;
    Statement - 2;
    Statement - 3
}
```

# JavaScript – Function with one argument

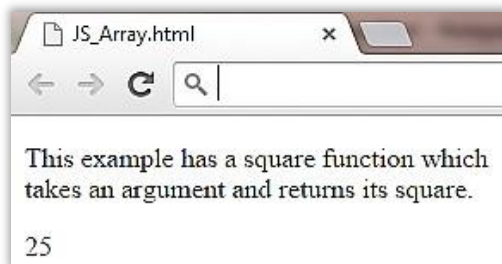
## Example

```
<!DOCTYPE html>
<html>
<body>
<p>This example has a square function which
takes an argument and returns its square.</p>
<p id="sqr"></p>
<script>
function square(a )
{ return a * a; }
document.getElementById("sqr").innerHTML = square(5);
</script>
</body>
</html>
```

→ This is the function named square and it takes an argument and returns its square

→ Square method is called by passing the value of the argument as 5

## Output



# JavaScript – Function with multiple arguments

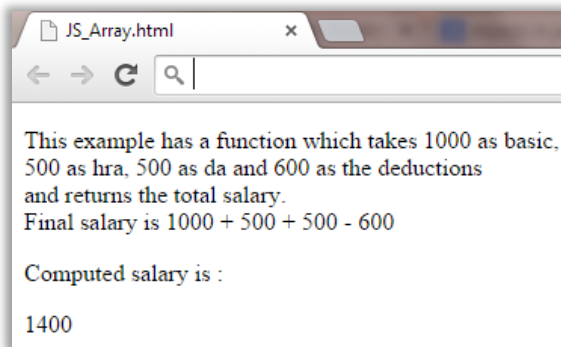
→ Here the ComputeSalary function takes basic, hra, da and deductions as the arguments and computes the final salary

## Example

```
<p> Computed salary is : </p>
<p id="salary"></p>
<script>
function ComputeSalary(basic, hra, da, deductions ) {

    var gross_salary = basic + hra + da;
    var total_salary = gross_salary - deductions;
    return total_salary;
}
document.getElementById("salary").innerHTML =
ComputeSalary(1000, 500, 500, 600);
</script>
</body>
</html>
```

## Output



# JavaScript – Function Arguments

→ Function `SumAndAverage()` is defined. In this function a loop is iterated for all the arguments passed to the function.


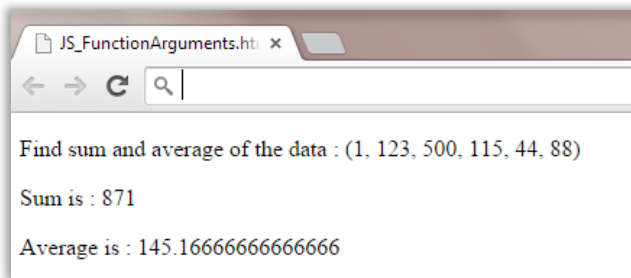
Sum of all the arguments are added in the loop

→ At the end `SumAndAverage()` function is called with the data

## Example

```
<script>
function SumAndAverage() {
var i, sum = 0;
for(i = 0; i < arguments.length; i++)
{ sum += arguments[i];}
document.getElementById("sum").innerHTML =
"Sum is : "+sum;
document.getElementById("Average").innerHTML =
"Average is : "+sum / arguments.length;
}
SumAndAverage(1, 123, 500, 115, 44, 88);
</script>
```

## Output



Arguments for the function need not be passed in the parenthesis of the function. They can be passed while calling a function and can be accessed by using arguments property

# JavaScript – this Pointer

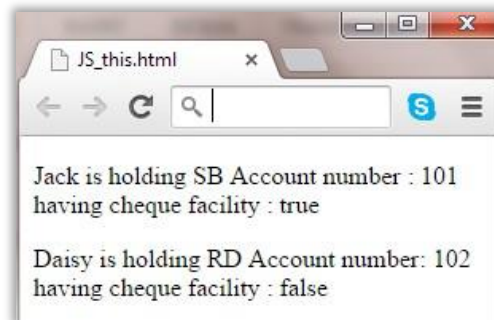
→ “this” pointer is one of the most powerful keyword in JavaScript. It is used to point to a particular attribute of the object

## Example

```
<script>
function BankAccount(AcctNumber, firstName, lastName, TypeOfAcct, ChequeFacility)
{
  this.AcctNumber = AcctNumber;
  this.firstName = firstName;
  this.lastName = lastName;
  this.TypeOfAcct = TypeOfAcct;
  this.ChequeFacility = ChequeFacility;
}

var act1 = new BankAccount(101,"Jack", "Thomas", "SB", "true");
var act2 = new BankAccount(102,"Daisy", "Dan", "RD", "false");
document.getElementById("account1").innerHTML =
act1.firstName + " is holding " + act1.TypeOfAcct +
" Account number : "+act1.AcctNumber +
" having cheque facility : " + act1.ChequeFacility;
document.getElementById("account2").innerHTML =
act2.firstName + " is holding " + act2.TypeOfAcct +
" Account number: "+act2.AcctNumber +
" having cheque facility : " + act2.ChequeFacility;
</script>
```

## Output



# JavaScript – this Pointer (Contd.)

- A function by name BankAccount receives the arguments
  - » Account number
  - » first name
  - » last name
  - » Type of account and
  - » Cheque Facilities
- **this pointer** points to the current object
- this pointer attribute is defined and the arguments are assigned to it
- Two accounts are created with the values
- The account objects information is displayed in the HTML paragraphs with id account1 and account2

# Showing an Alert

→ When you want to present the output in the message box, alert() function is used

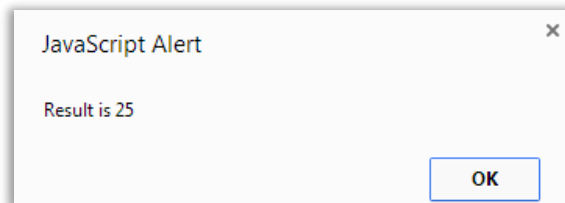
→ alert () is a function used to display the string in a message box

## Example

```
<!DOCTYPE html>
<html>
<body>
<script>
  var x = 5, y = 20;
  var z = x + y;
  var result = "Result is " + z;
  alert (result);
</script>
</body>
</html>
```

- x and y variables are declared and values are assigned
- Variable x and y are added and result is stored in the variable z
- The value stored in the variable z is appended to the string "Result is"
- Displays the result in the Message Box

## Output



alert function is frequently  
used for debugging JavaScript  
code

# Asking for Confirmation

→ A confirm box is often used if you want the user to verify or accept something

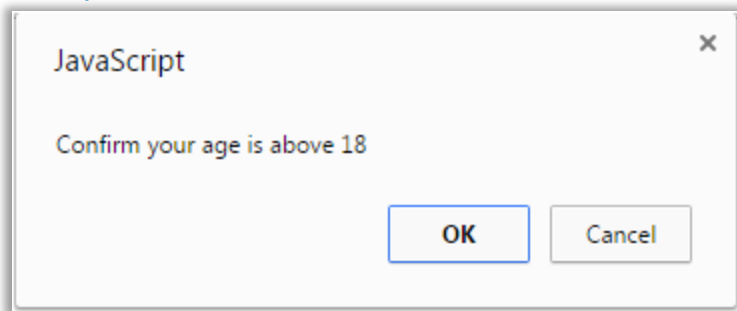
→ JavaScript `confirm()` method displays a dialog box with a specified message, along with an OK and a Cancel button

Example

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to display a confirm box.</p>
<button onclick="aFunction()">Click me</button>
<script>
function aFunction() {
    confirm("Confirm your age is above 18");
}
</script>

</body>
</html>
```

Output



`confirm()` method returns true if the user clicked "OK", and false otherwise.



# Taking User Input Through prompt Box


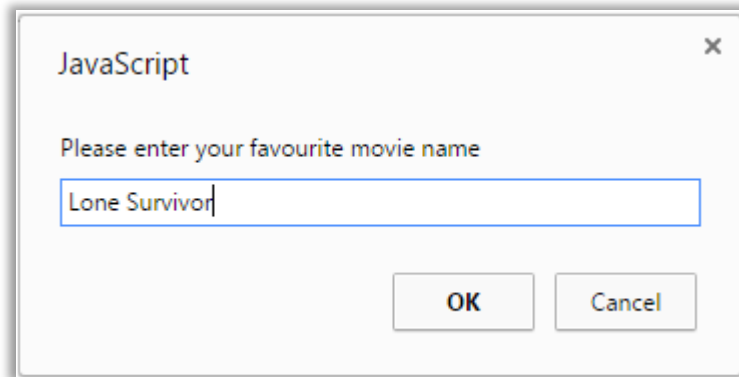
→ A prompt box is used if we want user to input a value

→ JavaScript prompt() method displays a dialog box that prompts the user for input

## Example

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to see the prompt box.</p>
<button onclick="aFunction()">Try it</button>
<p id="demo"></p>
<script>
function aFunction() {
    var movie= prompt("Please enter your favourite movie name", "Bourne Ultimatum");
    if (movie!= null) {
        document.getElementById("demo").innerHTML =
            "Your favourite movie - "+ movie;
    }
}
</script>
</body>
</html>
```

## Output



The prompt() method returns the input value if the user clicks "OK". If the user clicks "cancel" the method returns null

# JavaScript – Events

→ Events are signals that happen when a specific action occur

**Example:** On clicking a button, it generates `onclick` event

→ Some of the most commonly used events are:

- » `onclick`: User clicked on an HTML element (e.g. user clicked on a button)
- » `onchange`: After changing the HTML element (e.g. user changed his selection from dropdown)
- » `onmouseover`: User has moved the mouse on HTML element (e.g. user moved the mouse on img element)
- » `onmouseout`: User has moved the mouse away from HTML element (e.g. user moved the mouse on img element)
- » `onkeydown`: User has pressed a key on keyboard
- » `onload`: After the page is completely loaded

# JavaScript – Events (Contd.)

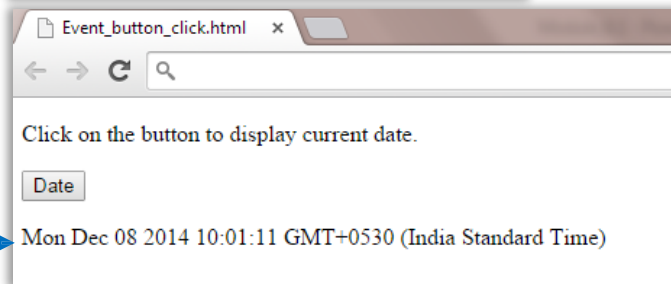
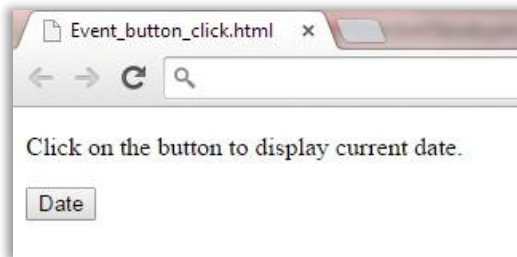
→ **onclick event** is generated when the user clicks on an HTML element

→ In our example, the onclick event executes **displayDate()** function. In the **displayDate()** function the current date and time is displayed to the user

## Example

```
<button onclick="displayDate()">Date</button>
<script>
function displayDate() {
  document.getElementById("display").innerHTML = Date();
}
</script>
<p id="display"></p>
```

## Output



When the date button is clicked, the current date and time is displayed



# Annie's Question



What is the difference between onclick and onmouseover?

# Annie's Answer



**Ans.** When an item is clicked onclick event is generated and when mouse is taken over the HTML element then onmouseover event is generated for that HTML element

# JavaScript – String functions

→ JavaScript provides a handful of utility methods to manipulate Strings

→ Some of the string functions are:

- » `charAt()`: Returns the character at the given position
- » `concat()`: Adds two strings
- » `replace()`: Replaces a string with another
- » `substr()`: Obtains the substring from the given string
- » `toUpperCase()`: to convert a String to uppercase
- » `toLowerCase()`: to convert a String to lowercase



The length of a String can be found using built in length property e.g. `var aString= "JavaScript"; alert(aString.length);` above code will generate an alert box with length of the string which is 10

# JavaScript – String functions (Contd.)

## Example

```
<p id="strlen"></p>
<script>
var str="I am learning java script";
document.getElementById("strlen").innerHTML = str.length;

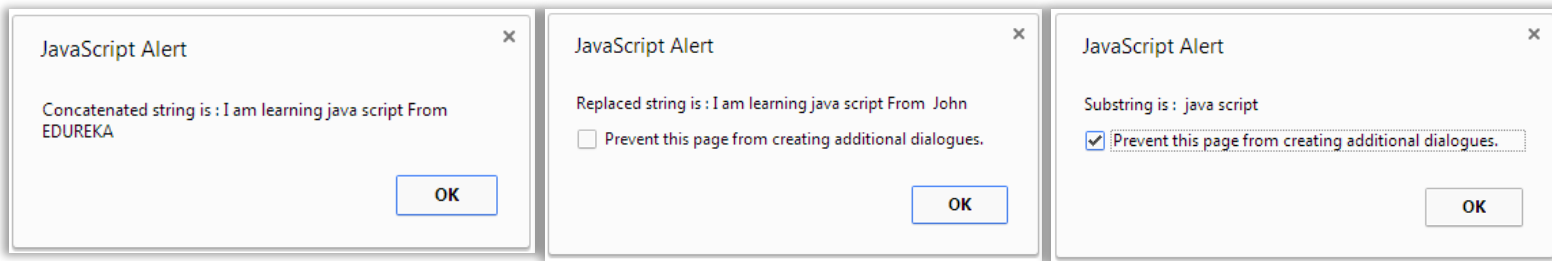
var str1 = str.charAt(2);

var str2 = str.concat (" From EDUREKA");
alert ("Concatenated string is : " + str2);

var str3 = str2.replace ("EDUREKA", "John");
alert ("Replaced string is : " + str3);

var str4 = str2.substr (13, 12);
alert ("Substring is : " + str4);
</script>
```

## Output



# JavaScript – Number functions

JavaScript provides functions to manipulate JavaScript numbers. Some of them are listed below :

- `toFixed()` returns a string, with a number rounded and written with a specified number of decimals
- `toPrecision()` returns a string, with a number written with a specified length
- `parseInt()` parses a string and returns a **whole number**
- `parseFloat()` parses a string and returns a **number**
- `toString()` returns a number as a string



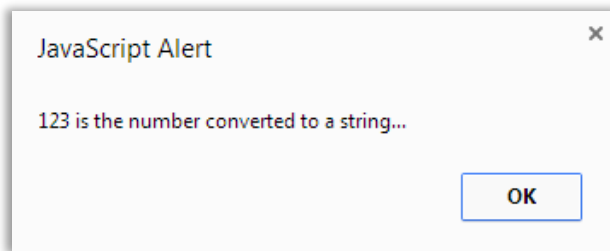
# JavaScript – Number functions (Contd.)

→ `toString()` converts the given number to string. As seen in the example, though  $(100+23)$  is given in the program, `toString` function evaluates the numbers and then converts it to a string

## Example

```
<script>
var x = 123;
var str = (100+23).toString();
str = str + " is the number converted to a string...";
alert (str);
</script>
```

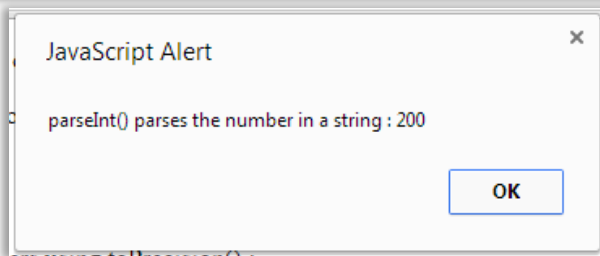
## Output



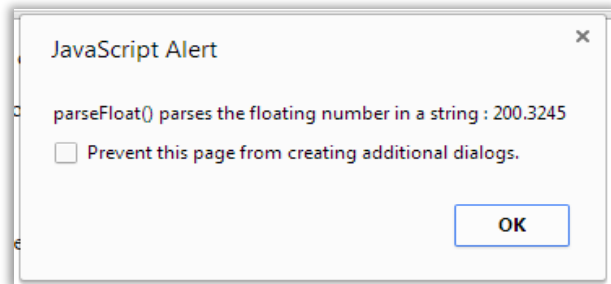
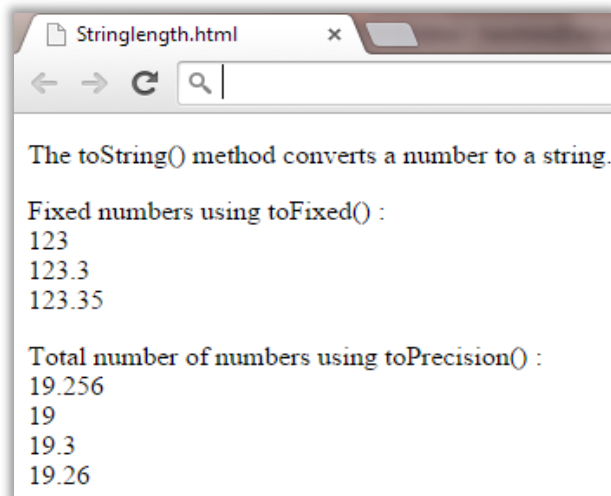
# JavaScript – Number functions (Contd.)

## Example

```
<p id="demo"></p>
<p id="fixed_num"/>
<p id="precision_num"/>
<script>
var x = 123.3456;
document.getElementById("fixed_num").innerHTML
= "Fixed numbers using toFixed() :<br/> "
+ x.toFixed(0) + "<br/>" + x.toFixed(1)
+ "<br/>" + x.toFixed(2);
var y = 19.256;
document.getElementById("precision_num").innerHTML
= "Total number of numbers using toPrecision() :<br/> "
+ y.toPrecision() + "<br/>" + y.toPrecision(2)
+ "<br/>" + y.toPrecision(3) + "<br/>" + y.toPrecision(4);
var str = "200.234";
var num = parseInt(str);
alert ("parseInt() parses the number in a string : " + num);
var str = "200.3245";
var num1 = parseFloat(str);
alert ("parseFloat() parses the floating number in a string : " + num1 );
</script>
```



## Outputs



# Annie's Question



What is the output of  $100\%2$ ?

# Annie's Answer



**Ans.** 0 is the answer as the reminder of  $100\%2 = 0$

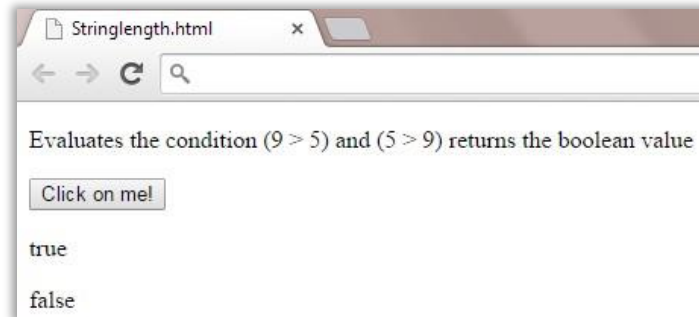
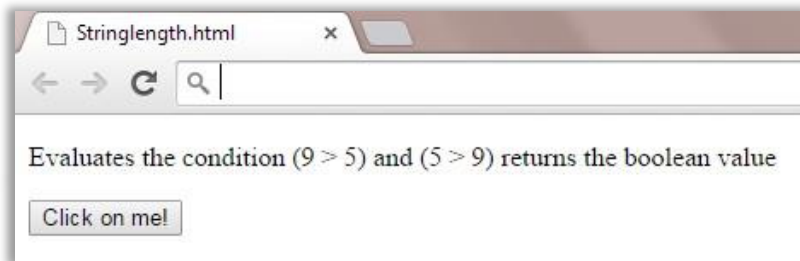
# JavaScript – Boolean function

→ JavaScript provides a `Boolean()` function which can be used to find out if an expression (or a variable) is true or false

## Example

```
<script>
function BooleanDisplay() {
document.getElementById("bool").innerHTML = Boolean(9 > 5);
document.getElementById("bool1").innerHTML = Boolean(5 > 9);
}
</script>
```

## Output



# JavaScript – Math functions

Math object provides some very important functions that can be used to perform mathematical computation

Some of the functions of Math object are:

- `random()` : Returns a random number
- `max()` : Returns the maximum number from a list of arguments
- `min()` : Returns the minimum number from a list of arguments
- `round()` : Rounds a number to the nearest integer and returns it
- `ceil()` : Rounds a number **up to the nearest integer** and returns it
- `floor()` : Rounds a number **down to the nearest integer** and returns it



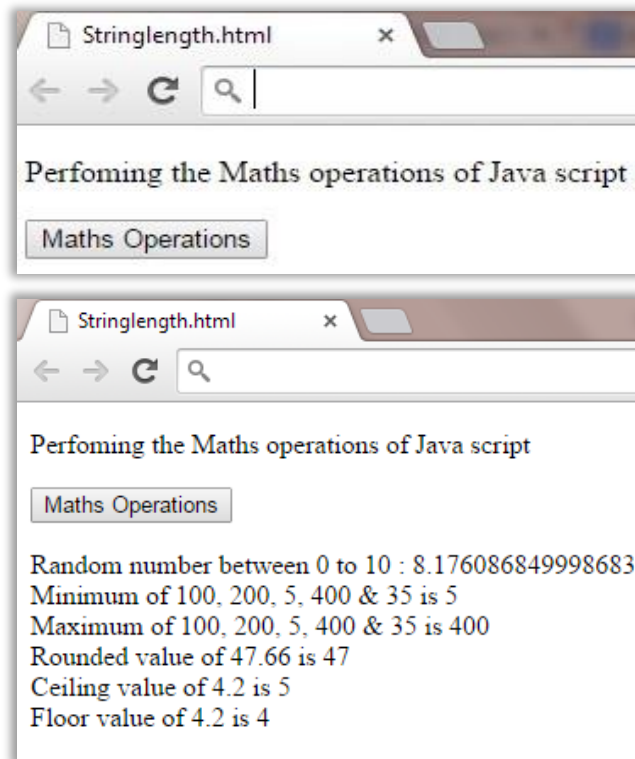
`random()` returns a number between 0 (inclusive), and 1 (exclusive) so on calling random function you will always get a number lower than 1

# JavaScript – Math functions (Contd.)

## Example

```
<script>
function Math_ops()
{
var random_num = Math.random() * 10;
var min_num = Math.min (100,200,5,400, 35);
var max_num = Math.max (100,200,5,400, 35);
var round_num = Math.round (47.36);
var ciel_num = Math.ceil (4.2);
var floor_num = Math.floor (4.2);
document.getElementById("maths").innerHTML =
"Random number between 0 to 10 : " + random_num + "<br/>" +
"Minimum of 100, 200, 5, 400 & 35 is " + min_num + "<br/>" +
"Maximum of 100, 200, 5, 400 & 35 is " + max_num + "<br/>" +
"Rounded value of 47.66 is " + round_num + "<br/>" +
"Ceiling value of 4.2 is " + ciel_num + "<br/>" +
"Floor value of 4.2 is " + floor_num + "<br/>";
}
</script>
```

## Output



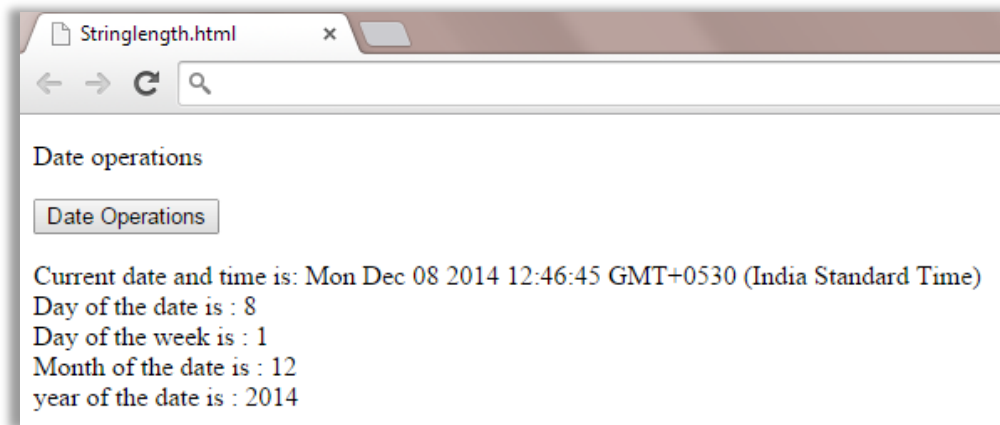
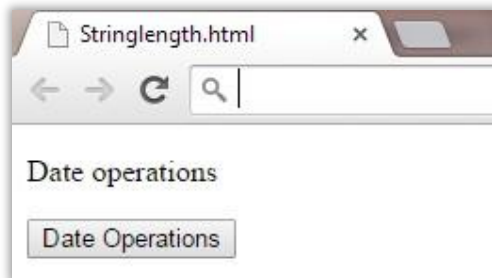
# JavaScript – Working with Dates

## Example

```
<script>
function Date_ops()
{
var dt = new Date();
var date = dt.getDate();
var day = dt.getDay();
var month = dt.getMonth()+1;
var year = dt.getFullYear();
document.getElementById("dates").innerHTML =
"Current date and time is: " + dt + "<br/>" +
"Day of the date is      : " + date + "<br/>" +
"Day of the week is      : " + day + "<br/>" +
"Month of the date is    : " + month + "<br/>" +
"year of the date is     : " + year + "<br/>" ;
}
</script>
```

- Date object with current date is created
- Gets the day of the month
- Gets the day of the week
- Return the month (0 - 11)
- Returns the year of the given date

## Output





# JavaScript – Comparison Operators

→ JavaScript provides following Comparison operators :

Relational Operators	Operations
==	Equals to
===	Equals to with the same type
!=	Not equals to
!==	Not equals or not the same type
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

# JavaScript – Logical Operators

→ Following logical operators can be used for comparisons

Logical Operators	Operations
&& (And operator)	Returns true if both the conditions are satisfied
(Or Operator)	Returns true if either condition is satisfied
! (Not Operator)	Negates the Boolean value from true to false and false to true

# JavaScript – Reserved Words

→ Reserved words are words with a specific meaning that cannot be redefined by user. Some of the reserved words in JavaScript are:

- » abstract
- » arguments
- » boolean
- » break
- » byte
- » case
- » catch
- » char
- » class\*
- » const
- » continue
- » if
- » implements
- » static
- » debugger

- » import
- » in
- » instanceof
- » int
- » interface
- » let long
- » native
- » new
- » null
- » package
- » private
- » protected
- » public
- » return
- » short

- » switch
- » synchronized
- » this
- » throw
- » throws
- » transient
- » true
- » try
- » typeof
- » var
- » void
- » volatile
- » while
- » with
- » yield

- » default
- » delete
- » do
- » double
- » else
- » enum\*
- » eval
- » export
- » extends
- » false
- » final
- » finally
- » float
- » for
- » function

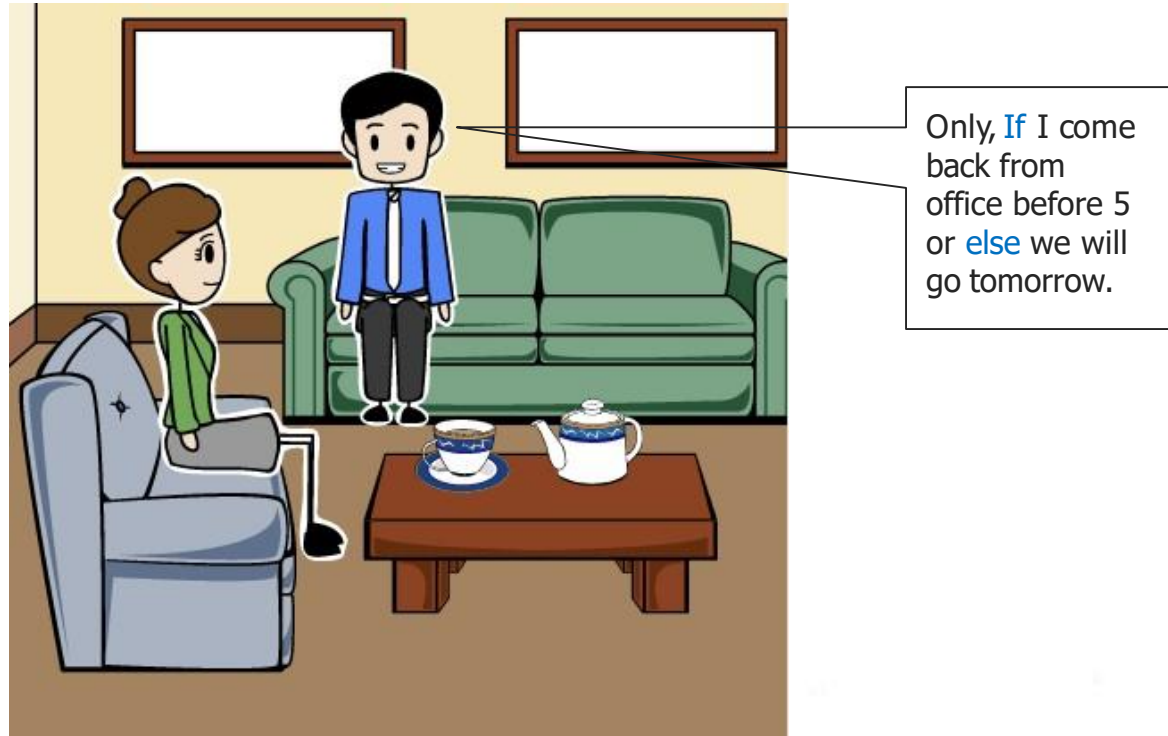
Reserved words cannot be used as variables or functions

# Choices in Real Life

Honey can we  
go for a movie  
today evening?



# If...else Condition



# JavaScript Conditionals if-else statement

→ **if-else condition** is used to check whether the condition is satisfied (evaluates to true) or not

→ Based on the result, appropriate action is taken

**Syntax:**

```
if (condition)
{
  st-1;
  st-2;
}
else
{
  st-3;
  st-4;
}
```

→ If the condition is satisfied, st-1, st-2 will be executed. If the condition fails, the else part of the if condition.  
i.e., st-3 and st-4 will be executed

# JavaScript – break statement

→ We can use break statement to terminate the execution of a loop and come out of a loop.

When the “value of i” is greater than 16 then control comes out of the loop using the break statement

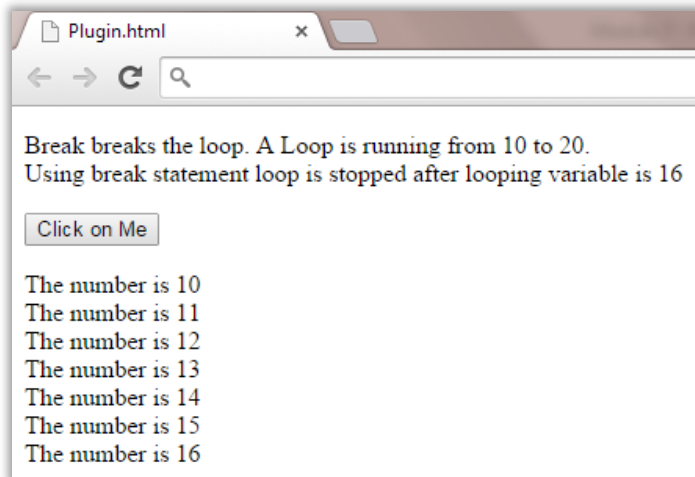
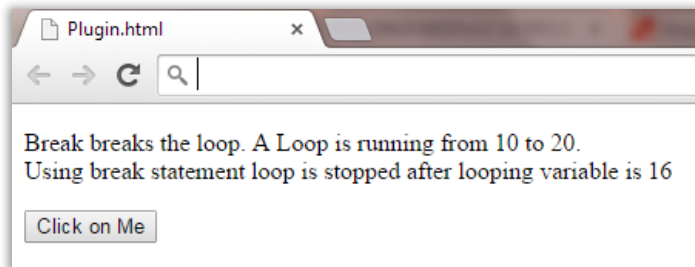
Example

```
<script>
function BreakFunction() {
    var text = ""
    var i;
    for (i = 10; i < 20; i++) {
        if (i > 16) {break;}
        text += "The number is " + i + "<br>";
    }
    document.getElementById("brk").innerHTML = text;
}
</script>
```



break statement can also be used with switch statement

## Output



# JavaScript Conditionals if-else statement

## Example

```
<p id="vote"></p>
<script>
function VotingStatus() {
var age = prompt ("Enter your age :");
if (age >= 18) {
status = "You are eligible for voting...Happy voting!";
} else {
status = "Sorry, you are not eligible for voting..";
}
document.getElementById("vote").innerHTML = status;
}
</script>
```

Displays a dialog box  
and asks for the age

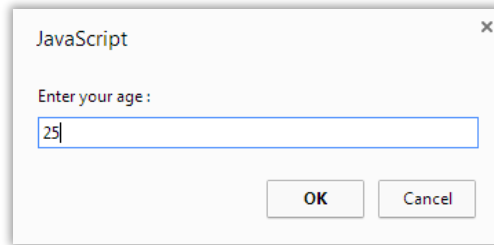


JavaScript is case sensitive language. So make sure you use the JavaScripts keywords appropriately. So if you use IF or If for conditional statement it will not be evaluated.

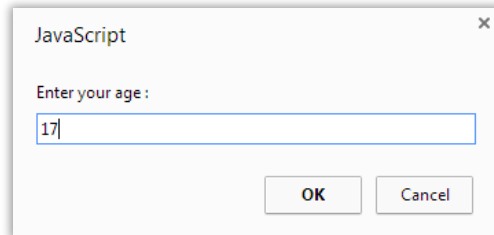
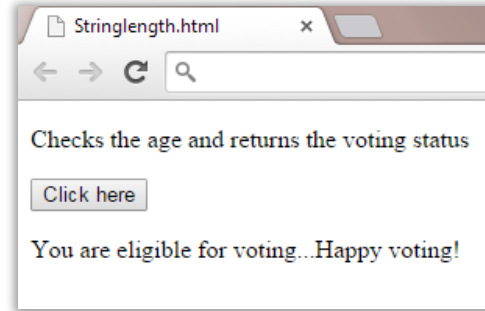


# JavaScript Conditionals if-else statement (Contd.)

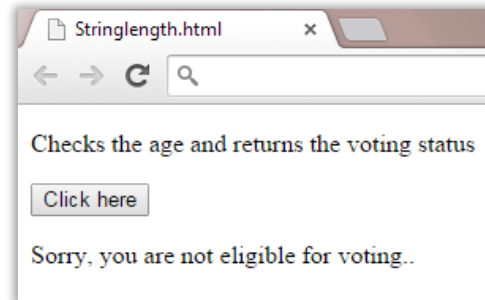
## Output



A JavaScript dialog box titled "JavaScript" with a close button (X). It contains the text "Enter your age :" followed by a text input field containing the number "25". At the bottom are "OK" and "Cancel" buttons.



A JavaScript dialog box titled "JavaScript" with a close button (X). It contains the text "Enter your age :" followed by a text input field containing the number "17". At the bottom are "OK" and "Cancel" buttons.



# JavaScript Conditionals if-else if statement

→ We can use JavaScript else if statement to specify a new condition if the first condition is false

```
if (first_condition) {  
    st-1;  
    st-2;  
} else if (second_condition) {  
    st-3;  
    st-4;  
} else {  
    st-5;  
}
```

- » If first\_condition is true then st-1 and st-2 will be executed and second condition will not be checked
- » If first\_condition is false then second\_condition will be checked and if it is true st-3 and st-4 will be executed
- » If both first\_condition and second\_condition are false st-5 within the else block will be executed

# JavaScript – Switch statement

→ If you have multiple conditions to check its better to use switch statement.

Syntax:

```
switch (value){  
  case value1: st-1; st-2 break;  
  case value2: st-3; st-4 break;  
  default : st-5; st-6;}
```

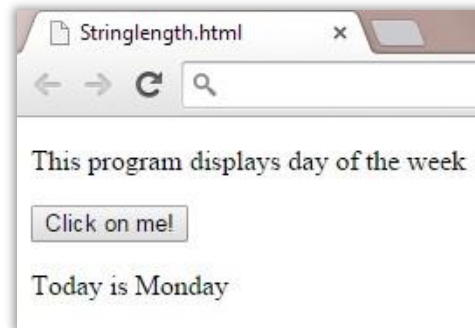
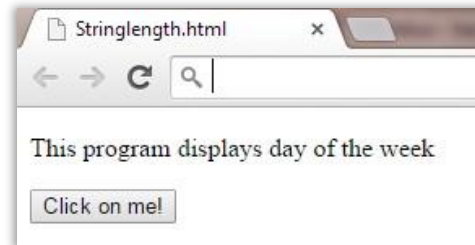
→ Data given in the switch statement is evaluated with each case. If the case matches then the statements inside that block is executed. If the value does not match with any case then default code block is executed

# JavaScript – Switch statement (Contd.)

## Example

```
<p id="dow"></p>
<script>
function DayOfWeek() {
var dt = new Date();
var DayOfWeek = dt.getDay();
var day;
switch (DayOfWeek) {
case 0: day = "Sunday"; break;
case 1: day = "Monday"; break;
case 2: day = "Tuesday"; break;
case 3: day = "Wednesday"; break;
case 4: day = "Thursday"; break;
case 5: day = "Friday"; break;
case 6: day = "Saturday"; break;
}
document.getElementById("dow").innerHTML = "Today is " + day;
}
</script>
```

## Output



# Annie's Question



Why switch is used when if condition is there?

# Annie's Answer



**Ans.** Rather than using so many if conditions, switch statement can be used.

# JavaScript – Error

- While executing JavaScript code, errors can occur
- Errors can be coding errors made by the programmer or due to wrong input etc.
- JavaScript `try...catch...finally` blocks can be used for **error handling**
- A block of code that can result in an error is surrounded by **try statement**
- The **catch** statement lets you handle the error
- The **finally** statement lets you execute code after try and catch
- A **throw** statement lets you throw a custom error

**Syntax:** `try { //code to run; }`

`catch(e) { //code to run if an exception occurs; }`

`finally { //code that is always executed regardless of an exception occurs or not; }`



The try block must be followed by either a catch block or a finally block

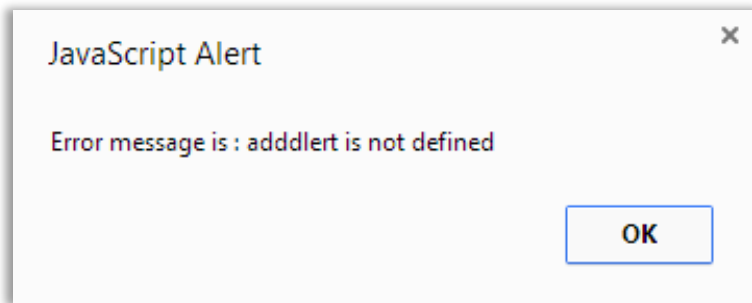
# Using try catch statement

→ In the code, `adddalert ()` is used instead of the `alert` command, since it is not valid command, control goes to catch block and `err.message` contains the error and this error message is displayed in an alert box

## Example

```
<script>
try {
  adddalert("Are you concentrating???");
}
catch(err) {
  alert("Error message is : " + err.message);
}
</script>
```

## Output





# JavaScript – Error - try...catch...finally

## Example

```
<script>
function TestData() {
    var error_msg, data;
    error_msg = document.getElementById("error_msg");
    error_msg.innerHTML = "";
    data = document.getElementById("data").value;
    try {
        if(data == "") throw "Given input is empty";
        if(isNaN(data)) throw "Given input is not a number";
        if(data > 200) throw "Given number is too high. Please enter between 100 to 200";
        if(data < 100) throw "Given number is too low. Please enter between 100 to 200";

        alert ("Congratulations...Correct data is entered: "+data);
    }
    catch(err) {
        error_msg.innerHTML = err;
    }
    finally
    {
        alert ("In the finally block...");
    }
}
</script>
```

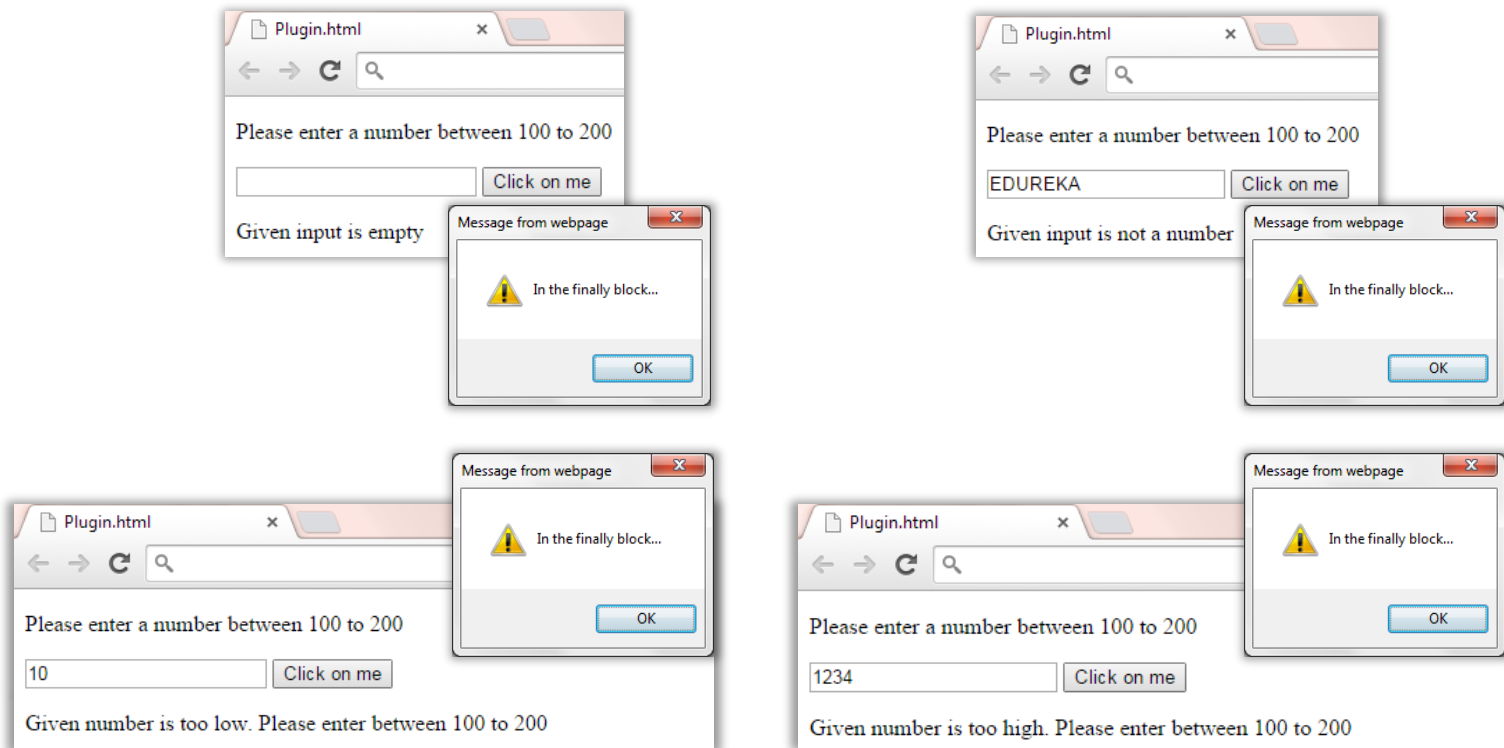
Try block

Catch block

Finally block

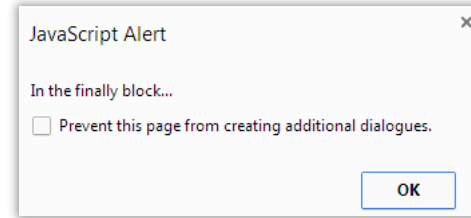
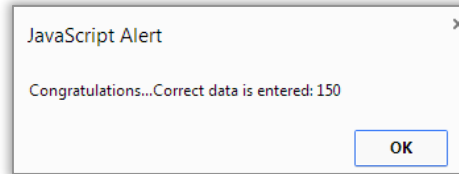
# JavaScript – Error - try...catch...finally (Contd.)

## Output




# JavaScript – Error - try...catch...finally (Contd.)

## Output



- Entered data in the text box is obtained using `TestData()` function
- If no data is entered error message "Given number is empty" is thrown. Similarly, based on the given input corresponding error message is thrown
- If the user entered the data between 100 to 200 then an alert message is displayed as "Congratulations...Correct data is entered"



finally block is always executed regardless of whether an error is thrown or not

# JavaScript – Variable Hoisting

- Variable declarations can be done at any point in the program before executing the code
- JavaScript has a unique feature where all the declarations are moved to the top of that particular function or program. This is called "Hoisting"
- In JavaScript, a variable can also be declared after its been used. This is not the correct way of executing the code. This can be avoided using strict mode which will be explained in the next topic

## Example

```
<script>
var a;
a = 10;

elem = document.getElementById("demo");
elem.innerHTML = a;
</script>
```

Declaring variable at the start

```
<script>
a = 5;

elem = document.getElementById("demo");
elem.innerHTML = a;

var a;
</script>
```

Declaring variable at the end

# JavaScript – Variable Hoisting (Contd.)

- In the example, variable `int_data` is assigned to 15 without declaring it. This variable is declared at the end of the script tag. JavaScript will automatically **hoist the declaration to the top**
- Object of info is obtained and stored in `elem` variable and it is updated with the `int_data`

## Example

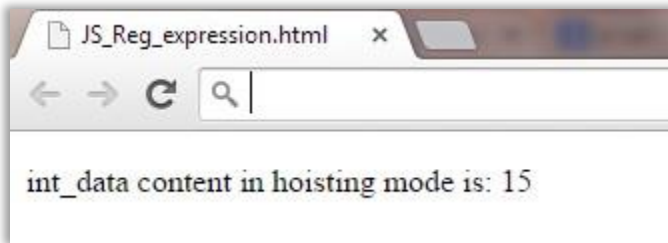
```
<script>
int_data = 15;
elem = document.getElementById("info");
elem.innerHTML =
"int_data content in hoisting mode is: " + int_data;
var int_data, elem;
</script>
```

Assign 15 to `int_data`

Display `int_data` in the element

Declare `int_data`

## Output



# JavaScript – Strict Mode

- With strict mode, hoisting is not possible
- Strict mode can be enabled by using "use strict"; keyword
- "use strict"; should be used in the start of the script or function else the script works in hoisting mode
- Defining the duplicate variables declaration is an error in strict mode

## Example

```
<script>
"use strict";
y= 600;
CallFunction();
function CallFunction()
{
  alert ("Inside a function");
  x = 500;
  alert (x);
  alert (y);
}
</script>
```

→ Strict mode is defined here

→ Error, cannot assign a value to y, unless its declared

→ Error

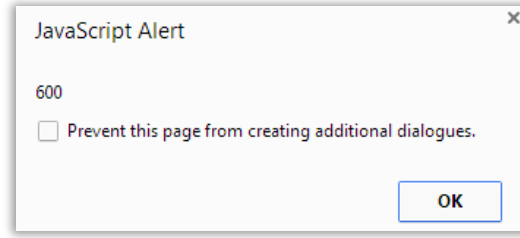
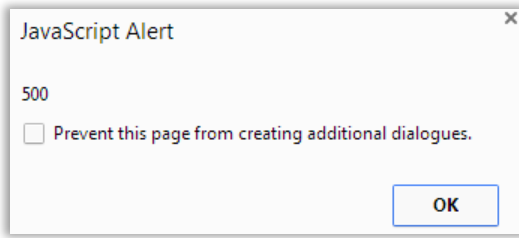
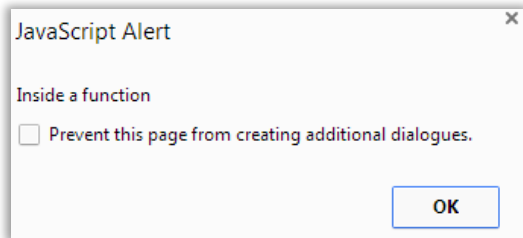
This is an example showing the use of "use strict" in the script beginning

# JavaScript – Strict and Non-strict Mode

## Output in strict mode:



## Output in non-strict mode:



Now let us analyze the difference between the two outputs

# JavaScript – Strict Mode in the Function

→ Here "use strict"; command is used inside the function. So the strict mode is applicable only to the function

→ This should be used in the start of the function

## Example

```
<script>
y= 600;
alert (y);
CallFunction();
function CallFunction()
{
  "use strict";
  alert ("Inside a function");
  x = 500;
  alert (x);
  alert (y);
}
</script>
```

→ This is not an error since "use strict" is not used in the script



# JavaScript – Performance

To increase the speed of execution of the JavaScript, the performance factors have to be considered

Some of them are listed below:

- **Reduce activity in loops:** If certain statements are not required to be as part of loops then place them outside the loop
  - » The unnecessary statements in the loop will consume the CPU execution time and degrades the performance of the JavaScript.
- **Avoid declaring variables and objects unnecessarily**  
Example: `var full_name = first_name + last_name;`  
`document.getElementById("test").innerHTML = full_name`

The above example can be reframed as: `document.getElementById("test").innerHTML = first_name + last_name;`

# JavaScript – Performance (Contd.)

- **Reduce the DOM access:** DOM stands for **Document Object model**. Functions like `getElementById()` returns the object of the element
    - » The object returned by `getElementById()` can be stored in a variable and that variable can be used when ever required. This avoids the problem of using `getElementbyId()` every time to return the object
    - » Every time DOM is used, it searches the tree for the object's address. This can be costly in terms of performance
  - **Define the script at the end:** Place the script at the end of the body so that the page is first loaded and then the script
- Otherwise time is consumed to load the scripts and then page is displayed. User will see a delay in seeing the webpage which is not the right form of webpage development

# JavaScript – JSON

- When you design an application, you will have to communicate with the remote computer. To do this you will have to select a data format and protocol that guide the communication between the browser and the server
- JSON stands for JavaScript Object Notation. It is a standard way of storing and transporting data and is best-suited for web applications
- Many applications uses JSON as the format for exchanging the data between the applications
- We can convert a JSON format text to a JavaScript object using JSON.parse() function
- JSON data can be stored separately in a file. The file must be saved using the extension “.json”

# JavaScript – JSON Example

- Bank account details are stored in `bank variable` which is stored in `text JSON object`
- `JSON.parse(text)` reads the JSON data and stores in an object `obj`
- Account Number, Name and Address from JSON object is printed in the paragraph which has the id `json_data`

## Example

```
<p id="json_data"></p>
<script>
var text = '{"bank":[' +
'{"ActNumber":"SB101","Name":"Doe","Address":"California" },' +
'{"ActNumber":"SB102","Name":"Divya","Address":"Bangalore" },' +
'{"ActNumber":"SB103","Name":"Jones","Address":"New York" }]}';

obj = JSON.parse(text);
document.getElementById("json_data").innerHTML =
obj.bank[1].ActNumber + " " + obj.bank[1].Name+ " " + obj.bank[1].Address;
</script>
```

## Output



we have used JavaScript built-in function `JSON.parse()` to convert the string into a JavaScript object

# JavaScript – Function as Variable Name

→ JavaScript allows you to assign a function to a variable and then the variable name itself can be used as a function

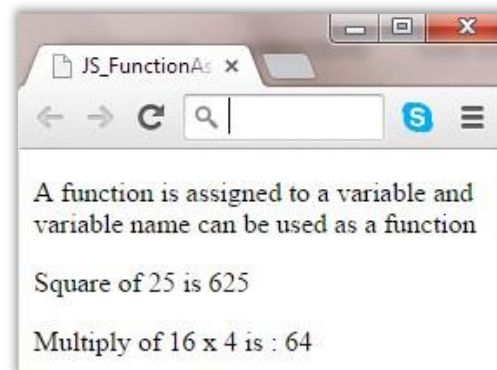
## Example

```
<script>
var sqr = function (a) {return a * a};
var multiply = function (a, b) {return a * b};
document.getElementById("sqr_para").innerHTML
= "Square of 25 is " + sqr(25);
document.getElementById("mul_para").innerHTML
= "Multiply of 16 x 4 is : " + multiply(16, 4);
</script>
```



Here we have assigned functions to variables `sqr` and `multiply` respectively

## Output



# JavaScript – Function Closures

→ A closure is an inner function that has access to the outer function's variables

→ In the below example, you can see a function defined inside another function which increments the counter value by 1, when ever you click on the counter button

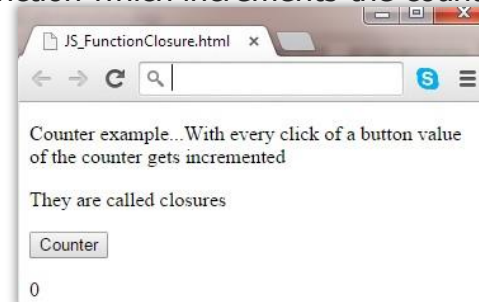
Example

Output

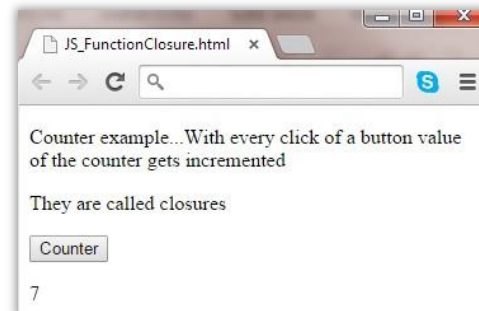
```
<p id="value">0</p>
<script>
var add = (function () {
  var counter = 0;
  return function () {return counter += 1;}
})();
function MyCounter() {
  document.getElementById("value").innerHTML = add();
}
</script>
```



statement counter=0; will be executed only once.



Before counter button is clicked



After counter button is clicked

# JavaScript – Function Closures (Contd.)

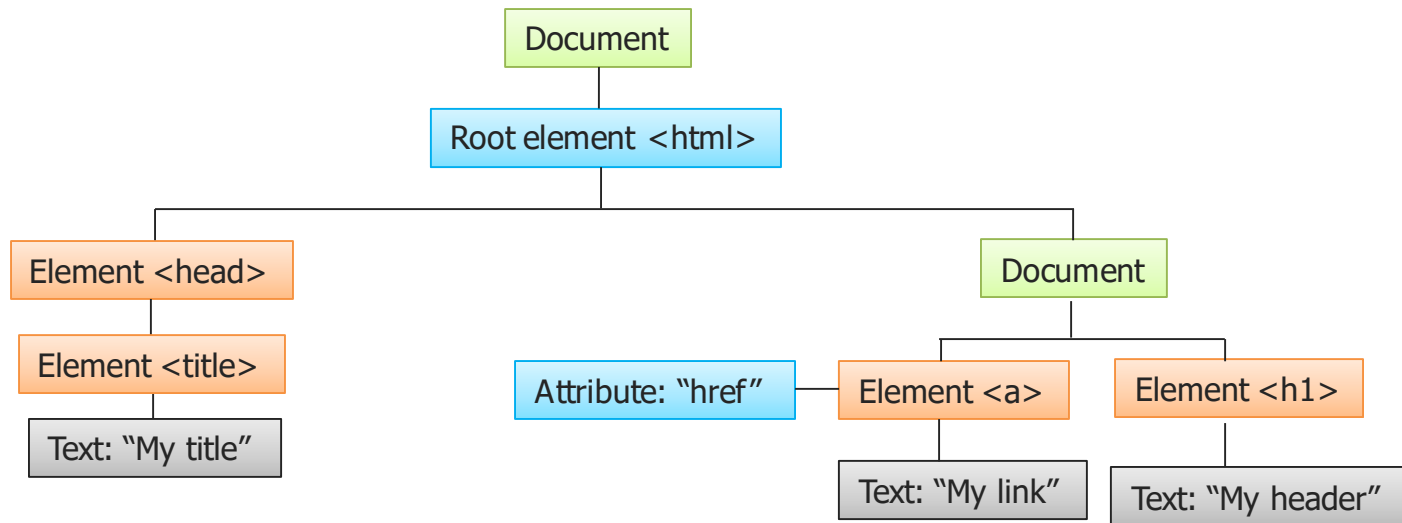
- A counter button is displayed. On click of this button, `MyCounter()` function is invoked
- In `MyCounter()` function, paragraph with id value is updated by calling `add()` function
- In `add()` function, counter variable is initialized to 0 and it calls another inner function which increments the value of counter by one and returns it
- Hence with every click of the button counter, value of counter gets incremented and the value is displayed

# JavaScript – DOM

→ **DOM** stands for **Document Object Model**

→ HTML elements are constructed as a tree and it can be accessed using DOM

→ **Tree** structure looks like this:



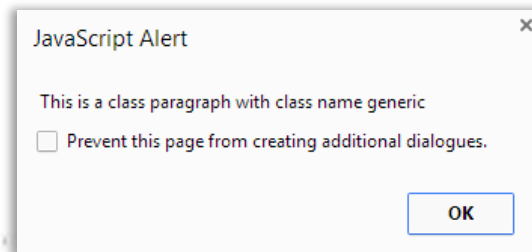
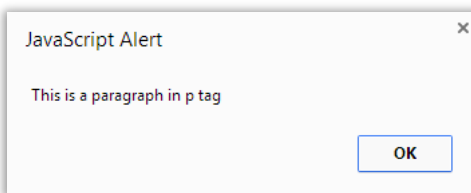


# DOM – Methods

## Example

```
<p> This is a paragraph in p tag</p>
<p id="test"> 5345 </p>
<p class="generic">
  This is a class paragraph with class name generic </p>
<script>
document.getElementById("test").innerHTML = 6700;
var x = document.getElementsByTagName("p");
alert (x[0].innerHTML);
var y = document.getElementsByClassName("generic");
alert (y[0].innerHTML);
</script>
```

## Output



# DOM – Elements

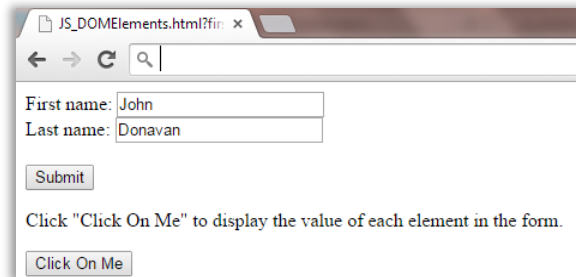
→ All the elements of the form is obtained using `getElementById()` and it is displayed

## Example

```
<form id="form1" >
First name: <input type="text" name="first_name" value="John"><br>
Last name: <input type="text" name="last_name" value="Donavan"><br><br>
<input type="submit" value="Submit">
</form>

<p>Click "Click On Me" to display the value of each element in the form.</p>
<button onclick="FormDataDisplay()">Click On Me</button>
<p id="DisplayInfo"></p>
<script>
function FormDataDisplay() {
var data = document.getElementById("form1");
var text = "";
var i;
for (i = 0; i < data.length ;i++)
{text += data.elements[i].value + "<br>";}
document.getElementById("DisplayInfo").innerHTML = text;
}
</script>
```

## Output



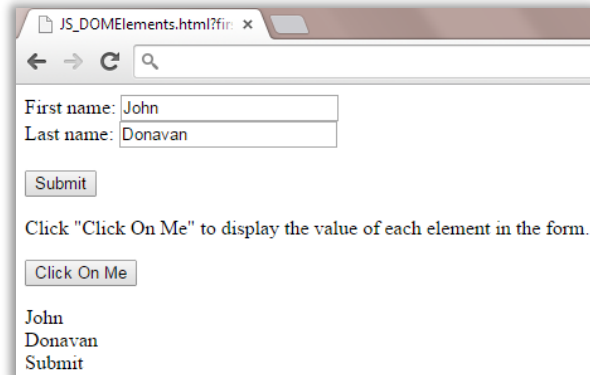
JS\_DOMElements.html?fir: x

First name: John  
Last name: Donavan

Submit

Click "Click On Me" to display the value of each element in the form.

Click On Me



JS\_DOMElements.html?fir: x

First name: John  
Last name: Donavan

Submit

Click "Click On Me" to display the value of each element in the form.

Click On Me

John  
Donavan  
Submit

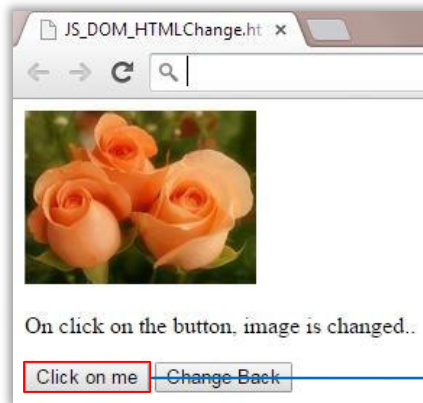
# DOM – HTML

→ Below we are changing the src attribute of an image element on clicking the buttons

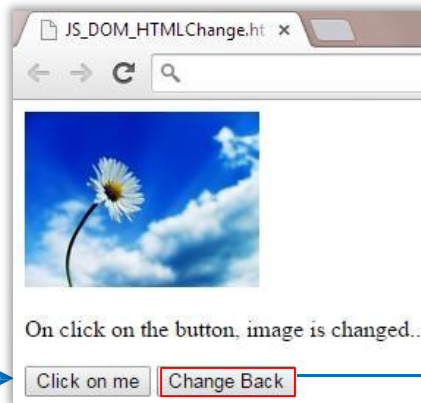
Example

```
<script>
function ChangePicture()
{document.getElementById("image1").src = "single_flower.jpg";}
function ChangeBack()
{document.getElementById("image1").src = "roses.jpg";}
</script>
```

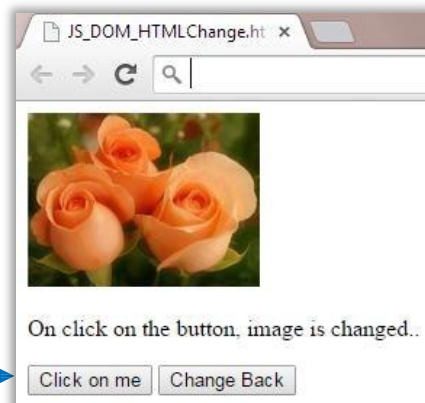
Output



On click of "Click on me", the image is changed



On click of "Change Back", the previous image is displayed



# DOM – CSS

- In this example, we are changing the color of the text
- A button is displayed and on click of this button, header color is changed from black to red
- `style.color` property of DOM is used to change the color of the header

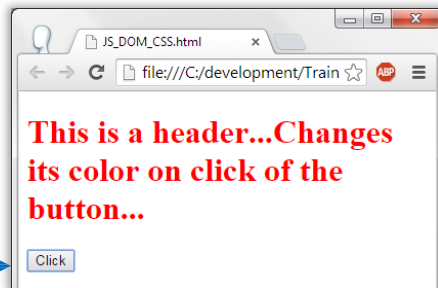
Example

```
<h1 id="header">  
This is a header...Changes its color on click of the button...</h1>  
<button type="button"  
onclick="document.getElementById('header').style.color = 'red'">  
Click</button>
```

Output



On click of "Click",  
the text color is  
changed to red





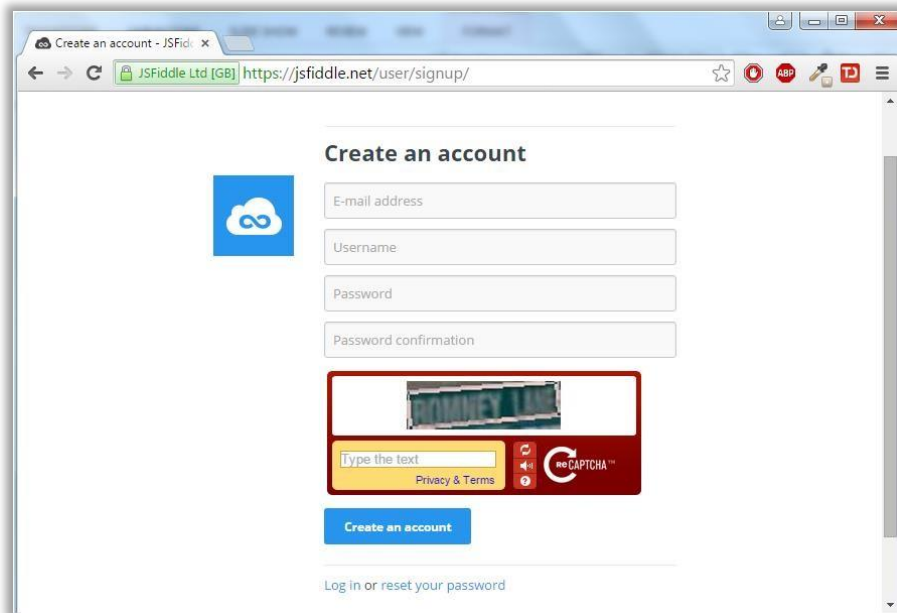
# Working with Online Code Editors

# Working with Online Code Editors

- There are times when you will need to share your code with a colleague or some other person. Rather than sending the source code through email, we can use online code editing facility to share your code to public
- There are various websites that provide the online code editing facility. Some of them are listed below :
  - » [JSFiddle](#)
  - » [Codepen](#)
  - » [CSS Deck](#)
  - » [JS Bin](#)
  - » [Codeanywhere](#)

# Working with JSFiddle - Signup

→ Go to <https://jsfiddle.net/> and sign up for an account. Although JSFiddle lets you create fiddles without creating an account but its advisable to create an account on JSFiddle so that you can track all of your previous fiddles from a dashboard.



The screenshot shows a web browser window with the address bar displaying 'https://jsfiddle.net/user/signup/'. The page title is 'Create an account - JSFiddle'. The main heading is 'Create an account'. To the left of the form is the JSFiddle logo, a blue square with a white infinity symbol. The form contains four input fields: 'E-mail address', 'Username', 'Password', and 'Password confirmation'. Below these fields is a reCAPTCHA widget with a red border, showing a distorted image of a street sign that says 'ROMNEY LANE'. Below the reCAPTCHA is a yellow button labeled 'Type the text' and a link for 'Privacy & Terms'. At the bottom of the form is a blue button labeled 'Create an account'. Below the button is a link that says 'Log in or reset your password'.

# Working with JSFiddle – Signup (Contd.)

→ Once you signed up and logged into your account, you are ready to create your first JSFiddle. Below is our first fiddle :

```
<h1 id="header">My First JSFiddle</h1>
<button onclick="cal();">Press Me</button>
```

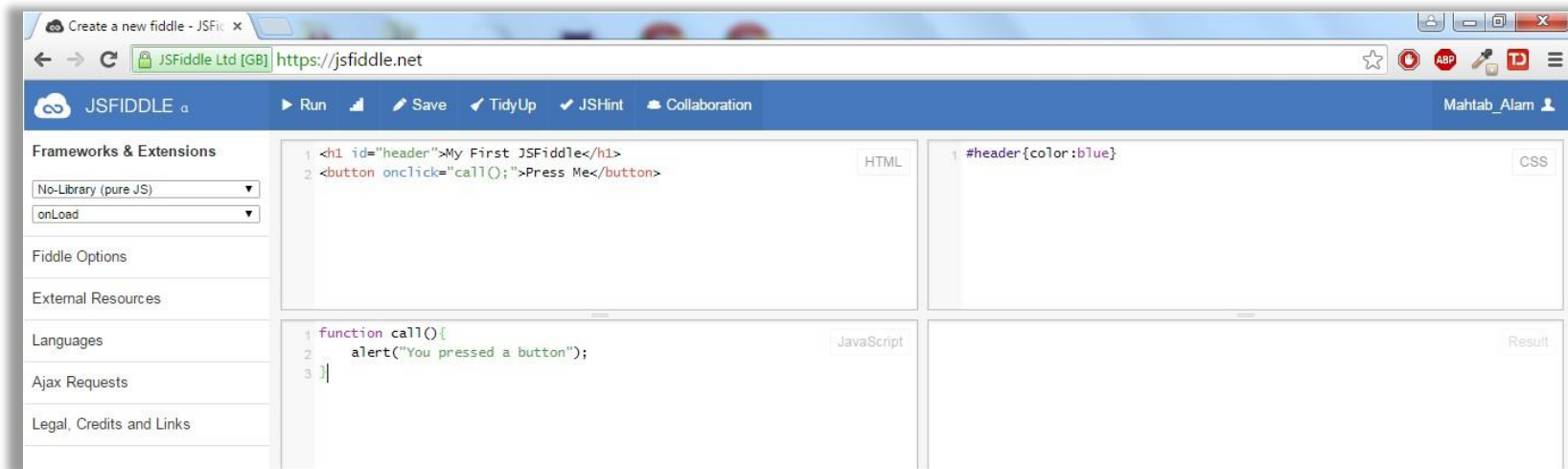
HTML

```
#header{color:blue}
```

CSS

```
function cal(){
  alert("You pressed a button");
}
```

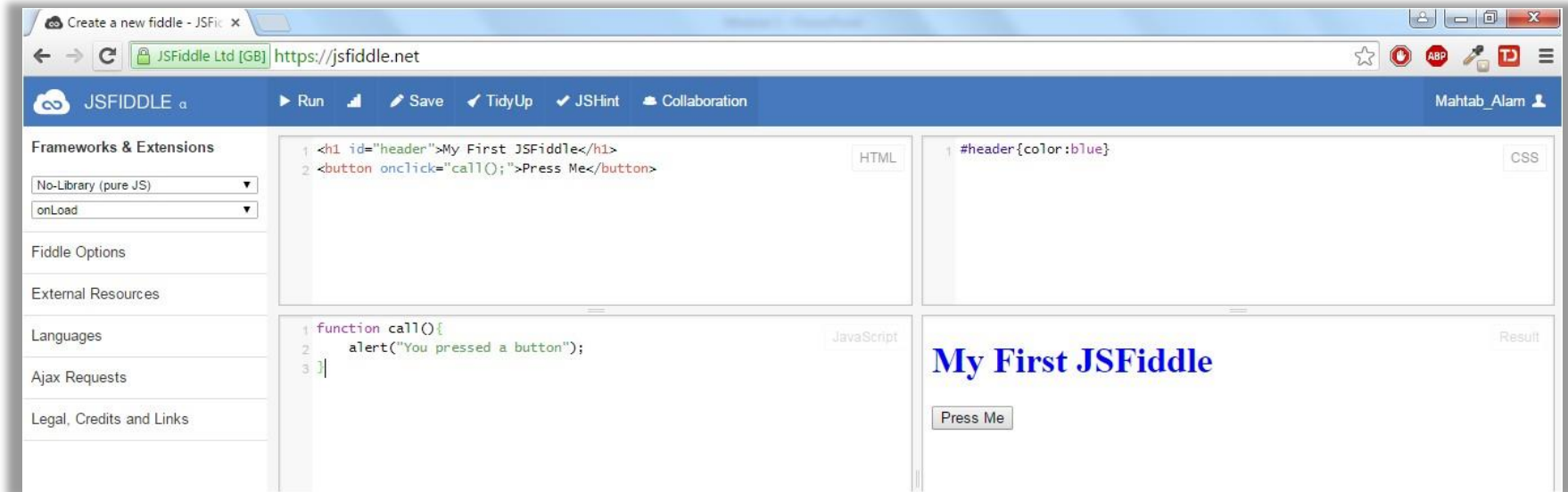
JavaScript





# Running the fiddle

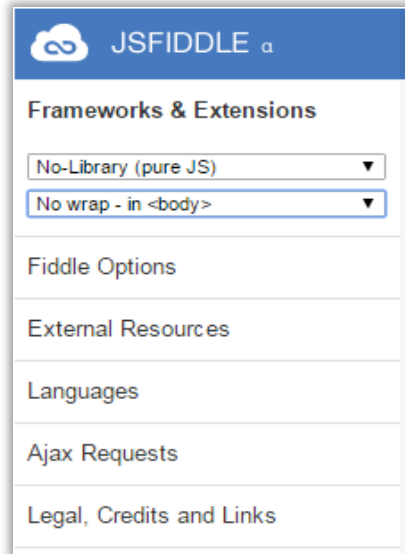
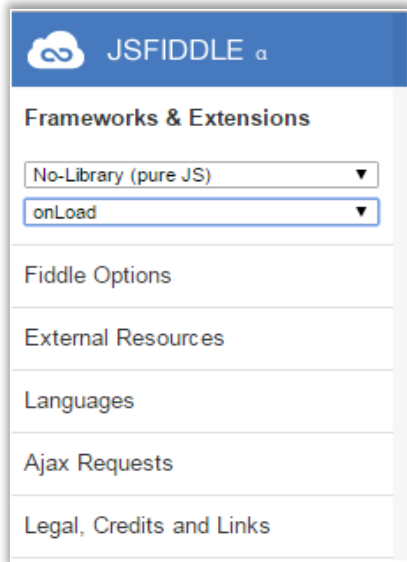
→ Once you are done writing your code click on Run button to execute your fiddle



You can see in above snapshot, we have a h1 tag and a button with CSS which provides blue color to text. On pressing the button it will call the JavaScript function

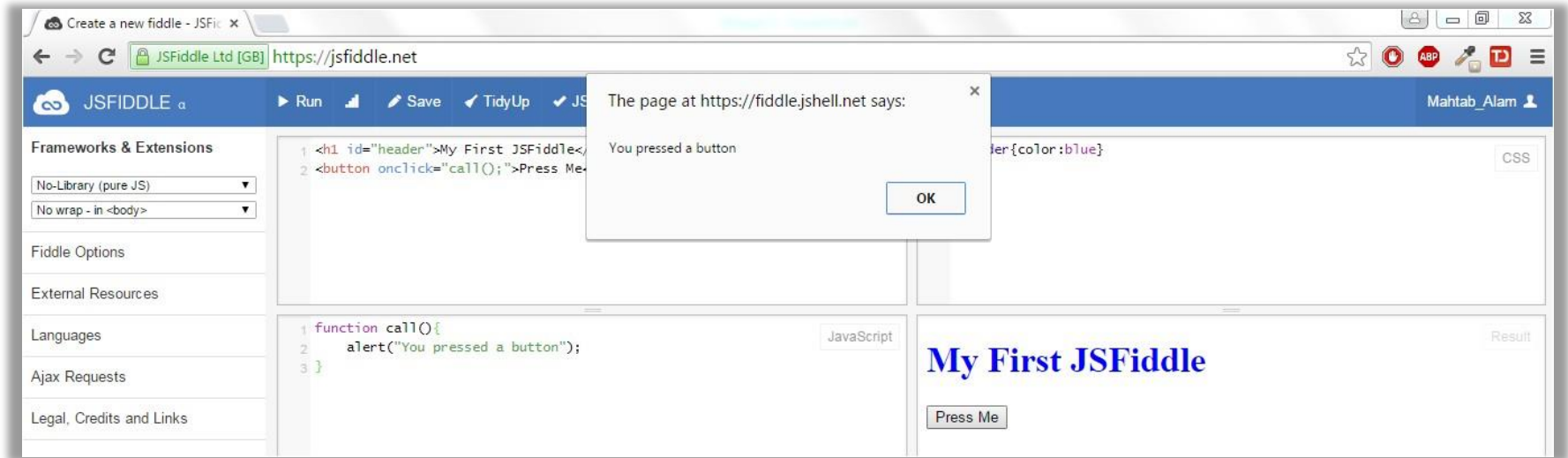
# Making JavaScript Work

- When you press the button you will notice that JavaScript is not executed. This is because, by default JavaScript is executed onLoad. When you select onLoad, your JavaScript is wrapped within onload function
- This means your JavaScript code will run when the page has finished loading, but is no longer available in the global scope. There are couple of workarounds for this problem and the easiest one is, changing the JavaScript **onLoad** to **No wrap – in <body>**



# Making JavaScript Work (Contd.)

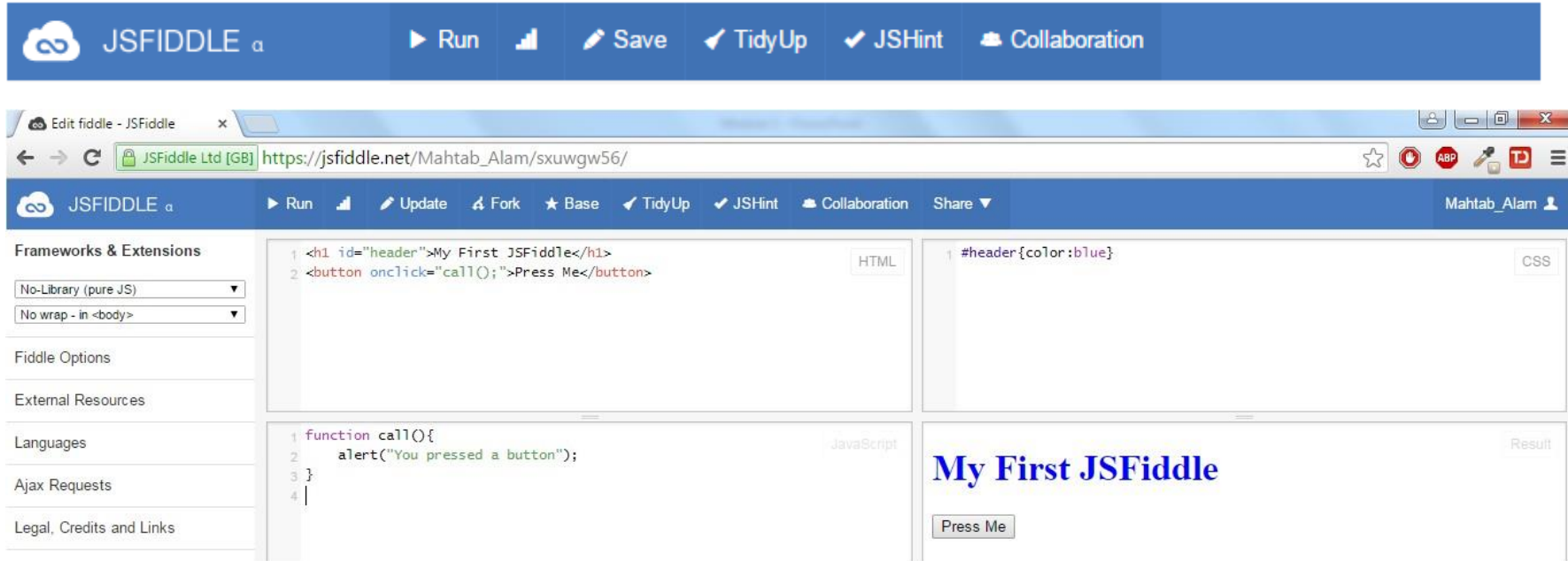
→ After changing the JavaScript from **onLoad** to **No wrap – in <body>**. Click on Run button to execute the code



As shown in the above snapshot now you will be able to execute your JavaScript code

# Sharing your fiddle

→ Now our fiddle is up and running. To share your fiddle with others. Save your fiddle by clicking on Save button



The screenshot displays the JSFiddle web application interface. At the top, a blue navigation bar contains the JSFIDDLE logo and buttons for Run, Save, TidyUp, JSHint, and Collaboration. Below this, a browser window shows the URL [https://jsfiddle.net/Mahtab\\_Alam/sxuwgw56/](https://jsfiddle.net/Mahtab_Alam/sxuwgw56/). The main interface is divided into several sections: a left sidebar with 'Frameworks & Extensions' (set to 'No-Library (pure JS)' and 'No wrap - in <body>'), 'Fiddle Options', 'External Resources', 'Languages', 'Ajax Requests', and 'Legal, Credits and Links'; a central code editor with three panels (HTML, CSS, JavaScript) containing the following code:

```
HTML
1 <h1 id="header">My First JSFiddle</h1>
2 <button onclick="call();">Press Me</button>

CSS
1 #header{color:blue}

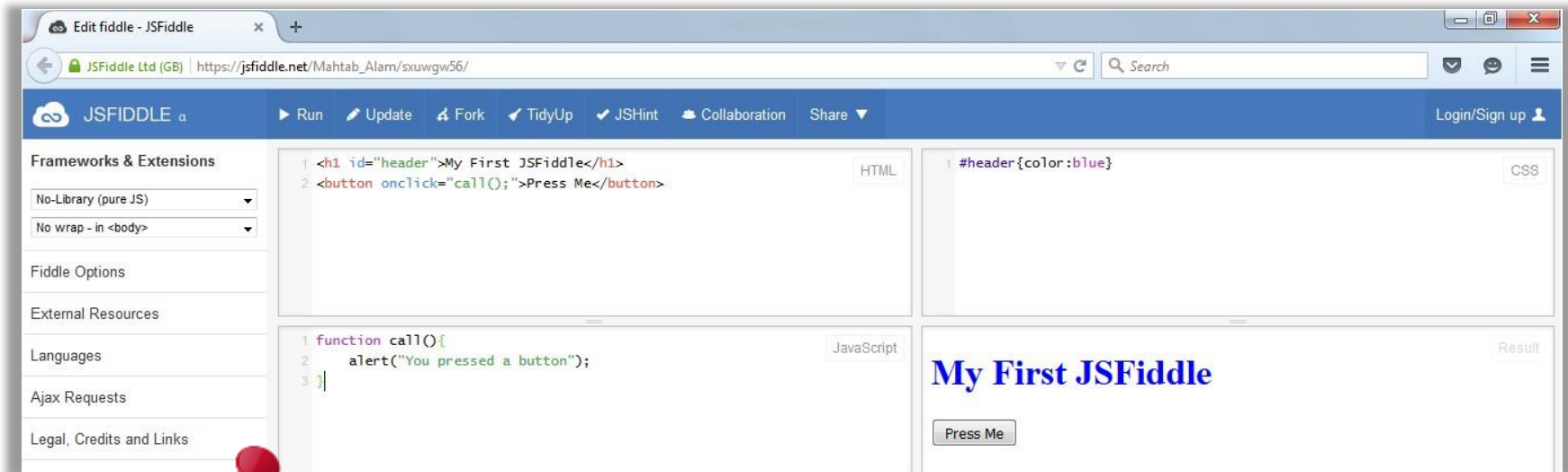
JavaScript
1 function call(){
2   alert("You pressed a button");
3 }
4 |
```

The rightmost panel, labeled 'Result', displays the rendered output: 'My First JSFiddle' in large blue text, with a 'Press Me' button below it.

Once you saved your fiddle you will get a link , in this case it is [https://jsfiddle.net/Mahtab\\_Alam/sxuwgw56/](https://jsfiddle.net/Mahtab_Alam/sxuwgw56/)

# Sharing your fiddle

→ To share your fiddle just share the link [https://jsfiddle.net/Mahtab\\_Alam/sxuwgw56/](https://jsfiddle.net/Mahtab_Alam/sxuwgw56/) and anyone will be able to execute your fiddle

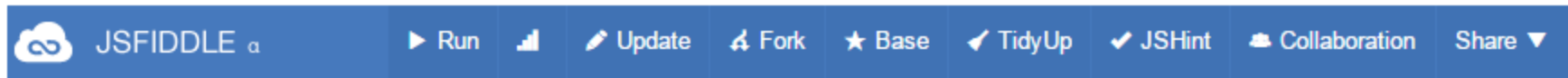


Anyone with a link to your fiddle can execute it even without creating an account at JSFiddle

# Customizing the fiddle

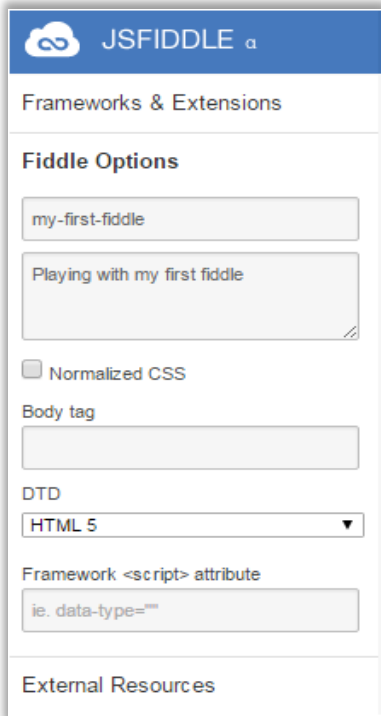
→ Its always a good practice to give your fiddle an appropriate name, so that later on you can easily find your fiddles and edit them if required

**Step 1:** Click on the Base link to make the current version of the fiddle as your base code



# Customizing the fiddle (Contd.)

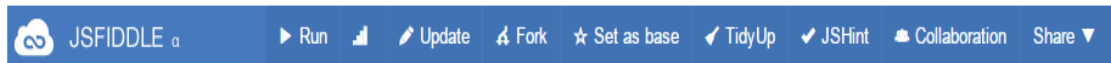
**Step 2:** Click on Fiddle Options (on left pane) , provide a fiddle name and a description. In this example we have named our fiddle as [my-first-fiddle](#) and provided the description as [Playing with my first fiddle](#)



The screenshot shows the 'Fiddle Options' sidebar in the JSFIDDLE interface. It contains the following fields and controls:

- Fiddle Name:** A text input field containing 'my-first-fiddle'.
- Description:** A text area containing 'Playing with my first fiddle'.
- Normalized CSS:** A checkbox that is currently unchecked.
- Body tag:** A text input field.
- DTD:** A dropdown menu currently set to 'HTML 5'.
- Framework <script> attribute:** A text input field containing 'ie. data-type=""'.
- External Resources:** A section header at the bottom of the sidebar.

**Step 3:** After providing the fiddle name and a description click on Update link to update the fiddle



# Customizing the fiddle (Contd.)

**Step 4:** The last step is making the updated fiddle as our Base fiddle. To make the updated fiddle as our base fiddle click on [Set as base](#) link

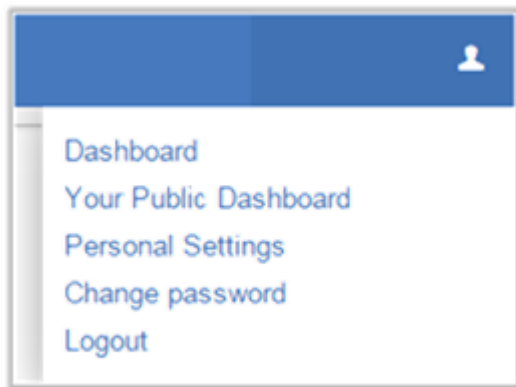


Now we are done with customizing the fiddle and you will be able to see the updated name of your fiddle under your JSFiddle dashboard



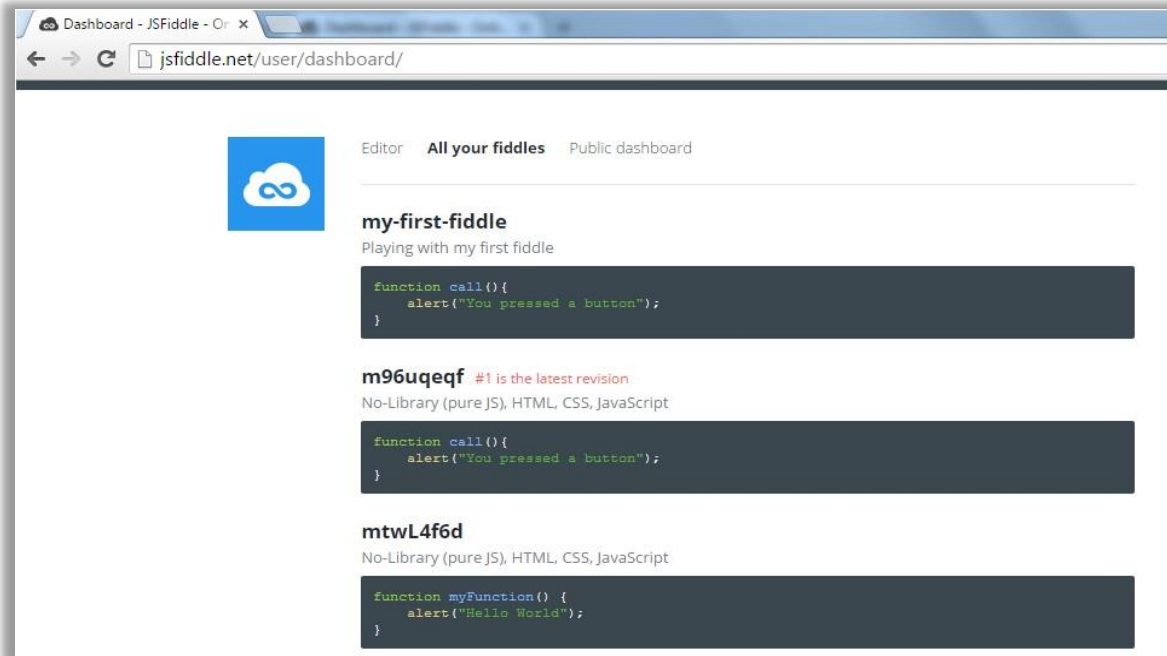
# Accessing the JSFiddle Dashboard

→ To access your [JSFiddle](#) dashboard just click on Dashboard link under your login icon (in upper right corner)



# Accessing the JSFiddle Dashboard

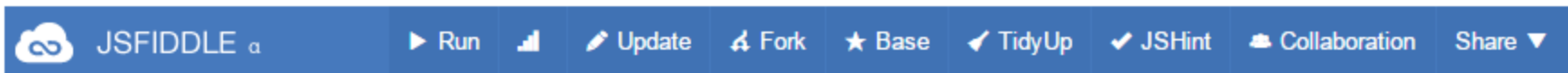
→ Dashboard displays all the fiddles that you created, note that only one of your fiddle have a meaningful name (e.g. [my-first-fiddle](#)) which makes it easy to find the fiddle



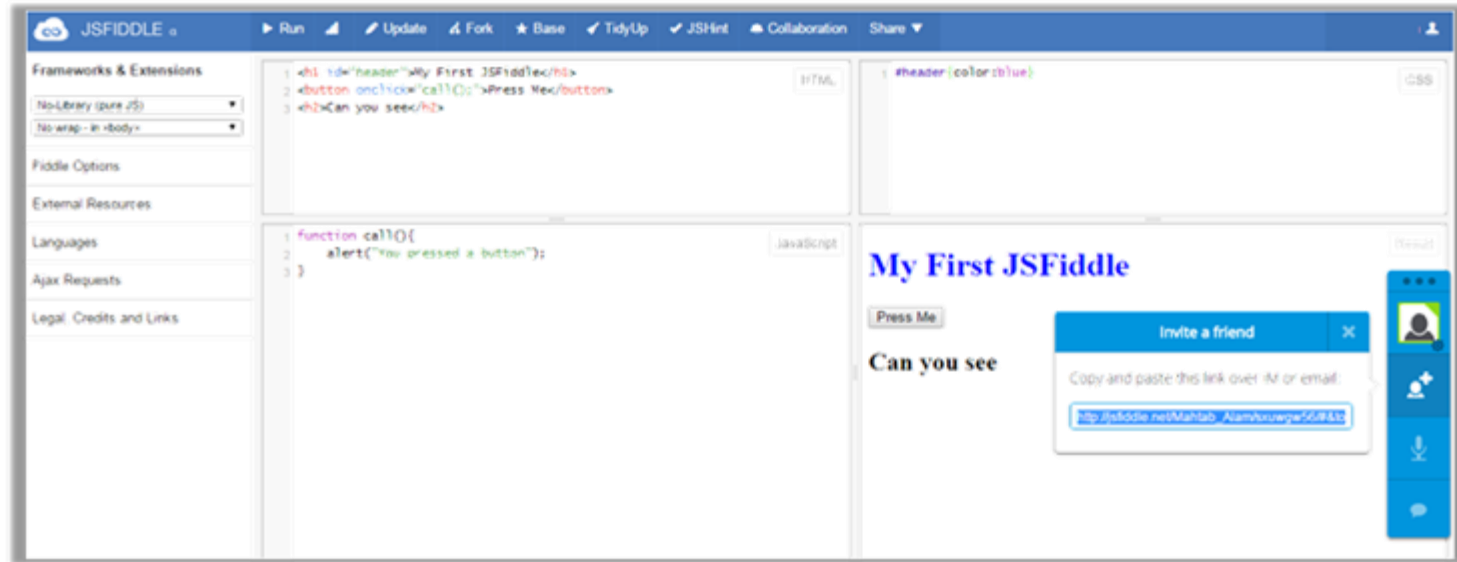
# Collaborative development with JSFiddle

- Another great feature that [JSFiddle](#) provides is collaborative development
- You can share the collaborative development link with your friends and can ask them to work on certain parts of the code
- You can also chat or do a voice call. It really helps in solving the problems and developing applications much faster

To start collaboration just click on Collaboration link which will popup a window and will provide a link which you can share with your friends and start the development collaboratively



# Collaborative development with JSFiddle (Contd.)



Collaborative development – Inviting a friend

# Collaborative development with JSFiddle (Contd.)

The screenshot displays the JSFiddle web application interface. The top navigation bar includes links for Run, Update, Fork, Base, TidyUp, JSHint, Collaboration, and Share. On the left, a sidebar menu lists various options: Frameworks & Extensions, Fiddle Options, External Resources, Languages, Ajax Requests, and Legal, Credits and Links. The main workspace is divided into three panels: HTML, CSS, and JavaScript. The HTML panel contains the following code:

```
<h1 id="header">My First JSFiddle</h1>
<button onclick="call();">Press Me</button>
<h2>Can you see</h2>
```

The CSS panel contains the following code:

```
#header {color:blue}
```

The JavaScript panel contains the following code:

```
function call(){
  alert("You pressed a button");
}
```

Below the JavaScript code, a purple button labeled "Intelligent Iguana" is visible. The right sidebar shows a chat window titled "My First JSFiddle" with a "Press Me" button. The chat window displays a conversation between "Intelligent Iguana & You". The chat history includes:

- me: Can you see (12:33 AM)
- Intelligent Iguana joined the session.
- Intelligent Iguana: change the css style (12:34 AM)






A text input field at the bottom of the chat window is labeled "Type your message here".

Collaborative development - Chat



# QUESTIONS

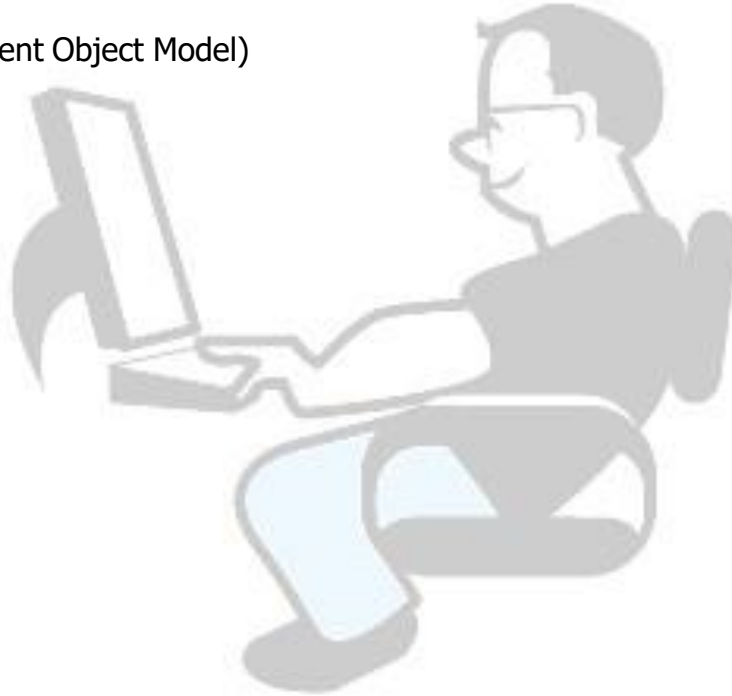


# Assignment

-  Arrays - Write JavaScript to display the names of computer languages and display all of them in an alert box on click of a button. Use Arrays to store the list of computer programming languages
-  Objects – Write JavaScript to store the employee object information (Employee name, employee no, age and dept.) and display as Employee <name> is working in <department> department. Here <name> is name of the employee and <department> is the employee's department
-  Functions - Write a JavaScript function to find largest of 3 numbers and display it
-  Event handlers - Write JavaScript to display an alert message "You have moved the mouse on the button" when the mouse is moved on a button
-  if conditions - Write JavaScript to accept a name in prompt and validate whether name is entered or not. If the name is not entered then display a message "Sorry, you have not entered any name". If the name is entered by the user then display "Hello <entered name>, welcome". Here <entered name> is the name entered by the user

# Pre-work for Next Class

-  Learn about Twitter Bootstrap 3 framework
-  Research on DOM (Document Object Model)





# Further Reading

- <http://conceptf1.blogspot.in/2014/01/data-types-in-javascript.html>
- <http://www.quirksmode.org/js/function.html>
- <http://javascriptweblog.wordpress.com/2010/10/25/understanding-javascript-closures/>
- <http://ericleads.com/2012/12/switch-case-considered-harmful/>



# Agenda for the next class

- In the next module, you will be able to:
- Get started with Twitter Bootstrap 3
- Build websites with Twitter Bootstrap 3



# Survey

---

Your feedback is important to us, be it a compliment, a suggestion or a complaint. It helps us to make the course better!

Please spare few minutes to take the survey after the webinar.

Thank you!

