Web Development (WD) (3151606)

Unit-3

Java Script (JS)

JavaScript

Prof. Mayur Prajapati
Computer Engineering Department
SVBIT, Gandhinagar

Outline

- ✓ Introduction
- ✓ Task Performed by Client side Scripts
- ✓ Pros & Cons of Client side Scripts
- ✓ Client side Scripts V/S Server side Scripts
- ✓ Variables
- ✓ Functions
- ✓ Conditions & Loops
- ✓ Pop up boxes
- ✓ External JavaScript
- ✓ JavaScript Objects
- ✓ DOM
- ✓ DHTML

Introduction

- A **script** is a program or sequence of instructions that is interpreted or carried out by another program rather than by the computer processor.

- A **scripting language** or script language is a programming language that supports scripts, programs written for a special run-time environment that can interpret (rather than compile) and automate the execution of tasks.

- Scripting languages, which can be embedded within HTML, commonly are used to add functionality to a Web page, such as different menu styles or graphic displays or to serve dynamic advertisements.

Introduction

- JavaScript (JS) is a dynamic language.

- It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed.

- It is also used in server-side network programming with runtime environments, game development and the creation of desktop and mobile applications.

Introduction

- For a Web page, HTML supplies document content and structure while CSS provides presentation styling.
- In addition, client-side scripts can control browser actions associated with a Web page.
- Client-side scripts are almost written in the JavaScript language to control browser's actions.
- Client-side scripting can make Web pages more dynamic and more responsive

Introduction

- HTML and CSS concentrate on a static rendering of a page; things do not change on the page over time, or because of events.

- To do these things, we use scripting languages, which allow content to change dynamically.

- Not only this, but it is possible to interact with the user beyond what is possible with HTML.

- Scripts are programs just like any other programming language; they can execute on the client side or the server.

Tasks performed by client-side scripts

- Checking **correctness** of user input
- **Monitoring** user events and **specifying reactions**
- **Replacing** and **updating** parts of a page
- Changing the **style** and **position** of displayed elements **dynamically**
- **Modifying** a page in **response** to **events**
- Getting browser **information**
- Making the Web page **different** depending on the browser and browser features
- **Generating HTML** code for parts of the page

Pros & Cons of Client Side Scripting

- **Pros**
  - Allow for **more interactivity** by immediately responding to users' actions.
  - **Execute quickly** because they do not require a trip to the server.
  - The web **browser** uses its own **resources**, and **eases** the **burden** on the **server**.
  - It **saves** network **bandwidth**.

- **Cons**
  - **Code** is loaded in the browser so it will be **visible** to the client.
  - Code is **modifiable**.
  - **Local files** and **databases cannot** be **accessed**.
  - User is **able** to **disable** client side scripting

Client V/S Server Side Scripting

Server Side Scripting	Client Side Scripting
Server side scripting is used to create dynamic pages based on a number of conditions when the users browser makes a request to the server.	Client side scripting is used when the users browser already has all the code and the page is altered on the basis of the users input.
The Web Server executes the server side scripting that produces the page to be sent to the browser.	The Web Browser executes the client side scripting that resides at the user's computer.
Server side scripting is used to connect to the databases and files that reside on the web server.	Client side scripting cannot be used to connect to the databases and files on the web server.

Client V/S Server Side Scripting

Server Side Scripting	Client Side Scripting
Server resources can be accessed by the server side scripting.	Browser resources can be accessed by the client side scripting.
Server side scripting can’t be blocked by the user.	Client side scripting is possible to be blocked by the user.
Examples of Server side scripting languages : PHP, JSP, ASP, ASP.Net, Ruby, Perl and many more.	Examples of Client side scripting languages : Javascript, VB script, etc.

What is difference between Java script and JAVA?

- Java is a statically typed language; JavaScript is dynamic.
- Java is class-based; JavaScript is prototype-based.
- Java constructors are special functions that can only be called at object creation; JavaScript "constructors" are just standard functions.
- Java requires all non-block statements to end with a semicolon; JavaScript inserts semicolons at the ends of certain lines.
- Java uses block-based scoping; JavaScript uses function-based scoping.
- Java has an implicit this scope for non-static methods, and implicit class scope; JavaScript has implicit global scope

Embedded JavaScript

- JavaScript can be embedded in an HTML document.
- To embed it in HTML you must write:

```
<script type=”text/javascript”>
</script>
```

- The script tag has effect of the stopping the JavaScript being printed out as well as indentifying the code enclosed.
- JavaScript can be embedded in an HTML document.
- The JavaScript can be placed in the head section of your HTML or the body.

Embedded JavaScript

- The Scripts placed in the body section are executed as the page loads and can be used to generate the content of the page.

- As well as the body section, JavaScript can also be placed in the head part.

- The advantages of putting a script in there are that it loads before the main body.

Code

```
<html>
  <head>
    <title>HTML script Tag</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("<h1>This is a heading</h1>");
  </script>
  </body>
</html>
```

External JavaScript

- If you want to use the same script on several pages it could be a good idea to place the code in a separate file, rather than writing it on each.
- That way if you want to update the code, or change it, you only need to do it once.
- Simply take the code you want in a separate file out of your program and save it with the extension .js.

Code

```
<html>
  <head>
    <title>HTML script</title>
  </head>
  <body>
    <script src="myScript.js"></script>
  </body>
</html>
```

External JavaScript

- We can create external JavaScript file and embed it in many html pages.
- It provides code reusability because single JavaScript file can be used in several html pages.
- An external JavaScript file must be saved by **.js** extension.
- To embed the External JavaScript File to HTML we can use ***script*** tag with ***src*** attribute in the head section to specify the path of JavaScript file.
- For Example :

```
<script type="text/javascript" src="message.js"></script>
```

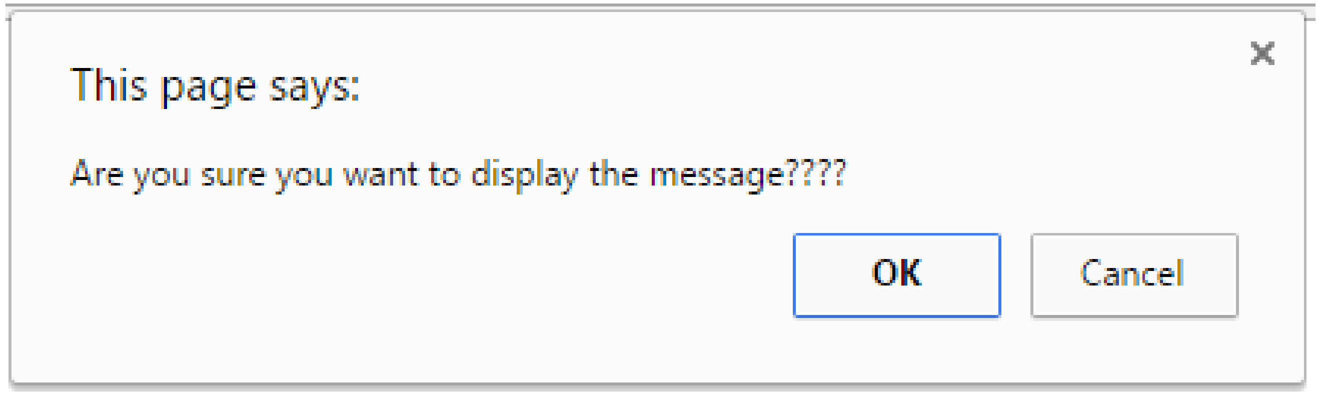
External JavaScript (Example)

message.js

```
function myAlert(msg) {
        if(confirm("Are you sure you want to display the message????")) {
                alert(msg);
        }
        else {
                alert("Message not Displayed as User Cancled Operation");
        }
}
```

myHtml.html

```
<html>
   <head>
      <script src="message.js"></script>
   </head>
   <body>
      <script> myAlert("Hello World"); </script>
   </body>
</html>
```



This page says:

Are you sure you want to display the message????

<script> tag

- The <script> tag is used to define a client-side script (JavaScript).
- The <script> element either contains **scripting statements, or** it points to an **external script** file through the **src** attribute.
- Example :

Code

```
<html>
  <head>
    <title>HTML script Tag</title>
  </head>
  <body>
    <script type="text/javascript">
      // Java Script Code Here
    </script>
  </body>
</html>
```

Code

```
<html>
  <head>
    <title>HTML script Tag</title>
  </head>
  <body>
    <script src=“PathToJS”>
    </script>
  </body>
</html>
```

JavaScript Variables

- Variables in JavaScript behave the same as variables in most popular programming languages (C, C++, etc) do, but in JavaScript you don't have to declare variables before you use them.
- A variable's purpose is to store information so that it can be used later.
- A variable is a symbolic name that represents some data that you set.
- When using a variable for the first time it is not necessary to use "var" before the variable name.
- Variable names must begin with a letter.

JavaScript Variables

- Variable names are case sensitive (y and Y are different variables).

```
var x=5;
var y=6;
var z=x+y;
```

- You can declare many variables in one statement.
- Just start the statement with var and separate the variables by comma:

```
var name="Doe", age=30, job="carpenter"; var name="Doe", age=30, job="carpenter";
```

JavaScript Variables

- Variable declared without a value will have the value undefined.

- If you re-declare a JavaScript variable, it will not lose its value.

- The value of the variable carname will still have the value "Volvo" after the execution of the following two statements.

```
varcarname="Volvo";
varcarname;
```

JavaScript Operators

- Operators in JavaScript are very similar to operators that appear in other programming languages.

- The definition of an operator is a symbol that is used to perform an operation.

- Most often these operations are arithmetic (addition, subtraction, etc), but not always.

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
=	Assignment

JavaScript Arrays

- An **array** is a **collection of data**, each item in array has an index to access it.

- Ways to use array in JavaScript

  -
    ```
    var myArray = new Array();
     myArray[0] = "Mayur";
     myArray[1] = 222;
     myArray[2] = false;
    ```

  -
    ```
    var myArray = new Array("Mayur" , 123 , true);
    ```

JavaScript Array

- An array is a special variable, which can hold more than one value at a time.
- The Array object is used to store multiple values in a single variable.
- An array can be created in three ways.
- The following code creates an Array object called myCars.

- **Regular**
- *varmyCars=new Array(); myCars[0]="Saab"; myCars[1]="Volvo"; myCars[2]="BMW";*

- **Condensed**
- *varmyCars=new Array("Saab","Volvo","BMW");*

JavaScript Array

- **Literal**
- *varmyCars=["Saab","Volvo","BMW"];*

- **Access an Array**
  - You refer to an element in an array by referring to the **index** number.
  - This statement access the value of the first element in myCars.

```
var name=myCars[0];
```

  - This statement modifies the first element in myCars:

```
myCars[0]="Opel";
```

JavaScript Functions

- A JavaScript function is a **block of code** designed to perform a particular task.
- A JavaScript function is **executed** when "**something**" **invokes** it.
- A JavaScript function is defined with the **function** keyword, **followed by a name, followed by parentheses ()**.
- The **parentheses** may **include parameter** names **separated** by **commas**: (parameter1, parameter2, ...)
- The **code to be executed**, by the function, is placed inside **curly brackets**.
- Example :

Code

```
function myFunction(p1, p2) {
        return p1 * p2;
}
```

JavaScript Function

- When JavaScript **reaches a return** statement, the function will **stop executing**.

- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

- The code inside the function will execute when "something" invokes (calls) the function:
  - When an **event occurs** (when a user clicks a button)
  - When it is invoked (**called**) from JavaScript code
  - **Automatically** (self invoked)

JavaScript Function

```
<html>
        <body>
        <script   type=”text/javascript”>
                var z= multXbyY(10,15);
                document.write(“The result is” +z);
                functionmultXbyY(x,y) {
                        document.write(“x is ” +x);
                        document.write(“y is ”+y);
                        return x*y;
                        }
        </script>
</body></html>
```

JavaScript Conditions

- Conditional statements are used to perform different actions based on different conditions.
- In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
- **if...else if...else** statement - use this statement to select one of many blocks of code to be executed.
- **switch statement** - use this statement to select one of many blocks of code to be executed

JavaScript Conditions

- **If...else Statement**
- Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

```
if (condition)
{
        code to be executed if condition is true
}
else
{
        code to be executed if condition is not true
}
```

JavaScript Conditions

- **If...else if...else Statement**

  Use the if....else if...else statement to select one of several blocks of code to be executed.

```
if (condition1)
 {
            code to be executed if condition1 is true
}
else if (condition2) {
code to be executed if condition2 is true
}
else {
            code to be executed if neither condition1 nor condition2 is true
            }
```

JavaScript Conditions

- **Switch Statement**
  - Use the switch statement to select one of many blocks of code to be executed.

```
switch(n)
{
case 1:
        execute code block 1 break;
case 2:
        execute code block 2 break;
default:
code to be executed if n is different from case 1 and 2
}
```

- **The *default* Keyword**
  - Use the *default* keyword to specify what to do if there is no match.

Conditions

If condition

```
if(a>10)
 {
          alert("A is > that 10");
 }
```

ternary operator

```
max = a>b ? a : b ;
```

switch

```
switch(expression)
{
    case lbl1:
          // code to execute
           break;
    case lbl2:
          // code to execute
          break;
}
```

Loops

for loop

Use when you know how many repetitions you want to do

syntax

```
for(initialize ; condition ; increment) { … }
```

example

```
for(x=0;x<10;x++) {
   // Code Here
}
```

while loop

Loops through block of code while condition is true

syntax

```
while(condition) { … }
```

example

```
while (x<10) {
    //Code Here
}
```

do while loop

Execute block at least once then repeat while condition is true

syntax

```
do{ … } while (condition);
```

example

```
do{
   // Code Here
} while (x<10)
```

Strings

- A **string** can be defined as a **sequence of letters, digits, punctuation and so on**.

- A **string** in a JavaScript is **wrapped** with **single or double quotes.**

- Strings can be **joined** together with the **+ operator**, which is called **concatenation**.

  For Example,

  ```
      mystring = “my college name is ” + “SVBIT”;
  ```

- As string is an object type it also has some useful features.

  For Example,

  ```
      lenStr = mystring.length;
  ```

  Which returns the **length** of the **string** in **integer**

Strings

- There are also number of methods available for string.

Method	Description
charAt	Returns the character at a specific index
indexOf	Find the first index of a character
lastIndexOf	Find the last index of a character
substring / substr	Return a section of a string.
replace	Replaces a specified value with another value in a string
toLowerCase	Convert a string to lower case.
toUpperCase	Convert a string to upper case.

Strings

- An **escape sequence** is a sequence of characters that **does not represent itself** when used inside a character or string, **but** is **translated into another character** or a **sequence of characters** that may be difficult or impossible to represent directly.
- Some Useful Escape sequences :

Sequence	Character
\t	Tab
\n	Newline
\r	Carriage return
\”	Double Quote
\’	Single Quote
\\	Backslash

Pop up Boxes

- Popup boxes can be used to raise an alert, or to get confirmation on any input or to have a kind of input from the users.

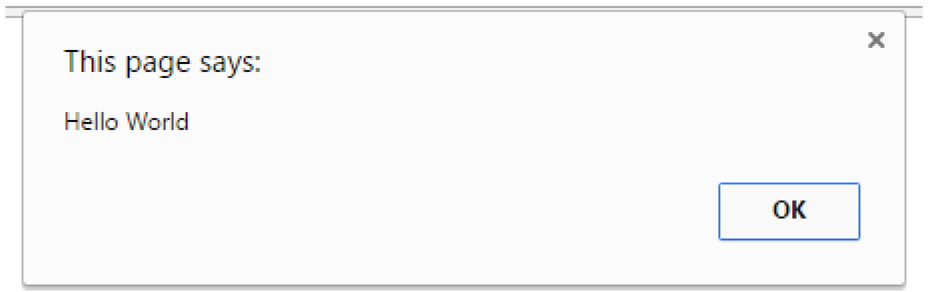
- JavaScript supports **three** types of popup boxes.
  - Alert box
  - Confirm box
  - Prompt box

Alert Box

- An **alert box** is used if you want to **make sure** information **comes through** to the **user**.
- When an alert box pops up, the user will **have to click "OK"** to proceed.
- It can be used to display the result of validation.

Code

```
<html>
    <head>
            <title>Alert Box</title>
    </head>
    <body>
            <script>
                    alert("Hello World");
            </script>
    </body>
</html>
```



This page says:

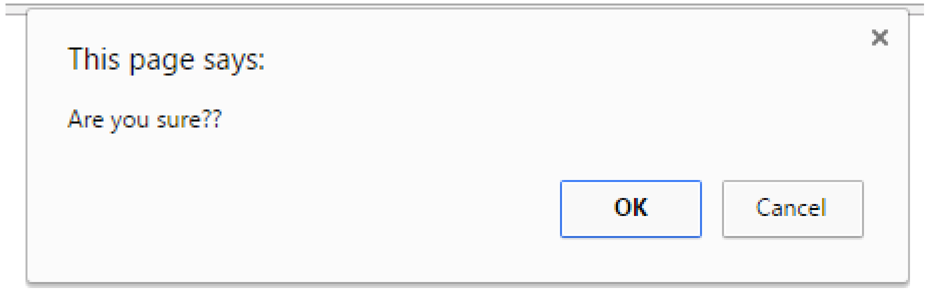
Hello World

Confirm Box

- A **confirm box** is used if you want the user to **accept something**.
- **When** a confirm box **pops up**, the user will have to **click** either "**OK**" or "**Cancel**" to proceed, If the user clicks "**OK**", the box **returns true**.
- If the user clicks "**Cancel**", the box **returns false**.
- Example :

Code

```
<script>
  var a = confirm("Are you sure??");
  if(a==true) {
            alert("User Accepted");
  }
  else  {
            alert("User Cancled");
  }
</script>
```


This page says:

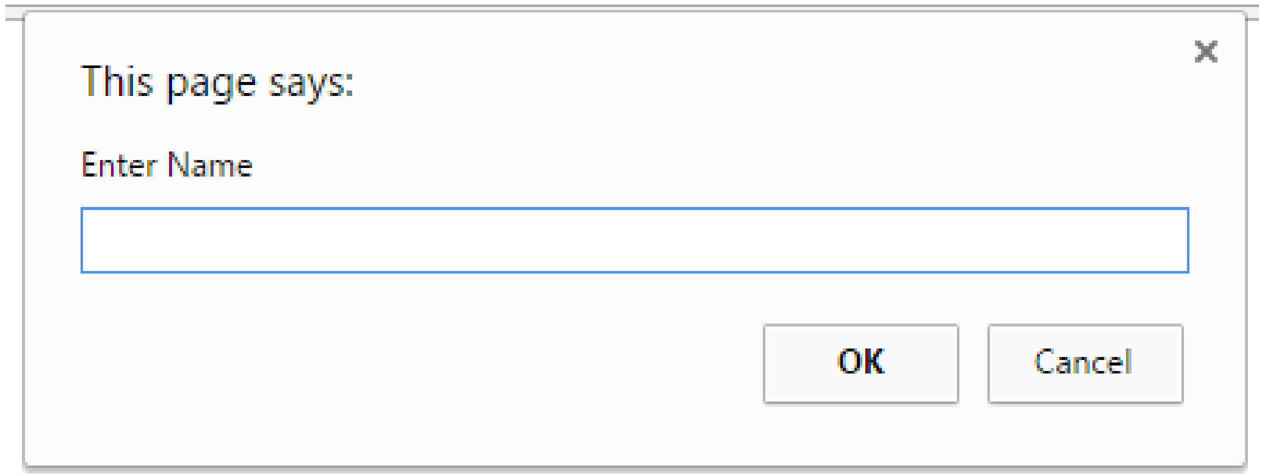
Are you sure??

Prompt Box

- A **prompt box** is used if you want the user to **input a value**.
- **When** a prompt box **pops up**, user have to click either "**OK**" or "**Cancel**" to proceed,
- If the user clicks "**OK**" the box **returns the input value**, If the user clicks "**Cancel**" the box **returns null**.

Code

```
<script>

   var a = prompt(“Enter Name");

   alert(“User Entered ” + a);

</script>
```


This page says:

Enter Name

JavaScript Objects

- An object is just a special kind of data, with properties and methods.

- Accessing Object Properties
  - Properties are the values associated with an object.
  - The syntax for accessing the property of an object is below

    objectName.propertyName

  - This example uses the length property of the Javascript's inbuilt object(String) to find the length of a string:

    var message="Hello World!";
    var x=message.length;

JavaScript Objects

- Accessing Object Methods
  - Methods are the actions that can be performed on objects.
  - You can call a method with the following syntax.

    objectName.methodName()

  - This example uses the toUpperCase method of the String object to convert string to upper case:

```
var message="Hello World!";
var x=message.toUpperCase();
```

JavaScript’s inbuilt Objects

- JavaScript comes with some inbuilt objects which are,
  - String
  - Date
  - Array
  - Boolean
  - Math
  - RegExp

  etc….

Math Object in JavaScript

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical constants and methods.
- Example for using properties/methods of Math:

Code

```
<script>
                var x=Math.PI;

               var y=Math.sqrt(16);

</script>c
```

Math Object (Cont.)

- Math object has some properties which are,

Properties	Description
E	Returns Euler's number(approx.2.718)
LN2	Returns the natural logarithm of 2 (approx.0.693)
LN10	Returns the natural logarithm of 10 (approx.2.302)
LOG2E	Returns the base-2 logarithm of E (approx.1.442)
LOG10E	Returns the base-10 logarithm of E (approx.0.434)
PI	Returns PI(approx.3.14)
SQRT1_2	Returns square root of ½
SQRT2	Returns square root of 2

Math Methods (Cont.)

Method	Description
abs(x)	Returns the absolute value of x
sin(x)	Returns the sine of x (x is in radians)
cos(x)	Returns the cosine of x (x is in radians)
tan(x)	Returns the tan of x (x is in radians)
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value
atan2(x)	Returns arctangent of x
random(x)	Returns random floating number between 0 to 1

Method	Description
exp(x)	Returns the value of Ex
ceil(x)	Returns x, rounded upwards to the nearest integer
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm(base E) of x
round(x)	Rounds x to the nearest integer
pow(x,y)	Returns the value of x to the power of y
max(x,y, z,...,n)	Returns the number with the highest value
sqrt(x)	Returns the square root of x

User Defined Objects

- JavaScript allows you to create your own objects.
- The first step is to use the new operator.

```
var myObj= new Object();
```

- This creates an empty object.
- This can then be used to start a new object that you can then give new properties and methods.
- In object- oriented programming such a new object is usually given a constructor to initialize values when it is first created.

User Defined Objects

- However, it is also possible to assign values when it is made with literal values.

example

```
<script>
        person={
                firstname: “Mayur",
                lastname: “Prajapati",
                age: 50,
                eyecolor: "blue"
        }
        alert(person.firstname person.lastname + " is "  + person.age + " years old.");
 </script>
```

Document Object Model (DOM)

- The Document Object Model is a platform and language neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.

- The **window** object is the primary point from which most other objects come.

- From the current window object **access** and **control** can be given to most aspects of the **browser features** and the **HTML document**.

- When we write :

```
document.write(“Hello World”);
```

- We are actually writing :

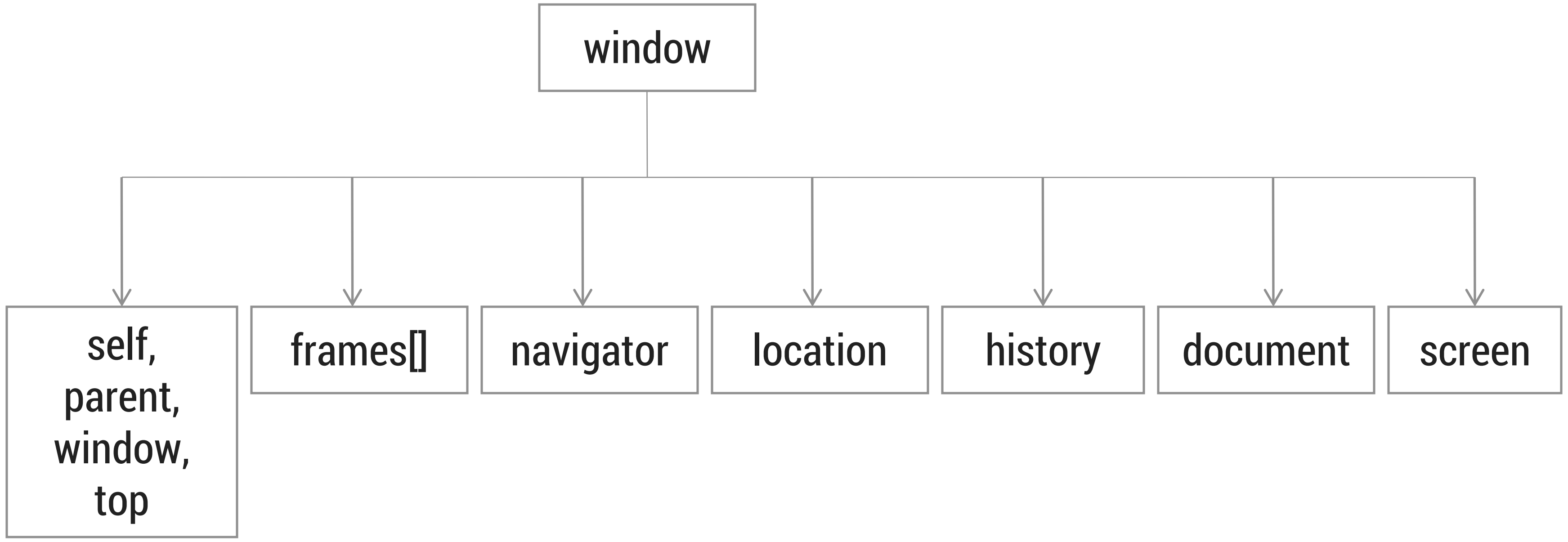
```
window.document.write(“Hello World”);
```

The **window** is just there by default

DOM (Cont)

- This **window** object represents the window or frame that displays the document and is the global object in client side programming for JavaScript.

- All the client side objects are connected to the window object.



Document Object Properties

Property	Description
anchors	Returns a collection of all the anchors in the document
applets	Returns a collection of all the applets in the document
body	Returns the body element of the document
cookie	Returns all name/value pairs of cookies in the document
domain	Returns the domain name of the server that loaded the document
forms	Returns a collection of all the forms in the document
images	Returns a collection of all the images in the document
links	Returns a collection of all the links in the document (CSSs)
referrer	Returns the URL of the document that loaded the current document
title	Sets or returns the title of the document
URL	Returns the full URL of the document

Document Object Methods

Method	Description
write()	Writes HTML expressions or JavaScript code to a document
writeln()	Same as write(), but adds a newline character after each statement
open()	Opens an output stream to collect the output from document.write() or document.writeln()
close()	Closes the output stream previously opened with document.open()
getElementById()	Accesses element with a specified id
getElementsByName()	Accesses all elements with a specified name
getElementsByTagName()	Accesses all elements with a specified tag name
setTimeout(), clearTimeout()	Set a time period for calling a function once; or cancel it.

getElementById()

- When we suppose to get the reference of the element from HTML in JavaScript using id specified in the HTML we can use this method.
- Example :

HTML

```
<html>
   <body>
      <input type="text" id="myText">
   </body>
</html>
```

JavaScript

```
<script>
   function myFunction()
   {
      var txt = document.getElementById("myText");
      alert(txt.value);
   }
</script>
```

getElementsByTagName()

- When we suppose to get the reference of the elements from HTML in JavaScript using name of the tag specified in the HTML we can use this method.
- It will return the array of elements with the provided tag name.
- Example :

HTML

```
<html>
   <body>
      <input type="text" name="uname">
      <input type="text" name="pword">
   </body>
</html>
```

JavaScript

```
<script>
function myFunction() {
  a=document.getElementsByTagName("input");
  alert(a[0].value);
  alert(a[1].value);
}
</script>
```

Forms using DOM

- We can access the elements of form in DOM quite easily using the name/id of the form.
- Example :

HTML

```
<html>
  <body>
   <form name="myForm">
     <input type="text" name="uname">
     <input type="text" name="pword">
     <input type="button" onClick="f()">
    </form>
  </body>
</html>
```

JS

```
function f()
{
   var a = document.forms["myForm"];
   var u = a.uname.value;
   var p = a.pword.value;
   if(u=="admin" && p=="123")
   {
      alert("valid");
   }
   else
   {
      alert("Invalid");
   }
}
```

Validation

- Validation is the process of **checking** data against a **standard** or **requirement**.
- Form validation normally used to occur at the server, after client entered necessary data and then pressed the Submit button.
- If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a lengthy process which used to put a lot of burden on the server.
- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server.

Validation (Cont.)

Form validation generally performs two functions.

1. **Basic Validation**
   - Emptiness
   - Confirm Password
   - Length Validation etc……

2. **Data Format Validation**

   Secondly, the data that is entered must be checked for correct **form** and **value**.
   - Email Validation
   - Mobile Number Validation
   - Enrollment Number Validation etc….

Validation using RegExp

- A regular expression is an object that describes a pattern of characters.
- Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.
- example:

```
var pattern = "^ [  w]$";   // will allow only words in the string
var regex = new RegExp(pattern);
If(regex.test(testString)){
    //Valid
} else {
    //Invalid
}
```

RegExp (Cont.) (Metacharacters)

- To find **word** characters in the string we can use **w**
  - We can also use [a-z], [A-Z] or [a-zA-Z] for the same
- To find **non-word** characters in the string we can use **W**
- to find **digit** characters in the string we can use **d**
  - We can also use [0-9] for the same
- To find **non-digit** characters in the string we can use **D**
- We can use **n** for **new line** and **t** for tab

RegExp (Cont.) (Quantifiers)

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
n{X}	Matches any string that contains a sequence of *X* *n*'s
n{X,Y}	Matches any string that contains a sequence of X to Y *n*'s
n{X,}	Matches any string that contains a sequence of at least X *n*'s

Email Validation Using RegExp

JavaScript

```
<script>
        function checkMail()
        {
                var a = document.getElementById("myText").value;
                var pattern ="^[\\w-_\.]*[\\w-_\.]\@[\\w]\.+[\\w]+[\\w]$”;
                var regex = new RegExp(pattern);
                if(regex.test(a))
                {
                        alert("Valid");
                }
                else
                {
                        alert("Invalid");
                }
        }
</script>
```

DHTML – Combining HTML,CSS & JS

- DHTML, or Dynamic HTML, is really just a combination of HTML, JavaScript and CSS.
- The main problem with DHTML, which was introduced in the 4.0 series of browsers, is **compatibility**.
- The main focus generally when speaking of DHTML is animation and other such dynamic effects.

DHTML (Cont)

- We can obtain reference of any HTML or CSS element in JavaSCript using below 3 methods.
  1. document.getElementById(“IdOfElement”)
  2. document.getElementsByName(“NameOfElement”)
  3. document.getElementsByTagName(“TagName”)

- After obtaining the reference of the element you can change the attributes of the same using reference.attribute syntax.

- For Example :

HTML Code

```
<img src=“abc.jpg” id=“myImg”>
```

JS Code

```
<script>
  var a = document.getElementById(‘myImg’);
  a.src  = “xyz.jpg”;
</script>
```

DHTML (Cont) (Example)

JavaScript

```
<html>
    <body>
        <div id="myDiv">
            Red Alert !!!!!!
        </div>
        <script>
          var objDiv = document.getElementById("myDiv");
          var colors = ['white','yellow','orange','red'];
          var nextColor = 0;
     setInterval("objDiv.style.backgroundColor = colors[nextColor++%colors.length];",500);
        </script>
    </body>
</html>
```

HTML Element Properties

Event	Description
className	Sets or returns the class attribute of an element
id	Sets or returns the id of an element
innerHTML	Sets or returns the HTML contents (+text) of an element
style	Sets or returns the style attribute of an element
tabIndex	Sets or returns the tab order of an element
title	Sets or returns the title attribute of an element
value	Sets or returns the value attribute of an element

Mouse Events

Event	Attribute	Description
click	onclick	The event occurs when the user clicks on an element
dblclick	ondblclick	The event occurs when the user double-clicks on an element
mousedown	onmousedown	The event occurs when a user presses a mouse button over an element
mousemove	onmousemove	The event occurs when a user moves the mouse pointer over an element
mouseover	onmouseover	The event occurs when a user mouse over an element
mouseout	onmouseout	The event occurs when a user moves the mouse pointer out of an element
mouseup	onmouseup	The event occurs when a user releases a mouse button over an element

Keyboard Events

Event	Attribute	Description
keydown	onkeydown	The event occurs when the user is pressing a key or holding down a key
keypress	onkeypress	The event occurs when the user is pressing a key or holding down a key
keyup	onkeyup	The event occurs when a keyboard key is released

Frame/Object Events

Event	Attribute	Description
abort	onabort	The event occurs when an image is stopped from loading before completely loaded (for <object>)
error	onerror	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)
load	onload	The event occurs when a document, frameset, or <object> has been loaded
resize	onresize	The event occurs when a document view is resized
scroll	onscroll	The event occurs when a document view is scrolled
unload	onunload	The event occurs when a document is removed from a window or frame (for <body> and <frameset>)

Form Events

Event	Attribute	Description
blur	onblur	The event occurs when a form element loses focus
change	onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
focus	onfocus	The event occurs when an element gets focus (for <label>, <input>, <select>, textarea>, and <button>)
reset	onreset	The event occurs when a form is reset
select	onselect	The event occurs when a user selects some text (for <input> and <textarea>)
submit	onsubmit	The event occurs when a form is submitted

Callbacks in Javascript

- A callback is a function passed as an argument to another function.
- This technique allows a function to call another function.
- A callback function can run after another function has finished.

Callbacks in Javascript

- **Function Sequence:**

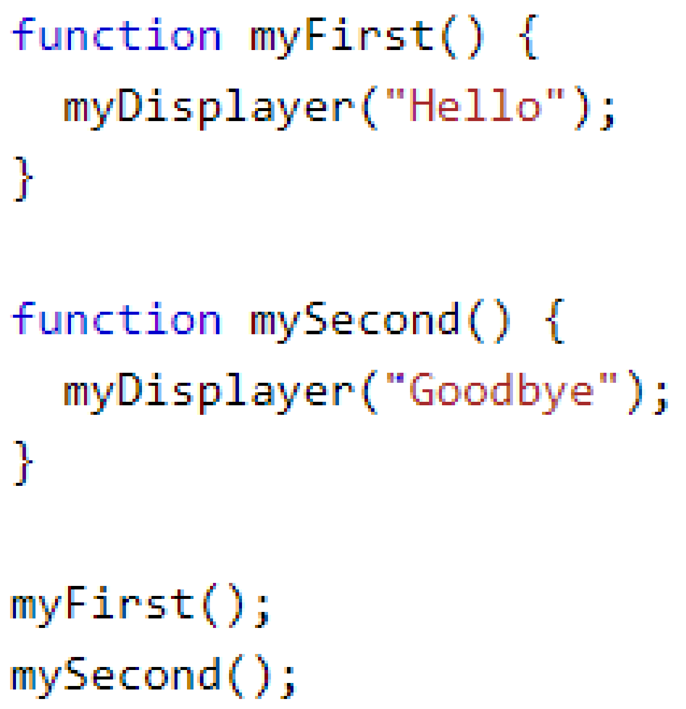
- JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined.
- This example will end up displaying "Goodbye":

```
function myFirst() {
  myDisplayer("Hello");
}

function mySecond() {
  myDisplayer("Goodbye");
}

myFirst();
mySecond();
```

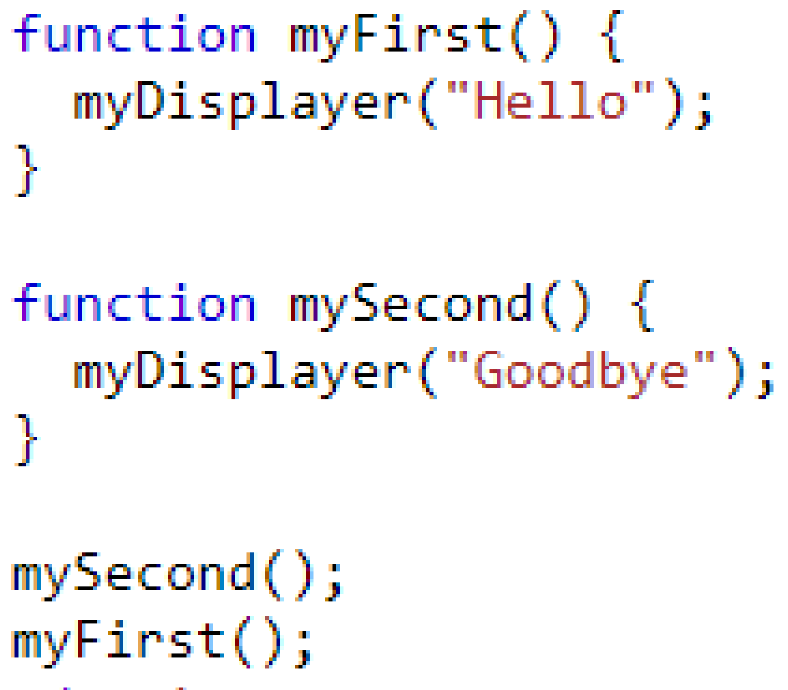
Callbacks in Javascript

- **Function Sequence:**

- JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined.
- This example 1 will end up displaying "Goodbye“ and example 2 will end up displaying "Hello":



```
function myFirst() {
  myDisplayer("Hello");
}

function mySecond() {
  myDisplayer("Goodbye");
}

myFirst();
mySecond();
```



```
function myFirst() {
  myDisplayer("Hello");
}

function mySecond() {
  myDisplayer("Goodbye");
}

mySecond();
myFirst();
```

Callbacks in Javascript

- **Sequence Control**
- Sometimes you would like to have better control over when to execute a function.
- Suppose you want to do a calculation, and then display the result.
- You could call a calculator function (myCalculator), save the result, and then call another function (myDisplayer) to display the result:

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2) {
  let sum = num1 + num2;
  return sum;
}

let result = myCalculator(5, 5);
myDisplayer(result);
```

Callbacks in Javascript

- **Sequence Control**
- Or, you could call a calculator function (myCalculator), and let the calculator function call the display function (myDisplayer):

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2) {
  let sum = num1 + num2;
  myDisplayer(sum);
}

myCalculator(5, 5);
```

Callbacks in Javascript

- The problem with the first example above, is that you have to call two functions to display the result.
- The problem with the second example, is that you cannot prevent the calculator function from displaying the result.
- Now it is time to bring in a callback.

Callbacks in Javascript

- A callback is a function passed as an argument to another function.
- Using a callback, you could call the calculator function (myCalculator) with a callback, and let the calculator function run the callback after the calculation is finished:

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```

Callbacks in Javascript

- In the example above, myDisplayer is the name of a function.
- It is passed to myCalculator() as an argument.
- When you pass a function as an argument, remember not to use parenthesis.
- Right: myCalculator(5, 5, myDisplayer);
- Wrong: myCalculator(5, 5, myDisplayer());

Function as arguments in Javascript

- **Function Parameters and Arguments**

Earlier in this tutorial, you learned that functions can have **parameters**:

```
function functionName(parameter1, parameter2, parameter3)
{
// code to be executed
}
```

- Function **parameters** are the **names** listed in the function definition.
- Function **arguments** are the real **values** passed to (and received by) the function.

Function as arguments in Javascript

- **The Arguments Object**

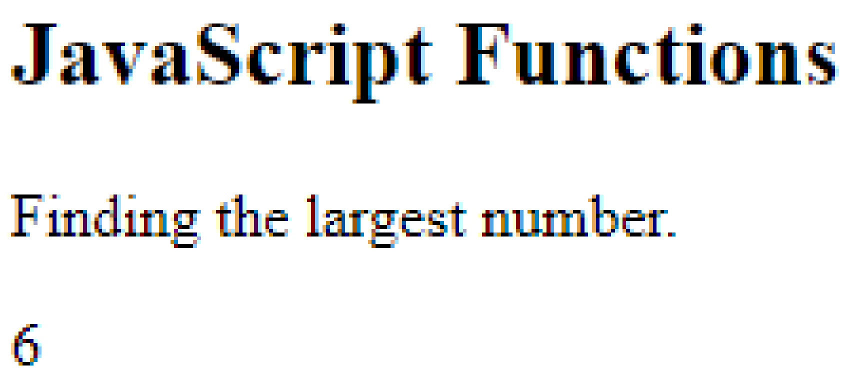
- JavaScript functions have a built-in object called the arguments object.

- The argument object contains an array of the arguments used when the function was called (invoked).

- This way you can simply use a function to find (for instance) the highest value in a list of numbers:

Function as arguments in Javascript

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>
<p>Finding the largest number.</p>
<p id="demo"></p>

<script>
function findMax() {
  let max = -Infinity;
  for(let i = 0; i < arguments.length; i++) {
    if (arguments[i] > max) {
      max = arguments[i];
    }
  }
  return max;
}
document.getElementById("demo").innerHTML = findMax(4, 5, 6);
</script>

</body>
</html>
```



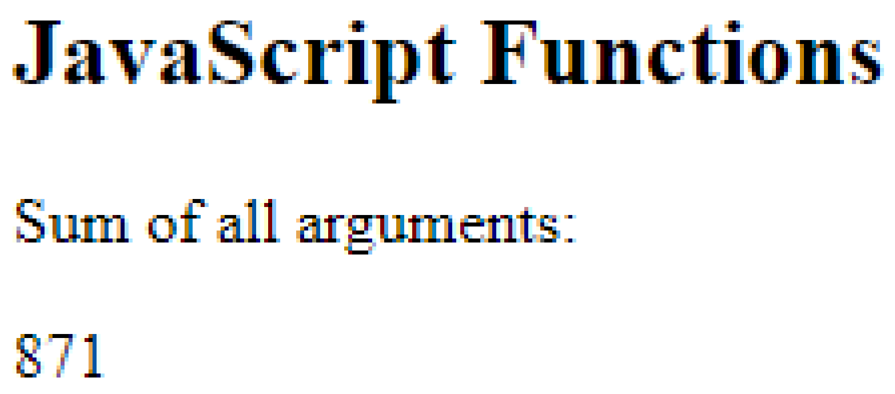
JavaScript Functions

Finding the largest number.

6

Function as arguments in Javascript

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>
<p>Sum of all arguments:</p>
<p id="demo"></p>

<script>
function sumAll() {
  let sum = 0;
  for(let i = 0; i < arguments.length; i++) {
    sum += arguments[i];
  }
  return sum;
}
document.getElementById("demo").innerHTML = sumAll(1, 123, 500, 115, 44, 88);
</script>

</body>
</html>
```



JavaScript Functions

Sum of all arguments:

871

JSON

- JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. Conventions used by JSON are known to programmers, which include C, C++, Java, Python, Perl, etc.

- JSON stands for JavaScript Object Notation.
- The format was specified by Douglas Crockford.
- It was designed for human-readable data interchange.
- It has been extended from the JavaScript scripting language.
- The filename extension is **.json**.
- JSON Internet Media type is **application/json**.
- The Uniform Type Identifier is public.json.

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Characteristics of JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

JSON - Syntax

- JSON syntax is basically considered as a subset of JavaScript syntax; it includes the following:
- Data is represented in name/value pairs.
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).
- Square brackets hold arrays and values are separated by ,(comma).

JSON - Syntax

Below is a simple example –

```
{
   "book": [

      {
         "id":"01",
         "language": "Java",
         "edition": "third",
         "author": "Herbert Schildt"
      },

      {
         "id":"07",
         "language": "C++",
         "edition": "second",
         "author": "E.Balagurusamy"
      }
   ]
}
```

JSON - Syntax

- JSON supports the following two data structures –

- **Collection of name/value pairs** – This Data Structure is supported by different programming languages.

- **Ordered list of values** – It includes array, list, vector or sequence etc.

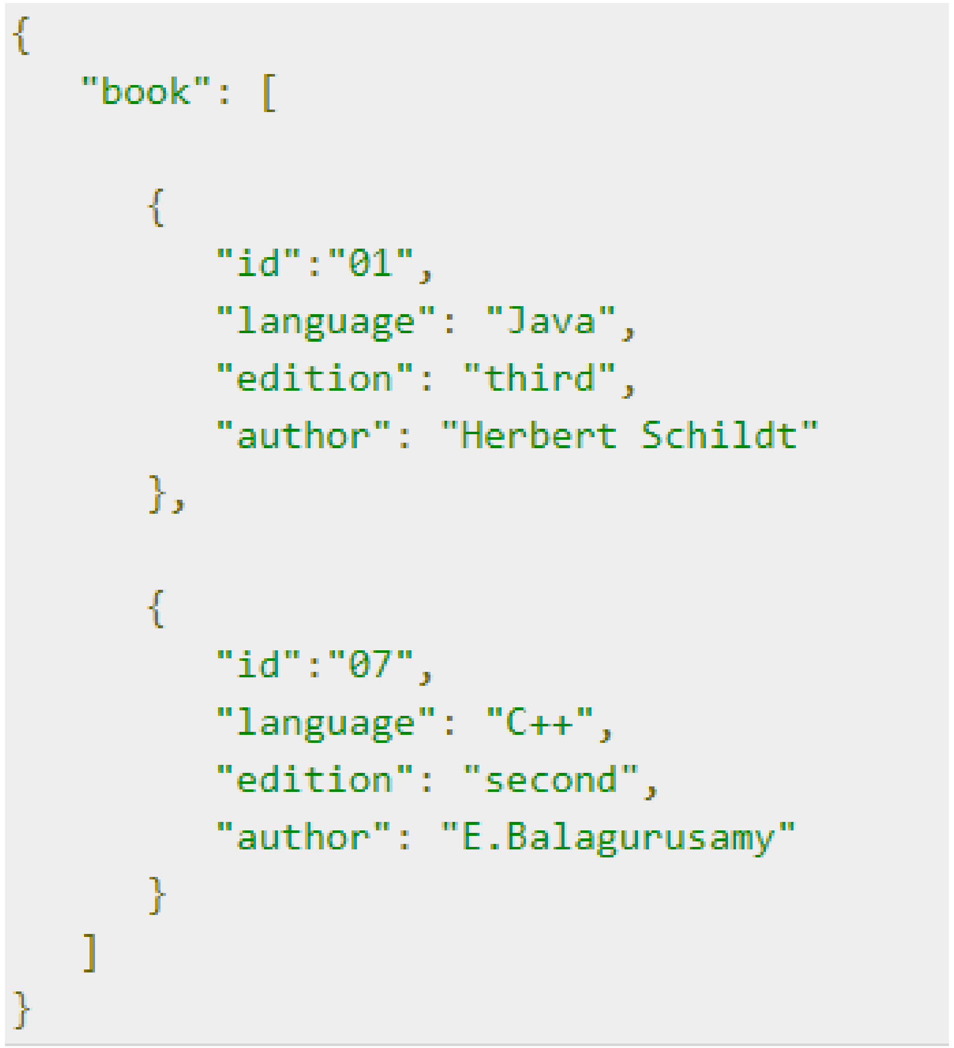
JSON - DataTypes

- JSON format supports the following data types –

Sr.No.	Type & Description
1	**Number** double- precision floating-point format in JavaScript
2	**String** double-quoted Unicode with backslash escaping
3	**Boolean** true or false
4	**Array** an ordered sequence of values
5	**Value** it can be a string, a number, true or false, null etc
6	**Object** an unordered collection of key:value pairs
7	**Whitespace** can be used between any pair of tokens
8	**null** empty

Simple Example in JSON

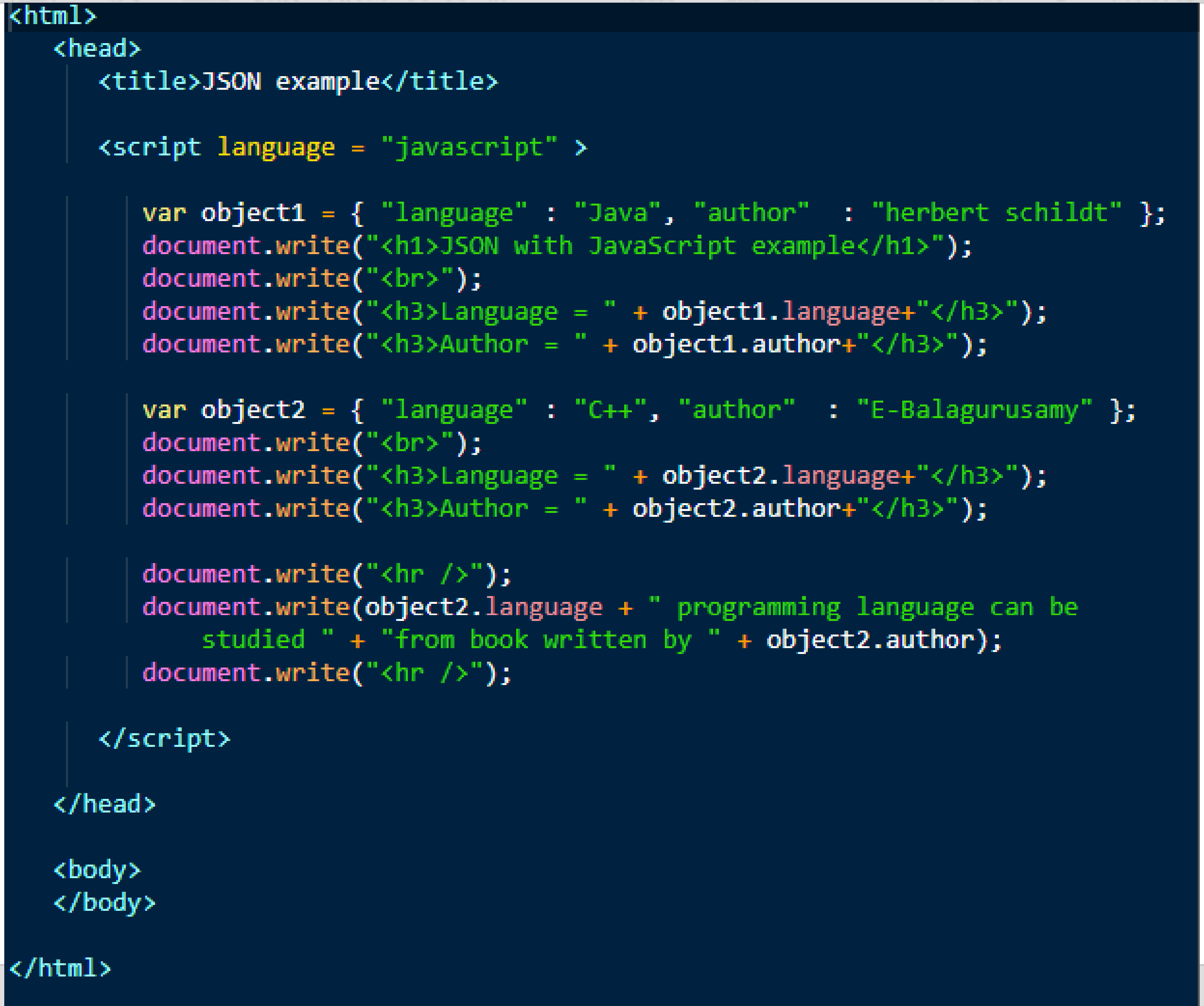
The following example shows how to use JSON to store information related to books based on their topic and edition.



```
{
   "book": [

      {
         "id":"01",
         "language": "Java",
         "edition": "third",
         "author": "Herbert Schildt"
      },

      {
         "id":"07",
         "language": "C++",
         "edition": "second",
         "author": "E.Balagurusamy"
      }
   ]
}
```

Simple Example in JSON

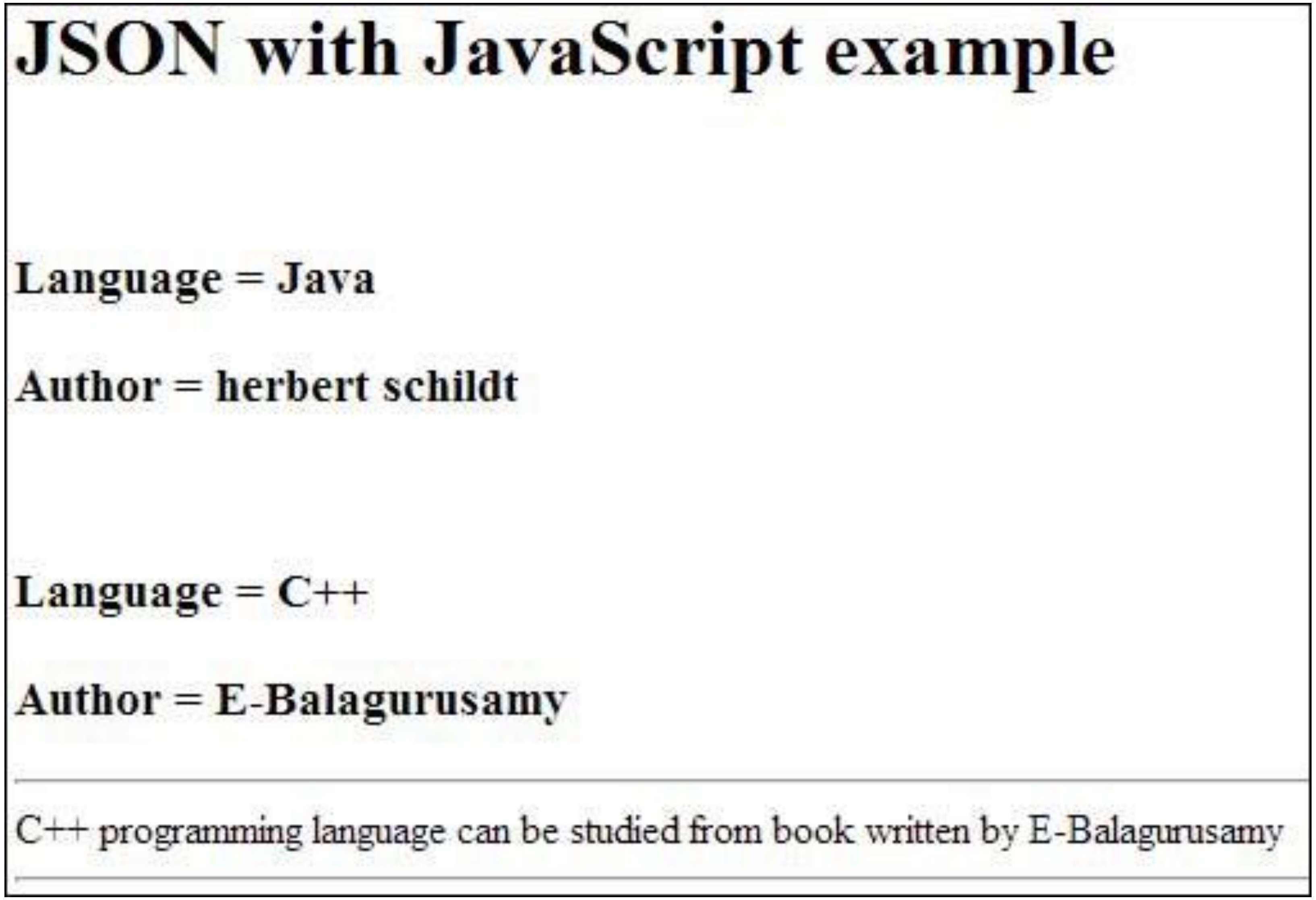
Index.html



```
<html>
    <head>
        <title>JSON example</title>

        <script language = "javascript" >

            var object1 = { "language" : "Java", "author"  : "herbert schildt" };
            document.write("<h1>JSON with JavaScript example</h1>");
            document.write("<br>");
            document.write("<h3>Language = " + object1.language+"</h3>");
            document.write("<h3>Author = " + object1.author+"</h3>");

            var object2 = { "language" : "C++", "author"  : "E-Balagurusamy" };
            document.write("<br>");
            document.write("<h3>Language = " + object2.language+"</h3>");
            document.write("<h3>Author = " + object2.author+"</h3>");

            document.write("<hr />");
            document.write(object2.language + " programming language can be
                studied " + "from book written by " + object2.author);
            document.write("<hr />");

        </script>

    </head>

    <body>
    </body>

</html>
```

Simple Example in JSON

Index.html



JSON with JavaScript example

Language = Java

Author = herbert schildt

Language = C++

Author = E-Balagurusamy

C++ programming language can be studied from book written by E-Balagurusamy

☺ THANK YOU ☺