JavaScript Intro

Mohamed Mukthar Ahmed

JavaScript Intro

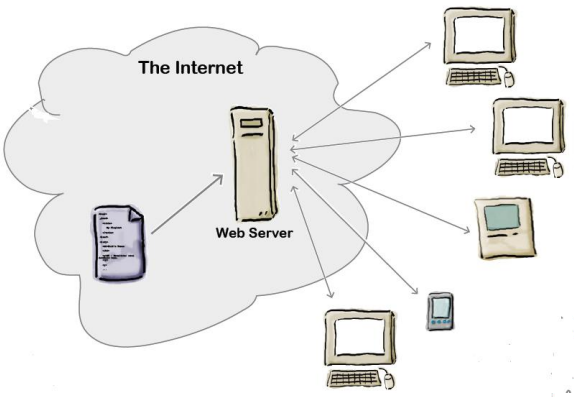
Outline

- JavaScript in Web Development
- Include JavaScript in HTML
- JavaScript Syntax
- Identifiers, Reserved Words, Comments
- Window Object
- Data Types
- Numeric Expressions
- Variables
- parseInt & parseFloat Methods
- Conditional Expression
- The if Statement
- while & do…while Loops
- The for Loop

Mohamed Mukthar Ahmed

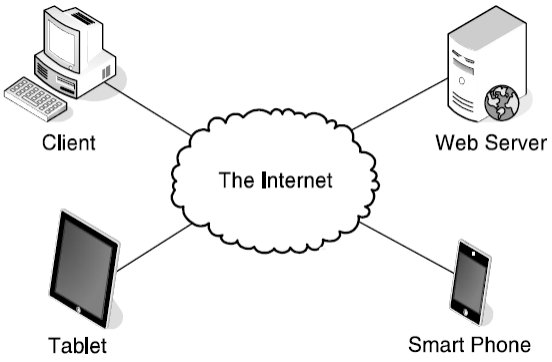
The Web

- The **Web** – Universal form of communication
- To use the Web effectively, we got to know about **HTML**



Web Application Components

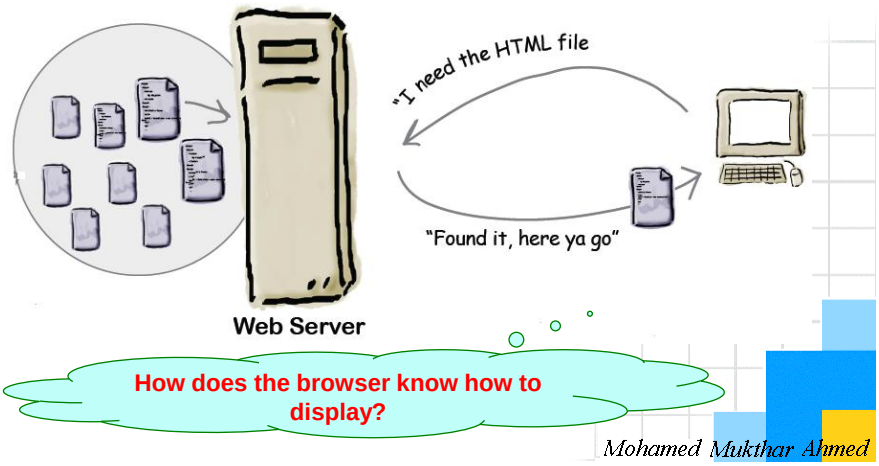
- Components of a Web Application



- A Web Application consists of clients, a web server and a network.

Web Server

- The **Web Server** – accepts **requests** from Web browsers and **responds**

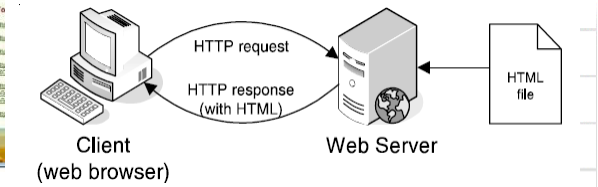


Web Browser

- Web servers store and serve web pages, which are created from HTML and CSS.
- Web Browsers retrieve pages and render their content based on the HTML and CSS

Static Web Pages

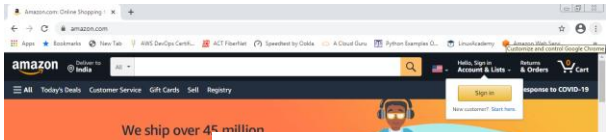
A static web page at http://www.valleytownhall.com/toobin.html

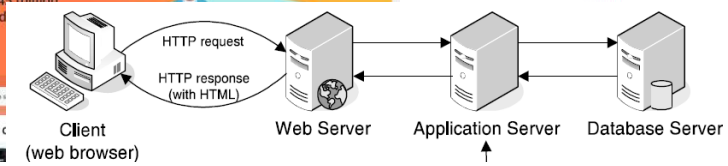


- A *static web page* is an HTML document that's stored on the web server and doesn't change. The filenames for static web pages have .htm or .html extensions.
- When the user requests a static web page, the browser sends an *HTTP request* to the web server that includes the name of the file that's being requested.
- When the web server receives the request, it retrieves the HTML for the web page and sends it back to the browser as part of an *HTTP response*.
- When the browser receives the HTTP response, it *renders* the HTML into a web page that is displayed in the browser.

Mohamed Mukthar Ahmed

Dynamic Web Pages



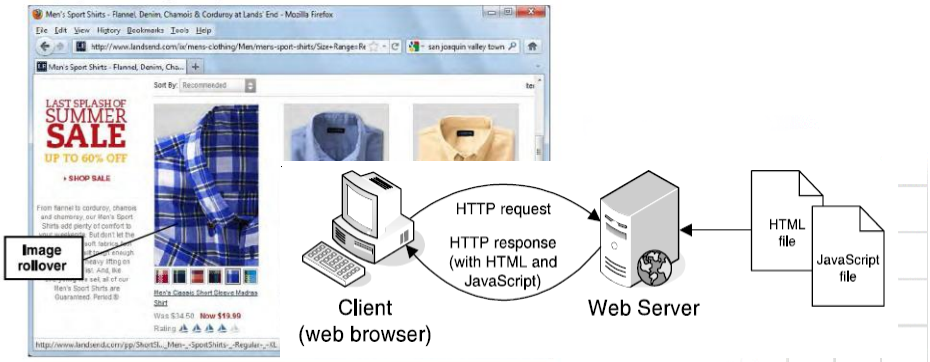


- A *dynamic web page* is a web page that's generated by a server-side program or script.
- When a web server receives a request for a dynamic web page, it looks up the extension of the requested file to find out which *application server* should process the request.
- When the application server receives a request, it runs the specified script. Often, this script uses the data that it gets from the web browser to get the appropriate data from a *database server*. This script can also store the data that it receives in the database.
- When the application server finishes processing the data, it generates the HTML for a web page and returns it to the web server. Then, the web server returns the HTML to the web browser as part of an HTTP response.

Mohamed Mukthar Ahmed

JavaScript in Web Development

A web page with image swaps and rollovers



- *JavaScript* is a *client-side scripting language* that is run by the *JavaScript engine* of a web browser and controls the operation of the browser.
- When the browser requests an HTML page that contains JavaScript or a link to a JavaScript file, both the HTML and the JavaScript are loaded into the browser.
- Because JavaScript runs on the client, not the server, it provides functions that don't require a trip back to the server. This can help an application run more efficiently.

Mohamed Mukthar Ahmed

Include JavaScript in HTML

- A **<script>** element in the head section is commonly used to identify an external JavaScript file
- The **<script>** element in the head section can also contain JavaScript statements. This is referred as Embedded JavaScript
- If we code more than one script element in the head section, the JavaScript is included in the sequence
- When a script element in the head section includes an external JavaScript file, the file runs as if it is coded in the script element

Attribute	Description
`src`	Specifies the location and name of an external JavaScript file.
`type`	With HTML5, this attribute can be omitted. If you code it, use "text/javascript" for JavaScript code.

Mohamed Mukthar Ahmed

Include JavaScript in HTML

A script element in the head section that loads an external JavaScript file

```
<script src="calculate_mpg.js"></script>
```

A script element that embeds JavaScript in the head section

```
<head>
    ...
    <script>
        alert("The Calculate MPG application");
        var miles = prompt("Enter miles driven");
        miles = parseFloat(miles);
        var gallons = prompt("Enter gallons of gas used");
        gallons = parseFloat(gallons);
        var mpg = miles/gallons;
        mpg = parseInt(mpg);
        alert("Miles per gallon = " + mpg);
    </script>
</head>
```


JavaScript in the body of HTML doc

- If the **<script>** element is coded in the body of the HTML doc, it is replaced with the output of JavaScript code.
- The **<noscript>** element can be used to display content when JavaScript is disabled

JavaScript in the body of an HTML document

```
<p>
    <script>
        var today = new Date();
        document.write("Current date: ");
        document.write(today.toDateString());
    </script>
</p>
<p>&copy; 
    <script>
        var today = new Date();
        document.write( today.getFullYear() );
    </script>
, San Joaquin Valley Town Hall
</p>
```

JavaScript in the body of HTML doc

- The **<noscript>** element can be used to display content when JavaScript is disabled

A noscript element in the body of an HTML document

```
<p>&copy; 
    <script>
        var today = new Date();
        document.write( today.getFullYear() );
    </script>
    <noscript>2012</noscript>
, San Joaquin Valley Town Hall
</p>
```

A noscript element at the start of an HTML document

```
<h2><noscript>To get the most from this web site,
    please enable JavaScript.</noscript></h2>
```


JavaScript Syntax

- A JavaScript statement has a syntax that's similar to the syntax of Java
- JavaScript will try to correct what it thinks is a missing semi-colon by adding a semi-colon at the end of a split line. To prevent this follow the guidelines for splitting a statement
- Split a statement after
  - an arithmetic or relation operator
  - an open brace {, bracket [or parenthesis (
  - a closing brace
- Do not split a statement after – an identifier, a value or a return keyword, a closing bracket] or parenthesis)

Identifiers

- Identifiers are the names given to variables, functions, objects, properties and methods
- In Camel Casing, all the words within an identifier except the first word starts with a capital letter
- Norms
  - Only letters, numbers, underscore and dollar sign
  - Can't start with a number

Valid identifiers in JavaScript

```
subtotal        index_1          $
taxRate         calculate_click  $log
```

Camel casing versus underscore notation

```
taxRate          tax_rate
calculateClick   calculate_click
emailAddress     email_address
firstName        first_name
futureValue      future_value
```


Reserved Words

- Words which are part of the JavaScript language

Reserved words in JavaScript

```
abstract    else        instanceof    switch
boolean     enum        int           synchronized
break       export      interface     this
byte        extends     long          throw
case        false       native        throws
catch       final       new           transient
char        finally     null          true
class       float       package       try
const       for         private       typeof
continue    function    protected     var
debugger    goto        public        void
default     if          return        volatile
delete      implements  short         while
do          import      static        with
double      in          super
```

Comments

- Block comments begin with /* and end with */
- Single line comments being with two forward slash //
- Guidelines
  - Use to describe code that is hard to understand
  - Used to disable code that we don't want to test
  - Don't use comment unnecessarily
- Comments are ignored when JavaScript is executed

Object, Methods & Properties

- An **object** has **methods** that perform action (functions) as well as **properties** that represent the data
- When we **call** a method, we may need to pass one or more **parameters** to is by coding them within parentheses
- The **window** object is the ***global object*** for JavaScript.
- JavaScript let's you omit the object name and dot when referring to the window object

The syntax for accessing a property of an object

```
objectName.propertyName
```

A statement that displays the location property of the window object

```
alert(window.location);        // Displays the URL of the current page
```

Window Object

- The common methods & property of window object

Common methods of the window object

Method	Description
`alert(string)`	Displays a dialog box that contains the string that's passed to it by the parameter along with an OK button.
`prompt(string,default)`	Displays a dialog box that contains the string in the first parameter, the default value in the second parameter, an OK button, and a Cancel button. When the user enters a value and clicks OK, that value is returned as a string. Or if the user clicks Cancel, null is returned.
`print()`	Issues a print request to the browser.

One property of the window object

Property	Description
`location`	The URL of the current web page.

Data Types

- When we develop applications we work with data
- JavaScript provides THREE Primitive Data Types
  - Number
  - String
  - Boolean
- Number data type represents an integer or a decimal value
  - If result stored in a number data type is smaller or larger than the data type can store, it will store a value **Infinity**
- String data type represents characters, surrounded in double or single quotes
- Boolean data types represent – **true** or **false**

Data Types

- Example on different data types

Examples of number values

```
15                  // an integer
-21                 // a negative integer
21.5                // a decimal value
-124.82             // a negative decimal value
-3.7e-9             // floating-point notation for -0.0000000037
```

Examples of string values

```
"JavaScript"        // a string with double quotes
'String Data'       // a string with single quotes
""                  // an empty string
```

The two Boolean values

```
true                // equivalent to true, yes, or on
false               // equivalent to false, no, or off
```

Mohamed Mukthar Ahmed

Numeric Expressions

- To code a numeric expression we use arithmetic operators
- The modulus operator returns the remainder
- Arithmetic expression are evaluated based on the order of precedence of the operators
- To override the order of precedence use ( )

Operator	Description	Example	Result
+	Addition	5 + 7	12
-	Subtraction	5 - 12	-7
*	Multiplication	6 * 7	42
/	Division	13 / 4	3.25
%	Modulus	13 % 4	1
++	Increment	counter++	adds 1 to counter
--	Decrement	counter--	subtracts 1 from counter

Mohamed Mukthar Ahmed

Numeric Expressions

- To code a numeric expression we use arithmetic operators
- The modulus operator returns the remainder
- Arithmetic expression are evaluated based on the order of precedence of the operators
- To override the order of precedence use ( )

The order of precedence for arithmetic expressions

Order	Operators	Direction	Description
1	++	Left to right	Increment operator
2	--	Left to right	Decrement operator
3	* / %	Left to right	Multiplication, division, modulus
4	+ -	Left to right	Addition, subtraction

Variables

- A variable stores a value that can change as the program executes
- To declare a variable, code the keyword **var** and a variable name
- To assign a value to a variable use the assignment operator (=)

Variables

How to declare numeric variables without assigning values to them

```
var subtotal;                           // declares one variable
var investment, interestRate, years;    // declares three variables
```

How to declare variables and assign values to them

```
var subtotal = 74.00;                   // subtotal = 74.00
var salesTax = subtotal * .1;           // salesTax = 7.4
```

How to code compound assignment statements

```
var subtotal = 74.95;                   // subtotal = 74.95
subtotal += 20.00;                      // subtotal = 94.95
```

Three ways to increment a variable named counter by 1

```
var counter = 1;                        // counter = 1
counter = counter + 1;                  // counter now = 2
counter += 1;                           // counter now = 3
counter++;                              // counter now = 4
```

A floating-point result that isn't precise

```
var subtotal = 74.95;                   // subtotal = 74.95
var salesTax = subtotal * .1;           // salesTax = 7.495000000000001
```


Variables

The concatenation operator for strings

Operator	Example	Result
+	"Ray " + "Harris"	"Ray Harris"
	"Months: " + 120	"Months: 120"

Escape sequences that can be used in strings

Operator	Description
\n	Starts a new line in a string.
\"	Puts a double quotation mark in a string.
\'	Puts a single quotation mark in a string.

How to declare string variables without assigning values to them

```
var zipCode;                            // declares one variable
var lastName, state, zipCode;           // declares three variables
```

How to declare string variables and assign values to them

```
var firstName = "Ray", lastName = "Harris";   // assigns two string values
var fullName = lastName + ", " + firstName;   // fullName is "Harris, Ray"
```

parseInt & parseFloat Methods

- The window object provides parseInt and parseFloat methods that helps in converting string values to integer or decimal number
- **NaN** (Not a Number) is a value that is returned by parseInt or parseFloat when it cannot convert

Method	Description
`parseInt(string)`	Converts the string that's passed to it to an integer data type and returns that value. If it can't convert the string to an integer, it returns NaN.
`parseFloat(string)`	Converts the string that's passed to it to a decimal data type and returns that value. If it can't convert the string to a decimal value, it returns NaN.

parseInt & parseFloat Methods

```
var entryA = prompt("Enter any value", 12345.6789);
alert(entryA);                              // displays 12345.6789
alert(parseInt(entryA));                    // displays 12345

var entryB = prompt("Enter any value", 12345.6789);
alert(entryB);                              // displays 12345.6789
alert(parseFloat(entryB));                  // displays 12345.6789

var entryC = prompt("Enter any value", "Hello");
alert(entryC);                              // displays Hello
alert(parseInt(entryC));                    // displays NaN
```

Conditional Expressions

- A conditional expression uses **relational operators**
- A compound conditional expression uses **logical operators**
- The **isNaN()** tests whether a string can be converted to a number or not

Operator	Description	Example
==	Equal	`lastName == "Harris"` `testScore == 10`
!=	Not equal	`firstName != "Ray"` `months != 0`
<	Less than	`age < 18`
<=	Less than or equal	`investment <= 0`
>	Greater than	`testScore > 100`
>=	Greater than or equal	`rate / 100 >= 0.1`

Mohamed Mukthar Ahmed

Conditional Expressions

- A compound conditional expression uses logical operators
- The **isNaN()** tests whether a string can be converted to a number or not

The logical operators in order of precedence

Operator	Description	Example
!	NOT	`!isNaN(age)`
&&	AND	`age > 17 && score < 70`
||	OR	`isNaN(rate) || rate < 0`

The syntax of the global isNaN method

```
isNaN(expression)
```

Examples of the isNaN method

```
isNaN("Harris") // Returns true since "Harris" is not a number
isNaN("123.45") // Returns false since "123.45" can be converted
```

Mohamed Mukthar Ahmed

The if Statement

The syntax of the if statement

```
if ( condition-1 ) { statements }
[ else if ( condition-2 ) { statements }
  ...
  else if ( condition-n ) { statements } ]
[ else { statements } ]
```

An if statement with an else clause

```
if ( age >= 18 ) {
    alert ("You may vote.");
} else {
    alert ("You are not old enough to vote.");
}
```

Mohamed Mukthar Ahmed

The if Statement

An if statement with else if and else clauses

```
if ( isNaN(rate) ) {
    alert ("You did not provide a number for the rate.");
} else if ( rate < 0 ) {
    alert ("The rate may not be less than zero.");
} else if ( rate > 12 ) {
    alert ("The rate may not be greater than 12.");
} else {
    alert ("The rate is: " + rate + ".");
}
```

An if statement with a compound conditional expression

```
if ( isNaN(userEntry) || userEntry <= 0 ) {
    alert ("Please enter a valid number greater than zero.");
}
```

Two ways to test whether a Boolean variable is true

```
if ( isValid == true ) { }
if ( isValid ) { }                // same as isValid == true
```

Three ways to test whether a Boolean variable is false

```
if ( isValid == false ) { }
if ( !isValid == true ) { }
if ( !isValid ) { }               // same as !isValid == true
```

Mohamed Mukthar Ahmed

while & do...while Loops

The syntax of a while loop

```
while ( condition ) { statements }
```

A while loop that adds the numbers from 1 through 5

```
var sumOfNumbers = 0;
var numberOfLoops = 5;
var counter = 1;
while (counter <= numberOfLoops) {
    sumOfNumbers += counter;     // adds counter to sumOfNumbers
    counter++;                   // adds 1 to counter
}
alert(sumOfNumbers);             // displays 15
```


while & do...while Loops

The syntax of a do-while loop

```
do { statements } while ( condition );
```

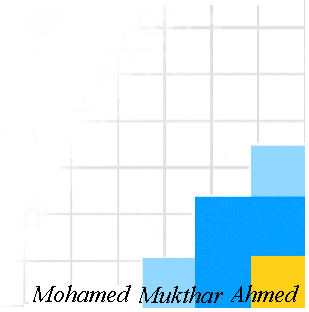
A while loop that adds the numbers from 1 through 5

```
var sumOfNumbers = 0;
var numberOfLoops = 5;
var counter = 1;
do {
    sumOfNumbers += counter;     // adds counter to sumOfNumbers
    counter++;                   // adds 1 to counter
}
while (counter <= numberOfLoops);
alert(sumOfNumbers);             // displays 15
```

Average of a series of numbers

A while loop that finds the average of a series of numbers

```
var total = 0, count = 0, number;
number = parseFloat( prompt("Enter a number:") );
while ( !isNaN(number) ) {
    total += number;
    count++;
    number = parseFloat(
        prompt("Enter another number or click Cancel to stop:") );
}
var average = total / count;
alert("The average is: " + average);
```



The for Loop

The syntax of a for statement

```
for ( counterInitialization; condition; incrementExpression ) {
    statements
}
```

A for loop that adds the numbers from 1 through 5

```
var sumOfNumbers = 0;
var numberOfLoops = 5;
for ( counter=1; counter <= numberOfLoops; counter++ ) {
    sumOfNumbers += counter;      // adds counter to sumOfNumbers
}
alert(sumOfNumbers);              // displays 15
```

Calculate Future Value of Investment

A for loop that calculates the future value of an investment

```
var investment = 10000;
var annualRate = 7.0;
var years = 10;
var futureValue = investment;
for ( i = 1; i <= years; i++ ) {
    futureValue += futureValue * annualRate / 100;
}
alert (futureValue);              // displays 19672
```

Other ways that the future value calculation could be coded

```
futureValue = futureValue + (futureValue * annualRate / 100);
futureValue = futureValue * (1 + (annualRate / 100))
```


Mohamed Mukthar Ahmed