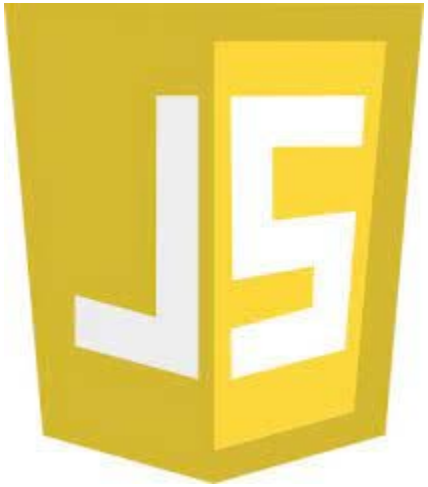
# Introduction to JavaScript

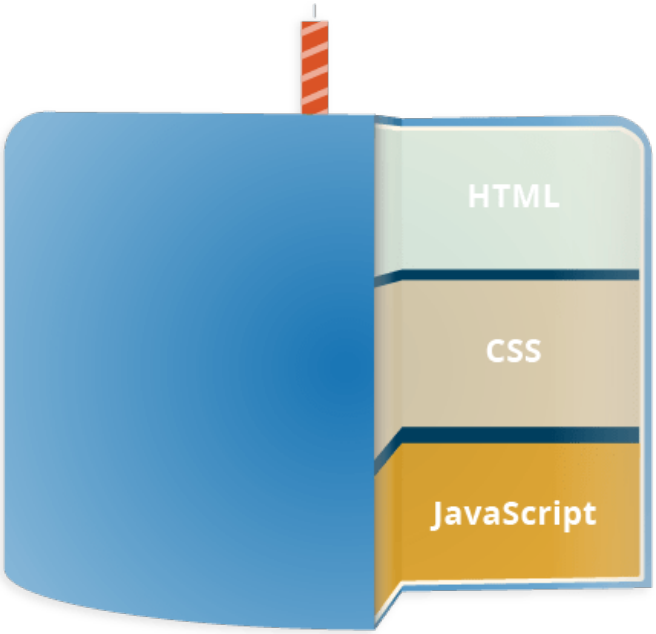


# Agenda

- What is JavaScript
- Features of JavaScript
- How to JavaScript
- JavaScript Programming
- JavaScript Objects
- HTML DOM Objects
- DOM Events
- JSON
- AJAX

# What is JavaScript?

- **JavaScript** is a scripting language widely used for client-side web development



# What can a JavaScript Do?

- Java Script gives HTML designers a programming tool
- Java Script can put dynamic text into an HTML page
- Java Script can read and write HTML elements
- Java Script can react to events
- Java Script can calculate, manipulate and validate data

# JavaScript in Web pages

- JavaScript code can run locally in a user's browser it can respond to user actions quickly, making an application feel more responsive
- A JavaScript engine (also known as JavaScript interpreter or JavaScript implementation) is an interpreter that interprets JavaScript source code and executes the script accordingly

# JavaScript Security

- JavaScript and the DOM provide the potential for malicious authors to deliver scripts to run on a client computer via the web
  - Scripts run in a sandbox in which they can only perform web-related actions, not general-purpose programming tasks like creating files
  - Scripts are constrained by the same origin policy

# Features of JavaScript

- Imperative and structured
  - JavaScript supports all the structured programming syntax in C
    - if - switch statement, for - while loop, etc.
- Dynamic
  - Dynamic typing
    - JavaScript supports various ways to test the type of an object
  - Objects as associative arrays
    - The properties of an object can also be enumerated via a for...in loop
  - Run-time evaluation
    - an eval function can execute statements provided as strings at run-time

# Features of JavaScript

- Functional
  - First-class functions
    - Function have properties and can be passed around and interacted with like any other object
- Inner functions and closures
  - Inner functions are created each time the outer function is invoked, and variables of the outer functions for that invocation continue to exist as long as the inner functions still exist, even after that invocation is finished

# Features of JavaScript

- Prototype-based
  - Prototypes
    - JavaScript uses prototypes instead of classes for defining object properties, including methods, and inheritance.
  - Functions as object constructors
    - Functions double as object constructors along with their typical role
    - Prefixing a function call with *new* creates a new object and calls that function with its local *this* keyword bound to that object for that invocation
  - Functions as methods
    - A function can be called as a method

# Features of JavaScript

- Miscellaneous
  - Run-time environment
    - JavaScript typically relies on a run-time environment (in a web browser) to provide objects and methods by which scripts can interact with "the outside world"
  - Variadic functions
    - An indefinite number of parameters can be passed to a function. The function can both access them through formal parameters and the local *arguments* object
  - Array and object literals
    - Arrays and objects can each be created with a succinct shortcut syntax, literals form the basis of the JSON data format
  - Regular expressions
    - JavaScript also supports regular expressions in a manner similar to Perl, which provide a concise and powerful syntax for text manipulation that is more sophisticated than the built-in string functions

# How To JavaScript

- Getting Started
  - To embed JavaScript code in an HTML document, it must be preceded with

```
<script>
…
</script>
```

1

```
<script type="text/javascript">
//<![CDATA[
…
//]]>
</script>
```

2

```
<script language="JavaScript" type="text/javascript">
<!–
…
// -->
</script>
```

3

```
<script src="../path-to/source.js"></script>
```

4

# How To JavaScript

- Hello World

```
<HTML>
<HEAD>
<TITLE>Hello World</TITLE>
<script>
document.write("Hello World");
</script>
</HEAD>
<BODY>
</BODY>
</HTML>
```

# JavaScript Programming

- White Space
  - space, tab , newline
- Comment
  - // comment on one line
  - /* comment one or more lines */
- Separators
  - () arguments, parameters
  - {} array assignment, block
  - [] array definition & index
  - ; terminate statement
  - , declaration, for-loop definition

# Reserve Word

| | | | |
|---|---|---|---|
| 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 | |

# Variables

- Variables must conform
  - No conflict with reserve words
  - Case sensitive
  - They must start with a letter or underscore ("_")
  - Subsequent characters can also be digits (0-9) or letters (A-Z and/or a-z).

```
var myVar;
var MyVar = “String”;
var _myVar = true;
var myVar1 = null;
var x = 1, y = ‘Hello’, z = false;
```

# Data Type Conversion

- Dynamic Typing
  - JavaScript is a loosely typed language

```
var x = 10;
…
x = “Hello World”;
```

  - The + sign tells JavaScript to concatenate or stick together

```
var x = "The answer is " + 42;
var y = 42 + " is the answer";
var msg = "Hello" + "World";
```

# Literals

- Integer
  - decimal (base 10) ex. 12;
  - octal (base 8) ex. 012;
  - hexadecimal (base 16) ex. 0x12;
- Floating-point
  - standard ex. 0.123
  - science ex. 0.123E-1
- Boolean
  - true , false

# Literals

- String
  - A string literal is zero or more characters enclosed in double (") or single (') quotation marks
    - "blah"
    - 'blah'
    - "1234"
    - "one line \n another line"

# Literals

- Escape Characters

| Character | Meaning |
|---|---|
| \b | backspace |
| \f | form feed |
| \n | new line |
| \r | carriage return |
| \t | tab |
| \\ | backslash character |

```
var quote = "I read \"Ajax in Practice\" by Dave Crane"
var home = "c:\\temp"
```

# Operator

- Arithmetic Operator

  + , - , * , / , % , ++ , --

- Assignment Operator

| Shorthand operator | Meaning |
|---|---|
| x = y | assign |
| x += y | x = x + y |
| x -= y | x = x - y |
| x *= y | x = x * y |
| x /= y | x = x / y |
| x%=y | x = x%y |

# Operator

- Comparison Operator

| Operator | Description | Example |
|---|---|---|
| Equal (==) | Evaluates to true if the operands are equal. | x == y evaluates to true if x equals y. |
| Not equal (!=) | Evaluates to true if the operands are not equal. | x != y evaluates to true if x is not equal to y. |
| Greater than (>) | Evaluates to true if left operand is greater than right operand. | x > y evaluates to true if x is greater than y. |
| Greater than or equal (>=) | Evaluates to true if left operand is greater than or equal to right operand. | x >= y evaluates to true if x is greater than or equal to y. |
| Less than (<) | Evaluates to true if left operand is less than right operand. | x < y evaluates to true if x is less than y. |
| Less than or equal (<=) | Evaluates to true if left operand is less than or equal to right operand. | x <= y evaluates to true if x is less than or equal to y. |

# Operator

- Logical Operator

| Operator | Usage | Description |
|---|---|---|
| and (&&) | expr1 && expr2 | True if both logical expressions expr1 and expr2 are true. False otherwise. |
| or (\|\|) | expr1 \|\| expr2 | True if either logical expression expr1 or expr2 is true. False if both expr1 and expr2 are false. |
| not (!) | !expr | False if expr is true; true if expr is false. |

# Operator

- Ternary Operator

```
Expression ? X : Y
```

```
a = (1>0) ? true : false;

b = (x % 2 == 0) ? “even” : “odd”;
```

# Operator

- Precedence of Operator

| Priority | Operator |
|---|---|
| 1 | () |
| 2 | / |
| 3 | * |
| 4 | % |
| 5 | +/- |
| 6 | && |
| 7 | \|\| |
| 8 | +=, -=, /=, %= |

```
a = 2 * 100 / 10; => 20
//2 * (100 / 10)

b = 100 / 10 % 2; => 0
//(100/10) % 2

c = 100/10 + 30; => 40
//(100/10) + 30

d = 2*100/10+30; => 50
//2*(100/10))+30
```

# Control Statement

- if/else statement

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

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

# Control Statement

- Dangling Else
  - if (condition1) if (condition2) statement1;
    else statement2;

1.

```
if (condition1)
    if (condition2)
        statement1
else
    statement2
```

2.

```
if (condition1)
    if (condition2)
        statement1
    else
        statement2
```

# Control Statement

- switch statement

```
switch (expression) {
        case value1 :
                statements1
                break;
        case value2 :
                statements2;
                break;
        …
        default :
                default-statement;
                break;
}
```

# Loops

- for loop

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

- for …in loop

```
for (variable in object) {
    statements
}
```

# Loops

- while loop

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

- do … while loop

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

# Function

- A function is a JavaScript procedure
- A function definition has these basic parts
  - The function keyword
  - A function name
  - A comma-separated list of arguments to the function in parentheses
  - The statements in the function in curly braces: { }

# Function

- Defining Function

```
function functionname(var1,var2,...,var n){
        statements;
        result = statements;
        return result;
}
```

1

```
var functionname = function(var1,var2,...,var n){
        statements
}
```

2

```
function helloWorld(){
        document.write('Hello World');
}
```

```
function prod(a,b) {
        x=a*b;
        return x;
}
```

```
var hello = function(msg) {
        document.write("Hello "+msg);
}
```

# Function

- Calling Function

```
<button onclick="helloWorld()">Hello</button>
```

```
function anotherHello() {
        helloWorld();
        hello("World");
        product = prod(4,5);
}
```

```
function multiply() {
        document.write("result="+prod(5,10));
}
```

# Function

- Functions as Methods

```
function inc(x) {
        return ++x;
}
var obj = {};
obj.sqrt = function(x) { return x*x; }
obj.increment = inc;

document.write(obj.sqrt(10));
document.write(obj.increment(9));
```

# Function

- Variadic Function

```
<html>
<head>
<title>Arguments</title>
<script language="JavaScript">
function play() {
        for(var i=0;i<arguments.length;i++) {
                alert(play.arguments[i]);
        }
}
</script>
</head>
<body>
<button onclick="play(1,2,3)">Play 1</button>
<button onclick="play('hello','world')">Play 2</button>
</body>
</html>
```

# Exception Handling

- Catching Errors
  - By using the **try**...**catch** statement

```
<html>
<head>
<title>Try Catch</title>
<script language="JavaScript">
function test() {
        try {
                addAlert("Welcome guest!");
        } catch(ex) {
                txt="There was an error on this page.\n\n";
                txt+="Error description: " + ex.description + "\n\n";
                txt+="Click OK to continue.\n\n";
                alert(txt);
        }
}
</script>
</head>
<body>
<button onclick="test()">Test</button>
</body>
</html>
```

# Exception Handling

- Catching Errors
  - By using the **onerror** event

```
<html>
<head>
<title>On Error</title>
<script type="text/javascript">
onerror=handleError;
function handleError(msg,url,l) {
                var txt="There was an error on this page.\n\n";
                txt+="Error: " + msg + "\n";
                txt+="URL: " + url + "\n";
                txt+="Line: " + l + "\n\n";
                txt+="Click OK to continue.\n\n";
                alert(txt);
                return true;
}
function test() { addAlert("Welcome guest!"); }
</script>
</head>
<body>
<button onclick="test()">Test</button>
</body>
</html>
```

# Object

- Creating Object

```
personObj = new Object();
personObj.firstname="Tassun";
personObj.lastname="Oros";
personObj.address="Pakkred";
```

- Object as associative array

```
personObj = new Object();
personObj["firstname"]="Tassun";
personObj["lastname"]="Oros";
personObj["address"]="Pakkred";
```

# Object

- Creating Object

```
function display(person){
        for(var p in person){
                document.write(p+"="+person[p]+"<br>");
        }
}
function test(){
        {
                personObj = new Object();
                personObj.firstname="Tassun";
                personObj.lastname="Oros";
                personObj.address="Pakkred";
                display(personObj);
        }
        {
                personObj = new Object();
                personObj["firstname"]="Tassun";
                personObj["lastname"]="Oros";
                personObj["address"]="Pakkred";
                display(personObj);
        }
}
```

# Object

- Creating Object
  - Create a template of an object
    - Function as an object constructor

```
function Person(firstname,lastname,address) {
        this.firstname=firstname;
        this.lastname=lastname;
        this.address=address;
}
var tso = new Person("Tassun","Oros","Pakkred");
```

# Closure

- Inner Function

```
personObj = new Object();
personObj.firstname="Tassun";
personObj.lastname="Oros";
personObj.address="Pakkred";

personObj.setup = function() {
        var msg = "Hello ";
        this.sayHi = function() {
                alert(msg+" : "+this.firstname);
        }
}

personObj.setup();
personObj.sayHi();
```

# Prototype

- Wrap up an object function

```
function Person(firstname,lastname,address){
        this.firstname=firstname;
        this.lastname=lastname;
        this.address=address;
        this.getFullName = function(){
                return this.firstname+" "+this.lastname;
        }
}
```

```
function Person(firstname,lastname,address){
        this.firstname=firstname;
        this.lastname=lastname;
        this.address=address;
}
Person.prototype.getFullName = function(){
        return this.firstname+" "+this.lastname;
}
```

```
Person.prototype.HELLO_MESSAGE = "Hello";
Person.prototype.sayHi = function(){
        alert(this.HELLO_MESSAGE+" : "+this.firstname);
}
```

# Array

- Array Declaration

```
<html>
<head>
<title>Array</title>
<script language="JavaScript">
var ary1 = new Array();
ary1[0] = 'A';
ary1[2] = 10;
var ary2 = ['1','2','3'];
var ary3 = new Array(5);
var ary4 = new Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat");
function test() {
		for(var i=0;i<ary1.length;i++) {
			alert(ary1[i]);
		}
		for(var x in ary4) {
			alert(ary4[x]);
		}
}
</script>
</head>
<body>
<button onclick="test()">Test</button>
</body>
</html>
```

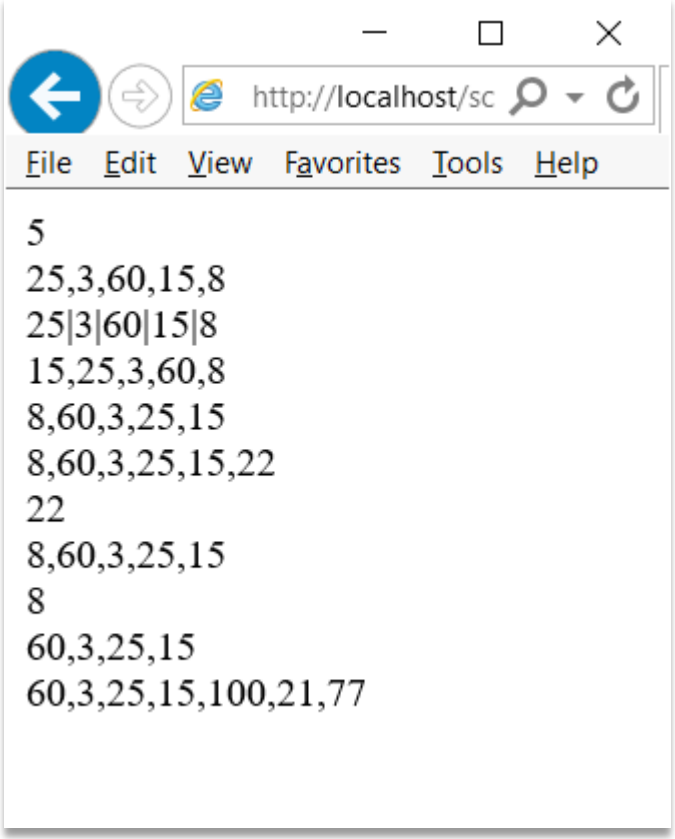
# Array

- Array Object Methods

| Method | Description |
|---|---|
| concat() | Joins two or more arrays and returns the result |
| join() | Puts all the elements of an array into a string. The elements are separated by a specified delimiter |
| pop() | Removes and returns the last element of an array |
| push() | Adds one or more elements to the end of an array and returns the new length |
| reverse() | Reverses the order of the elements in an array |
| shift() | Removes and returns the first element of an array |
| slice() | Returns selected elements from an existing array |
| sort() | Sorts the elements of an array |
| splice() | Removes and adds new elements to an array |
| toSource() | Represents the source code of an object |
| toString() | Converts an array to a string and returns the result |
| unshift() | Adds one or more elements to the beginning of an array and returns the new length |
| valueOf() | Returns the primitive value of an Array object |

# Array

- Example




```
<html>
<head>
<title>Array</title>
<script language="JavaScript">
var ary1 = [25,3,60,15, 8];
var ary2 = [100,21,77];
function test() {
        document.write(ary1.length);
        document.write("<br>");
        document.write(ary1.toString());
        document.write("<br>");
        document.write(ary1.join("|"));
        document.write("<br>");
        document.write(ary1.sort());
        document.write("<br>");
        document.write(ary1.reverse());
        document.write("<br>");
        ary1.push(22);
        document.write(ary1);
        var a = ary1.pop();
        document.write("<br>"+a+"<br>");
        document.write(ary1);
        var b = ary1.shift();
        document.write("<br>"+b+"<br>");
        document.write(ary1);
        document.write("<br>");
        document.write(ary1.concat(ary2));
}
</script>
</head>
<body>
<button onclick="test()">Test</button>
</body>
</html>
```

# JavaScript Function

- Top-level Functions

| Function | Description |
|---|---|
| escape() | Encodes a string |
| eval() | Evaluates a string and executes it as if it was script code |
| isFinite() | Checks if a value is a finite number |
| isNaN() | Checks if a value is not a number |
| Number() | Converts an object's value to a number |
| parseFloat() | Parses a string and returns a floating point number |
| parseInt() | Parses a string and returns an integer |
| String() | Converts an object's value to a string |
| unescape() | Decodes a string encoded by escape() |

# JavaScript Function

- eval Function

```
<html>
<head>
<title>eval</title>
<script language="JavaScript">
function hello(msg) { logger.innerHTML = logger.innerHTML+'<br>Hello : '+msg; }
function evalClick() {
                    logger.innerHTML = "Value is : "+eval(myval.value)*10;
}
function helloClick() {
                    var fn = eval("hello");
                    fn(myval.value);
}
function greetClick() {
                    greet(myval.value);
}
eval("function greet(msg) { logger.innerHTML = logger.innerHTML+'<br>Greet : '+msg; }");
</script>
</head>
<body>
<input type="text" id="myval"></input>
<button onclick="evalClick()">Eval</button>
<button onclick="helloClick()">Hello</button>
<button onclick="greetClick()">Greet</button>
<div id="logger"></div>
</body>
</html>
```

# JavaScript Objects

- String Object

| Method | Description |
|---|---|
| charAt() | Returns the character at a specified position |
| charCodeAt() | Returns the Unicode of the character at a specified position |
| concat() | Joins two or more strings |
| fromCharCode() | Takes the specified Unicode values and returns a string |
| indexOf() | Returns the position of the first occurrence of a specified string value in a string |
| lastIndexOf() | Returns the position of the last occurrence of a specified string value, searching backwards from the specified position in a string |
| match() | Searches for a specified value in a string |
| replace() | Replaces some characters with some other characters in a string |
| search() | Searches a string for a specified value |
| slice() | Extracts a part of a string and returns the extracted part in a new string |
| split() | Splits a string into an array of strings |
| substr() | Extracts a specified number of characters in a string, from a start index |
| substring() | Extracts the characters in a string between two specified indices |
| toLowerCase() | Displays a string in lowercase letters |
| toUpperCase() | Displays a string in uppercase letters |
| valueOf() | Returns the primitive value of a String object |

# JavaScript Objects

- Date Object

| Method | Description |
|---|---|
| Date() | Returns today's date and time |
| getDate() | Returns the day of the month from a Date object (from 1-31) |
| getDay() | Returns the day of the week from a Date object (from 0-6) |
| getFullYear() | Returns the year, as a four-digit number, from a Date object |
| getHours() | Returns the hour of a Date object (from 0-23) |
| getMilliseconds() | Returns the milliseconds of a Date object (from 0-999) |
| getMinutes() | Returns the minutes of a Date object (from 0-59) |
| getMonth() | Returns the month from a Date object (from 0-11) |
| getSeconds() | Returns the seconds of a Date object (from 0-59) |
| getTime() | Returns the number of milliseconds since midnight Jan 1, 1970 |
| getTimezoneOffset() | Returns the difference in minutes between local time and Greenwich Mean Time (GMT) |
| getUTCDate() | Returns the day of the month from a Date object according to universal time (from 1-31) |
| getUTCDay() | Returns the day of the week from a Date object according to universal time (from 0-6) |
| getUTCMonth() | Returns the month from a Date object according to universal time (from 0-11) |
| getUTCFullYear() | Returns the four-digit year from a Date object according to universal time |
| getUTCHours() | Returns the hour of a Date object according to universal time (from 0-23) |
| getUTCMinutes() | Returns the minutes of a Date object according to universal time (from 0-59) |

# JavaScript Objects

- Date Object

| Method | Description |
|---|---|
| getUTCSeconds() | Returns the seconds of a Date object according to universal time (from 0-59) |
| getUTCMilliseconds() | Returns the milliseconds of a Date object according to universal time (from 0-999) |
| getYear() | Returns the year, as a two-digit or a three/four-digit number, depending on the browser. Use getFullYear() instead !! |
| parse() | Takes a date string and returns the number of milliseconds since midnight of January 1, 1970 |
| setDate() | Sets the day of the month in a Date object (from 1-31) |
| setFullYear() | Sets the year in a Date object (four digits) |
| setHours() | Sets the hour in a Date object (from 0-23) |
| setMilliseconds() | Sets the milliseconds in a Date object (from 0-999) |
| setMinutes() | Set the minutes in a Date object (from 0-59) |
| setMonth() | Sets the month in a Date object (from 0-11) |
| setSeconds() | Sets the seconds in a Date object (from 0-59) |
| setTime() | Calculates a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970 |
| setUTCDate() | Sets the day of the month in a Date object according to universal time (from 1-31) |
| setUTCMonth() | Sets the month in a Date object according to universal time (from 0-11) |
| setUTCFullYear() | Sets the year in a Date object according to universal time (four digits) |

# JavaScript Objects

- Date Object

| Method | Description |
|---|---|
| setUTCHours() | Sets the hour in a Date object according to universal time (from 0-23) |
| setUTCMinutes() | Set the minutes in a Date object according to universal time (from 0-59) |
| setUTCSeconds() | Set the seconds in a Date object according to universal time (from 0-59) |
| setUTCMilliseconds() | Sets the milliseconds in a Date object according to universal time (from 0-999) |
| setYear() | Sets the year in the Date object (two or four digits). Use setFullYear() instead !! |
| toDateString() | Returns the date portion of a Date object in readable form |
| toGMTString() | Converts a Date object, according to Greenwich time, to a string. Use toUTCString() instead !! |
| toLocaleDateString() | Converts a Date object, according to local time, to a string and returns the date portion |
| toLocaleTimeString() | Converts a Date object, according to local time, to a string and returns the time portion |
| toLocaleString() | Converts a Date object, according to local time, to a string |
| toSource() | Represents the source code of an object |
| toString() | Converts a Date object to a string |
| toTimeString() | Returns the time portion of a Date object in readable form |
| toUTCString() | Converts a Date object, according to universal time, to a string |
| UTC() | Takes a date and returns the number of milliseconds since midnight of January 1, 1970 according to universal time |
| valueOf() | Returns the primitive value of a Date object |

# JavaScript Objects

- Math Object

| Property | Description |
|---|---|
| E | Returns Euler's constant (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. 3.14159) |
| SQRT1_2 | Returns the square root of 1/2 (approx. 0.707) |
| SQRT2 | Returns the square root of 2 (approx. 1.414) |

# JavaScript Objects

- Math Object

| Method | Description |
|---|---|
| abs(x) | Returns the absolute value of a number |
| acos(x) | Returns the arccosine of a number |
| asin(x) | Returns the arcsine of a number |
| atan(x) | Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians |
| atan2(y,x) | Returns the angle theta of an (x,y) point as a numeric value between -PI and PI radians |
| ceil(x) | Returns the value of a number rounded upwards to the nearest integer |
| cos(x) | Returns the cosine of a number |
| exp(x) | Returns the value of $E^x$ |
| floor(x) | Returns the value of a number rounded downwards to the nearest integer |
| log(x) | Returns the natural logarithm (base E) of a number |
| max(x,y) | Returns the number with the highest value of x and y |
| min(x,y) | Returns the number with the lowest value of x and y |
| pow(x,y) | Returns the value of x to the power of y |
| random() | Returns a random number between 0 and 1 |
| round(x) | Rounds a number to the nearest integer |
| sin(x) | Returns the sine of a number |
| sqrt(x) | Returns the square root of a number |
| tan(x) | Returns the tangent of an angle |

# JavaScript Objects

- RegExp Object
  - The regular expression object describes a pattern of characters

```
var txt=new RegExp(pattern,attributes);
var txt=/pattern/attributes;
```

    - pattern specifies the pattern of the regular expression
    - attributes specifies global ("g"), case-insensitive ("i"), and multiline matches ("m")

# JavaScript Objects

- RegExp Modifiers - Position Matching

| Modifier | Description |
|---|---|
| ^ | Get a match at the beginning of a string |
| $ | Get a match at the end of a string |
| \b | Word boundary. Get a match at the beginning or end of a word in the string |
| \B | Non-word boundary. Get a match when it is not at the beginning or end of a word in the string |
| ?= | A positive look ahead. Get a match if a string is followed by a specific string |
| ?! | A negative look ahead. Get a match if a string is not followed by a specific string |

# JavaScript Objects

- RegExp Modifiers - Character Classes

| Modifier | Description |
|---|---|
| [xyz] | Find any character in the specified character set |
| [^xyz] | Find any character not in the specified character set |
| . (dot) | Find any character except newline or line terminator |
| \w | Find any alphanumeric character including the underscore |
| \W | Find any non-word character |
| \d | Find any single digit |
| \D | Find any non-digit |
| \s | Find any single space character |
| \S | Find any single non-space character |

# JavaScript Objects

- RegExp Modifiers - Repetition

| Modifier | Description |
|---|---|
| {x} | Finds the exact (x) number of the regular expression grouped together |
| {x,} | Finds the exact (x) or more number of the regular expression grouped together |
| {x,y} | Finds between x and y number of the regular expression grouped together |
| ? | Finds zero or one occurrence of the regular expression |
| * | Finds zero or more occurrences of the regular expression |
| + | Finds one or more occurrences of the regular expression |

# JavaScript Objects

- Example – format date

```
function formatDateTime(time,includeDate){
        if(!time) return "";
        var result = "";
        var now;
        if(time instanceof Date) now = time;
        else now = new Date(eval(time));
        if(includeDate){
                var dd = now.getDate();
                var mm = now.getMonth()+1;
                var yy = now.getFullYear();
                result += ((dd < 10) ? "0" : "") + dd;
                result += ((mm < 10) ? "/0" : "/") + mm;
                result += "/"+yy;
                result += " ";
        }
        var hh = now.getHours();
        var mm = now.getMinutes();
        var ss = now.getSeconds();
        result += ((hh < 10) ? "0":"") + hh;
        result += ((mm < 10) ? ":0" : ":") + mm;
        result += ((ss < 10) ? ":0" : ":") + ss;
        return result;
}
```

# JavaScript Objects

- Example – check date

```
function isDate(mydate){
                    if (mydate=="" || mydate == null) return true;
                    if ( checkDate(mydate) ) return true;
                    return false;
}
function checkDate(mydate) {
                    var delimiter = "/";
                    var dmy = /\d{1,2}\/\d{1,2}\/\d{2,4}/ ;
                    if ( !mydate.match(dmy) ) { alert("Invalid date's format"); return false;}
                    var intDay = new Date();
                    var ary_date = mydate.split(delimiter);
                    // ------ check year
                    var y=ary_date[2];
                    if ( y.length==2 ) {
                                        if (y>=80) { y = intDay.getYear()-100-(intDay.getYear()%100)+eval(y) ;  }
                                        else { y = intDay.getYear()-(intDay.getYear()%100)+ eval(y) ;} }
                                        else if ( y.length==3 ) { alert("Invalid year."); return false; }
                                        intDay.setYear(y);

                    // ------ check month
                    var m = eval(ary_date[1]) ;
                    if ( (m>0) && (m<=12) ) { intDay.setMonth(m);   intDay.setDate(0); }
                    else {              alert("Invalid month");  return false ;}
                    // ------ check day
                    var d = eval(ary_date[0]);
                    if ( (d>0) && (d<=intDay.getDate()) ) { return true; }
                    else { alert("Invalid date"); return false;  }
}
```

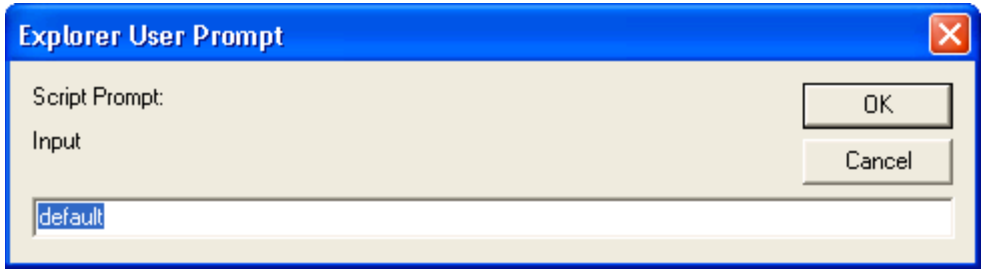
# User Interaction

- Alert dialog box
  - alert(“Hello World”);
- Confirm dialog box
  - var reply = confirm(“Close?”);
- Prompt dialog box
  - var result = prompt(“Input”,”default”);
- Status bar
  - window.status = “Hello World”;

# User Interaction

- IE






- FireFox



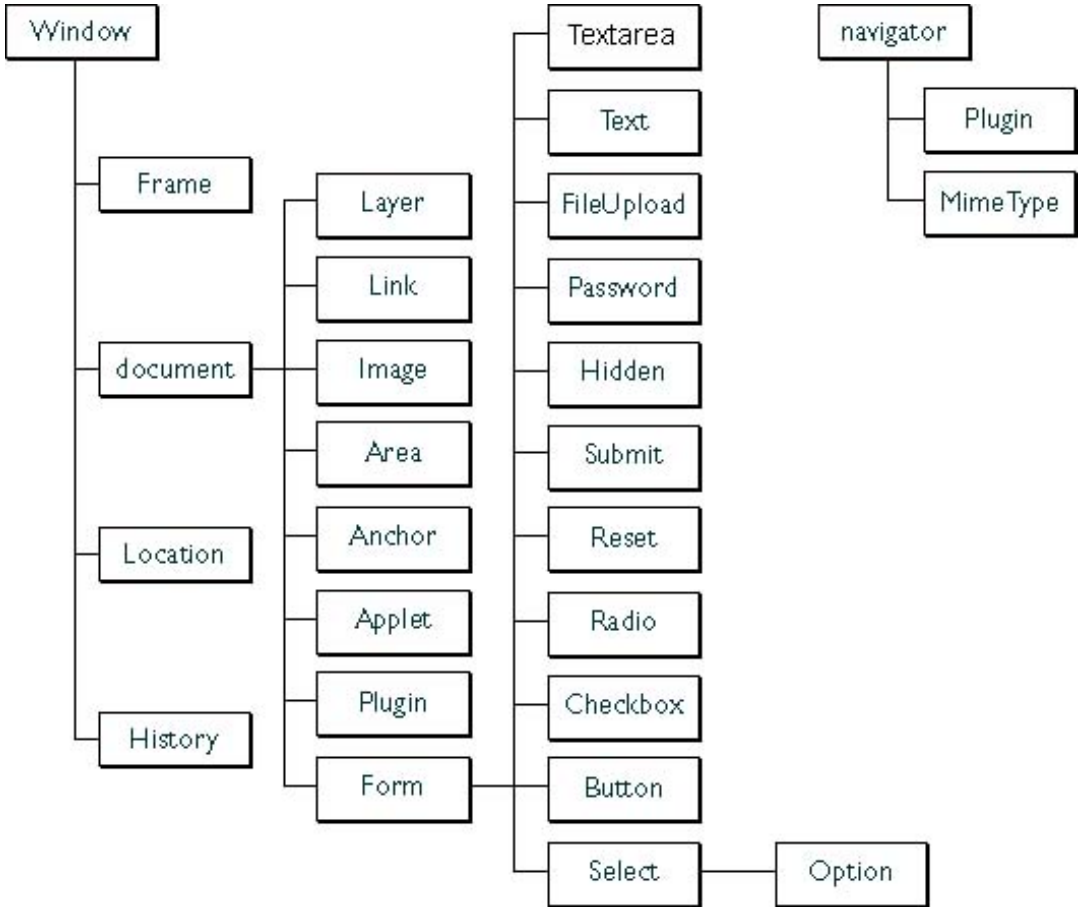
# HTML DOM

- More JavaScript Objects

| Object | Description |
|---|---|
| Window | The top level object in the JavaScript hierarchy. The Window object represents a browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag |
| Navigator | Contains information about the client's browser |
| Screen | Contains information about the client's display screen |
| History | Contains the visited URLs in the browser window |
| Location | Contains information about the current URL |

# DOM Tree

- DOM (Document Object Model)



# Window Object

- Properties

| Property | Description |
|---|---|
| closed | Returns whether or not a window has been closed |
| defaultStatus | Sets or returns the default text in the statusbar of the window |
| document | See Document object |
| history | See History object |
| length | Sets or returns the number of frames in the window |
| location | See Location object |
| name | Sets or returns the name of the window |
| opener | Returns a reference to the window that created the window |
| outerHeight | Sets or returns the outer height of a window |
| outerWidth | Sets or returns the outer width of a window |
| pageXOffset | Sets or returns the X position of the current page in relation to the upper left corner of a window's display area |
| pageYOffset | Sets or returns the Y position of the current page in relation to the upper left corner of a window's display area |
| parent | Returns the parent window |

# Window Object

- Properties

| Property | Description |
|---|---|
| personalbar | Sets whether or not the browser's personal bar (or directories bar) should be visible |
| scrollbars | Sets whether or not the scrollbars should be visible |
| self | Returns a reference to the current window |
| status | Sets the text in the statusbar of a window |
| statusbar | Sets whether or not the browser's statusbar should be visible |
| toolbar | Sets whether or not the browser's tool bar is visible or not (can only be set before the window is opened and you must have UniversalBrowserWrite privilege) |
| top | Returns the topmost ancestor window |

# Window Object

- Methods

| Method | Description |
|---|---|
| alert() | Displays an alert box with a message and an OK button |
| blur() | Removes focus from the current window |
| clearInterval() | Cancels a timeout set with setInterval() |
| clearTimeout() | Cancels a timeout set with setTimeout() |
| close() | Closes the current window |
| confirm() | Displays a dialog box with a message and an OK and a Cancel button |
| createPopup() | Creates a pop-up window |
| focus() | Sets focus to the current window |
| moveBy() | Moves a window relative to its current position |
| moveTo() | Moves a window to the specified position |
| open() | Opens a new browser window |
| print() | Prints the contents of the current window |
| prompt() | Displays a dialog box that prompts the user for input |
| resizeBy() | Resizes a window by the specified pixels |

# Window Object

- Methods

| Method | Description |
|---|---|
| resizeTo() | Resizes a window to the specified width and height |
| scrollBy() | Scrolls the content by the specified number of pixels |
| scrollTo() | Scrolls the content to the specified coordinates |
| setInterval() | Evaluates an expression at specified intervals |
| setTimeout() | Evaluates an expression after a specified number of milliseconds |

# Window Object

- Open New Window
  - window.open(URL,name,specs,replace)

| Parameter | Description |
|---|---|
| URL | Optional. Specifies the URL of the page to open. If no URL is specified, a new window with about:blank is opened |
| name | Optional. Specifies the target attribute or the name of the window. The following values are supported:<br>•_blank - URL is loaded into a new window. This is default<br>•_parent - URL is loaded into the parent frame<br>•_self - URL replaces the current page<br>•_top - URL replaces any framesets that may be loaded<br>•*name* - The name of the window |
| specs | Optional. A comma-separated list of items. The following values are supported: |
| replace | Optional.Specifies whether the URL creates a new entry or replaces the current entry in the history list. The following values are supported:<br>•true - URL replaces the current document in the history list<br>•false - URL creates a new entry in the history list |

# Window Object

- Open New Window
  - Specs or Options

| | |
|---|---|
| channelmode=yes\|no\|1\|0 | Whether or not to display the window in theater mode. Default is no |
| directories=yes\|no\|1\|0 | Whether or not to add directory buttons. Default is yes |
| fullscreen=yes\|no\|1\|0 | Whether or not to display the browser in full-screen mode. Default is no. A window in full-screen mode must also be in theater mode |
| height=pixels | The height of the window. Min. value is 100 |
| left=pixels | The left position of the window |
| location=yes\|no\|1\|0 | Whether or not to display the address field. Default is yes |
| menubar=yes\|no\|1\|0 | Whether or not to display the menu bar. Default is yes |
| resizable=yes\|no\|1\|0 | Whether or not the window is resizable. Default is yes |
| scrollbars=yes\|no\|1\|0 | Whether or not to display scroll bars. Default is yes |
| status=yes\|no\|1\|0 | Whether or not to add a status bar. Default is yes |
| titlebar=yes\|no\|1\|0 | Whether or not to display the title bar. Ignored unless the calling application is an HTML Application or a trusted dialog box. Default is yes |
| toolbar=yes\|no\|1\|0 | Whether or not to display the browser toolbar. Default is yes |
| top=pixels | The top position of the window |
| width=pixels | The width of the window. Min. value is 100 |

# Navigator Object

- Properties

| Property | Description |
|---|---|
| appCodeName | Returns the code name of the browser |
| appMinorVersion | Returns the minor version of the browser |
| appName | Returns the name of the browser |
| appVersion | Returns the platform and version of the browser |
| browserLanguage | Returns the current browser language |
| cookieEnabled | Returns a Boolean value that specifies whether cookies are enabled in the browser |
| cpuClass | Returns the CPU class of the browser's system |
| onLine | Returns a Boolean value that specifies whether the system is in offline mode |
| platform | Returns the operating system platform |
| systemLanguage | Returns the default language used by the OS |
| userAgent | Returns the value of the user-agent header sent by the client to the server |
| userLanguage | Returns the OS' natural language setting |

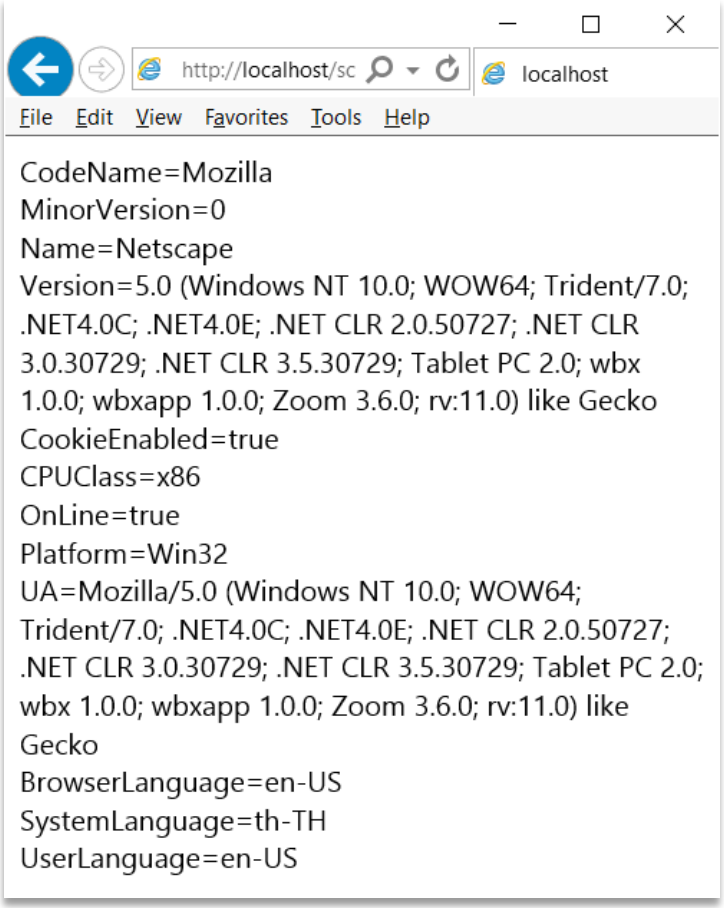
# Navigator Object

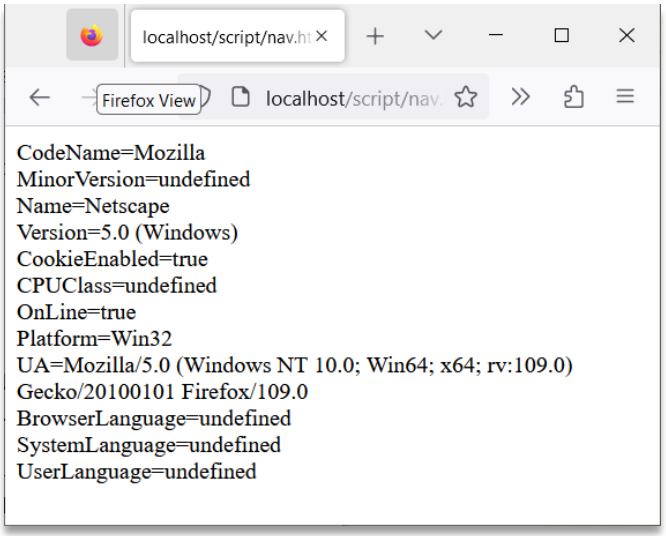
- Browser Detection

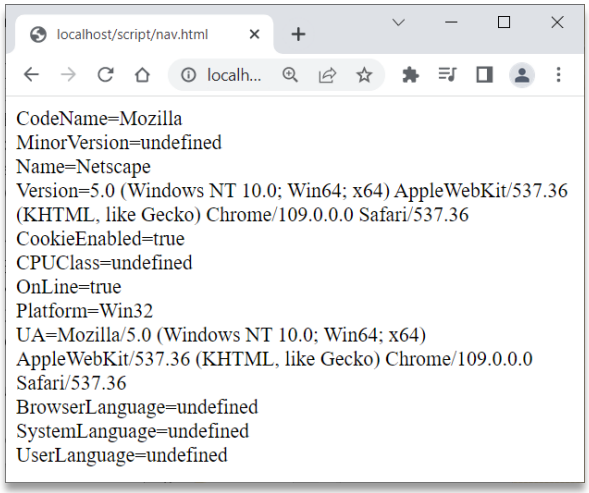
```
<html>
<body>
<script type="text/javascript">
var x = navigator;
document.write("CodeName=" + x.appCodeName);
document.write("<br />");
document.write("MinorVersion=" + x.appMinorVersion);
document.write("<br />");
document.write("Name=" + x.appName);
document.write("<br />");
document.write("Version=" + x.appVersion);
document.write("<br />");
document.write("CookieEnabled=" + x.cookieEnabled);
document.write("<br />");
document.write("CPUClass=" + x.cpuClass);
document.write("<br />");
document.write("OnLine=" + x.onLine);
document.write("<br />");
document.write("Platform=" + x.platform);
document.write("<br />");
document.write("UA=" + x.userAgent);
document.write("<br />");
document.write("BrowserLanguage=" + x.browserLanguage);
document.write("<br />");
document.write("SystemLanguage=" + x.systemLanguage);
document.write("<br />");
document.write("UserLanguage=" + x.userLanguage);
</script>
</body></html>
```

# Navigator Object

- Browser Detection



# Screen Object

- Properties

| Property | Description |
|---|---|
| availHeight | Returns the height of the display screen (excluding the Windows Taskbar) |
| availWidth | Returns the width of the display screen (excluding the Windows Taskbar) |
| bufferDepth | Sets or returns the bit depth of the color palette in the off-screen bitmap buffer |
| colorDepth | Returns the bit depth of the color palette on the destination device or buffer |
| deviceXDPI | Returns the number of horizontal dots per inch of the display screen |
| deviceYDPI | Returns the number of vertical dots per inch of the display screen |
| fontSmoothingEnabled | Returns whether the user has enabled font smoothing in the display control panel |
| height | The height of the display screen |
| logicalXDPI | Returns the normal number of horizontal dots per inch of the display screen |
| logicalYDPI | Returns the normal number of vertical dots per inch of the display screen |
| pixelDepth | Returns the color resolution (in bits per pixel) of the display screen |
| updateInterval | Sets or returns the update interval for the screen |
| width | Returns width of the display screen |

# History Object

- Properties

| Property | Description |
|---|---|
| length | Returns the number of elements in the history list |

- Methods

| Method | Description |
|---|---|
| back() | Loads the previous URL in the history list |
| forward() | Loads the next URL in the history list |
| go() | Loads a specific page in the history list |

# Location Object

- Properties & Methods

| Property | Description |
|---|---|
| hash | Sets or returns the URL from the hash sign (#) |
| host | Sets or returns the hostname and port number of the current URL |
| hostname | Sets or returns the hostname of the current URL |
| href | Sets or returns the entire URL |
| pathname | Sets or returns the path of the current URL |
| port | Sets or returns the port number of the current URL |
| protocol | Sets or returns the protocol of the current URL |
| search | Sets or returns the URL from the question mark (?) |

| Method | Description |
|---|---|
| assign() | Loads a new document |
| reload() | Reloads the current document |
| replace() | Replaces the current document with a new one |

# Document Object

- Properties

| Property | Description |
|---|---|
| body | Gives direct access to the <body> element |
| cookie | Sets or returns all cookies associated with the current document |
| domain | Returns the domain name for the current document |
| lastModified | Returns the date and time a document was last modified |
| referrer | Returns the URL of the document that loaded the current document |
| title | Returns the title of the current document |
| URL | Returns the URL of the current document |

# Document Object

- Methods

| Method | Description |
|---|---|
| close() | Closes an output stream opened with the document.open() method, and displays the collected data |
| getElementById() | Returns a reference to the first object with the specified id |
| getElementsByName() | Returns a collection of objects with the specified name |
| getElementsByTagName() | Returns a collection of objects with the specified tagname |
| open() | Opens a stream to collect the output from any document.write() or document.writeln() methods |
| write() | Writes HTML expressions or JavaScript code to a document |
| writeln() | Identical to the write() method, with the addition of writing a new line character after each expression |

# DOM Events

- DOM (Document Object Model) events allow event-driven programming languages to register various event handlers/listeners on the element nodes inside a DOM tree

# Event Handling

- Event Model
  - DOM Level 0
    - Inline model
    - Traditional model
  - DOM Level 2
  - Microsoft-specific model

# Event Handling

- DOM Level 0
  - Inline model
    - the inline model, event handlers are added as attribute of element

```
<html>
<head>
  <script type="text/javascript">
    function helloWorld( name )
    {
        window.alert( "Hello " + name );
    }

    function removeHandler(object)
    {
        object.onclick = null;
    }
  </script>
</head>
<body>
  <button onclick="helloWorld('Joe'); removeHandler(this);">Hello</button>
</body>
<html>
```

# Event Handling

- DOM Level 0
  - Traditional model
    - the traditional model, event handlers can be added/removed by scripts

```
<html>
<head>
<script type="text/javascript">
        function helloWorld() {
                window.alert( "Hello World" );
                // Remove the event handler just added
                helloButton.onclick = null;
        }
        window.onload = function() {
                // Add an event handler
                helloButton.onclick = helloWorld;
        }
</script>
</head>
<body>
  <button id="helloButton">Hello</button>
</body>
</html>
```

# Event Handling

- DOM Level 2

| Name | Description | Argument type | Argument name |
|---|---|---|---|
| addEventListener | Allows the registration of event listeners on the event target. | DOMString | type |
| | | EventListener | listener |
| | | boolean | useCapture |
| removeEventListener | Allows the removal of event listeners from the event target. | DOMString | type |
| | | EventListener | listener |
| | | boolean | useCapture |
| dispatchEvent | Allows to send the event to the subscribed event listeners. | Event | evt |

# Event Handling

- DOM Level 2
  - To prevent an event to bubble, developers must call the "stopPropagation()" method of the event object.
  - To prevent the default action of the event to be called, developers must call the "preventDefault" method of the event object.

# Event Handling

- DOM Level 2

```
<html>
<head>
  <script type="text/javascript">
    function helloWorld()
    {
        window.alert( "Hello World" );
    }

  // Add an event handler
    window.addEventListener( "load", helloWorld, false );  // bubbling phase

  // Add another event handler
    document.addEventListener( "click", helloWorld, true );  // capture phase

  // Remove the event handler just added
    document.removeEventListener( "click", helloWorld, true );
  </script>
</head>
<body>
  <p>Hello World!</p>
</body>
</html>
```

# Event Handling

- Microsoft-specific model

| Name | Description | Argument type | Argument name |
|---|---|---|---|
| attachEvent | Similar to W3C's addEventListener method. | String | sEvent |
| | | Pointer | fpNotify |
| detachEvent | Similar to W3C's removeEventListener method. | String | sEvent |
| | | Pointer | fpNotify |
| fireEvent | Similar to W3C's dispatchEvent method. | String | sEvent |
| | | Event | oEventObject |

# Event Handling

- Microsoft-specific model
  - To prevent an event bubbling, developers must set the event's "cancelBubble" property.
  - To prevent the default action of the event to be called, developers must set the event's "returnValue" property.
  - The **this** keyword refers to the global **window** object.

# Event Handling

- Microsoft-specific model

```
<html>
<head>
<script type="text/javascript">
        function helloWorld() {
                window.alert( "Hello World" );
                // Remove the event handler just added
                helloButton.detachEvent( "onclick", helloWorld );
        }
        function init() {
                helloButton.attachEvent( "onclick", helloWorld );
        }
  // Add an event handler
   window.attachEvent( "onload", init);
  </script>
</head>
<body>
  <button id="helloButton">Hello</button>
</body>
</html>
```

# Event Handling

- Using Event Handler
  - Internet Explorer
    - object.attachEvent(“event”,function);
  - Other Browser
    - object.addEventListener(“event”,function,capture);

# Event Handling

- Remove Event Handler
  - Internet Explorer
    - object.detachEvent(“event”,function);
  - Other Browser
    - object.removeEventListener(“event”,function,capture);

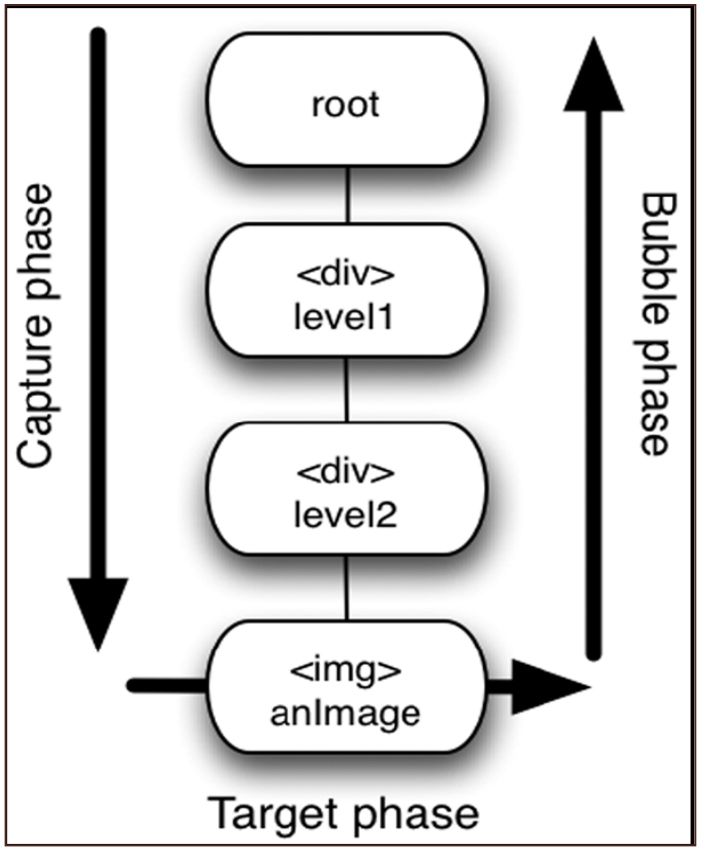
# Event Handling

- Stop Event
  - Stop current event that occurred
  - Internet Explorer
    - event.cancelBubble = true;
  - Other Browser
    - event.stopPropagation();

# Event Flow

- Event Propagation

```
<div id="level1">
        <div id="level2">
                <img
src="image.jpg" id="anImage"/>
        </div>
</div>
```



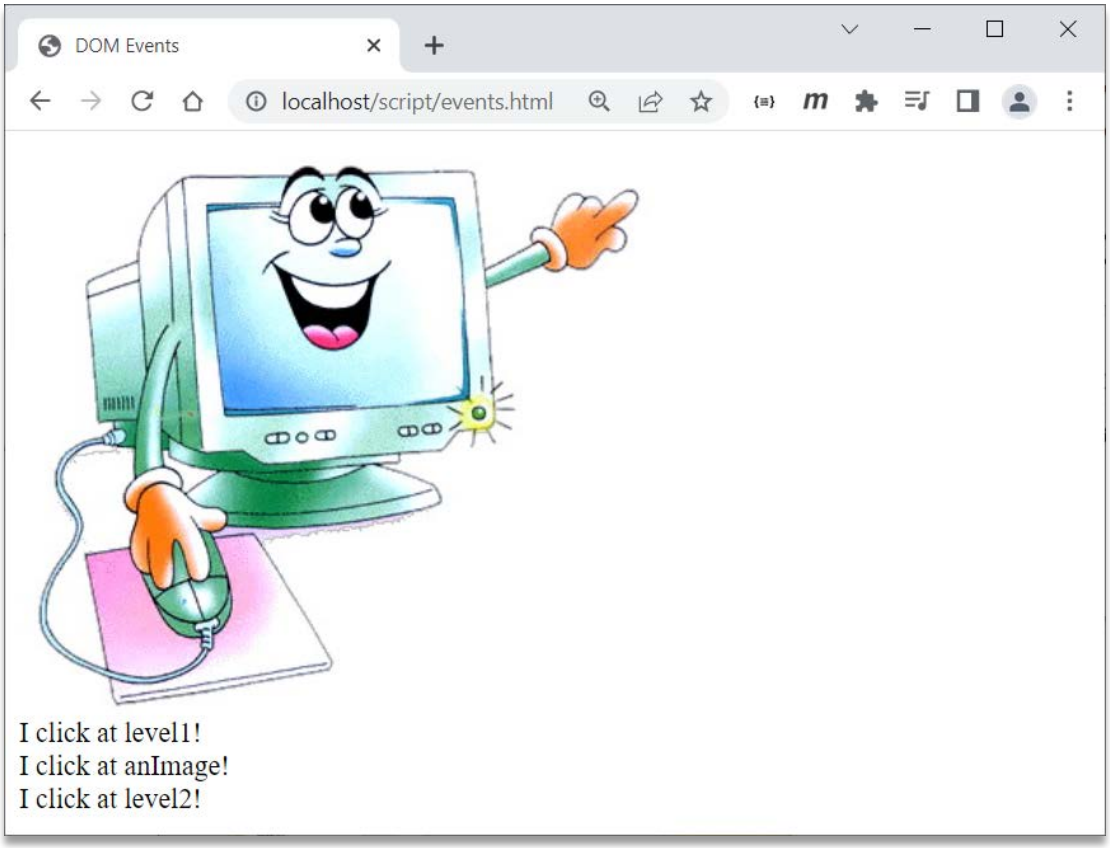
# Event Flow

- Event Propagation

```
<html>
<head>
<title>DOM Events</title>
<script type="text/javascript">
        window.onload = function() {
                document.getElementById('anImage').addEventListener('click', react, false);
                document.getElementById('level1').addEventListener('click', react, true);
                document.getElementById('level2').addEventListener('click', react, false);
        }
        function react(event) {
                document.getElementById('info').innerHTML += 'I click at ' + event.currentTarget.id + '!<br />';
        }
</script>
</head>
<body>
        <div id="level1">
                <div id="level2">
                        <img src="image.jpg" id="anImage"/>
                </div>
        </div>
        <div id="info"></div>
</body>
</html>
```

# Event Flow

- Event Propagation



# Event Object

- Event Properties

| Type | Name | Description |
|---|---|---|
| DOMString | type | The name of the event (case-insensitive). |
| EventTarget | target | Used to indicate the EventTarget to which the event was originally dispatched. |
| EventTarget | currentTarget | Used to indicate the EventTarget whose EventListeners are currently being processed. |
| unsigned short | eventPhase | Used to indicate which phase of event flow is currently being evaluated. |
| boolean | bubbles | Used to indicate whether or not an event is a bubbling event. |
| boolean | cancelable | Used to indicate whether or not an event can have its default action prevented. |
| DOMTimeStamp | timeStamp | Used to specify the time (in milliseconds relative to the epoch) at which the event was created. |

# Event Object

- Event Methods

| Name | Argument type | Argument name | Description |
|---|---|---|---|
| stopPropagation | | | To prevent further propagation of an event during event flow. |
| preventDefault | | | To cancel the event if it is cancelable, meaning that any default action normally taken by the implementation as a result of the event will not occur. |
| initEvent | DOMString | eventTypeArg | Specifies the event type. |
| | boolean | canBubbleArg | Specifies whether or not the event can bubble. |
| | boolean | cancelableArg | Specifies whether or not the event's default action can be prevented. |

# Timing Event

- setTimeout()
  - execute a code some time in the future

```
var t=setTimeout("javascript statement",milliseconds);
```

- clearTimeout()
  - cancel the setTimeout()

```
clearTimeout(setTimeout_variable)
```

# Timing Event

- setInterval()
  - keeps triggering *expression* again and again

```
var t = setInterval( expression, interval );
```

- clearInterval()
  - cancel the setInterval()

```
clearInterval(setInterval_variable)
```

# Cross Domain Communication

- postMessage
  - to send the message

```
window.postMessage(msg,"*");
```

- onmessage
  - to receive and process message

```
window.onmessage = function(e) {
        console.log(e.data);
}
```

```
window.addEventListener("message",function(e) {
        console.log(e.data);
});
```

# Example

- window – dialog.html

```
<html>
	<head>
		<title>Dialog</title>
		<script>
			function openWindow() {
				window.open("dialog.html","my_dialog");
			}
			function openDialog() {
				var winWidth = 500;
				var winHeight = 350;
				var sw = window.screen.availWidth;
				var sh = window.screen.availHeight;
				var wx = (sw - winWidth) / 2;
				var wy = (sh - winHeight) / 2;
				var features =
"top="+wy+",left="+wx+",width="+winWidth+",height="+winHeight+",toobar=no,menubar=0,location=no,directories=no,status=no,scrollbars=no
,resizable=yes";
				var awin = window.open("dialog.html","my_window",features);

				awin.opener = self;
				return awin;
			}
		</script>
	</head>
	<body>
		<button onclick="alert('Hello')">Alert</button>
		<button onclick="confirm('Close?')">Confirm</button>
		<button onclick="prompt('Input','default')">Prompt</button>
		<button onclick="openWindow()">Open Window</button>
		<button onclick="openDialog()">Open Dialog</button>
	</body>
</html>
```

# Example

- CSS – dyna.html (1)

```
<html>
<head>
<title>DOM Style</title>
<style>
button {
                    border: solid #aaaaaa 1px;
                    color: blue;
                    cursor: hand;
}
textarea {
                    width: 400px;
                    height: 100px;
}
.span {
                    background-color: #eeeeee;
                    padding: 3px;
}
</style>
<script language="JavaScript">
function changeFontFamily() {
                    var sel = document.getElementById("ffm");
                    var text = document.getElementById("text");
                    text.style.fontFamily = sel.options[sel.selectedIndex].value;
}
function changeFontSize() {
                    var sel = document.getElementById("fsz");
                    var text = document.getElementById("text");
                    text.style.fontSize = sel.options[sel.selectedIndex].value+"pt";
}
```

# Example

- CSS – dyna.html (2)

```
function fontBold() {
		var text = document.getElementById("text");
		if(text.style.fontWeight=="bold") {
			text.style.fontWeight = "normal";
		} else {
			text.style.fontWeight = "bold";
		}
		toggle();
}
function fontStyle() {
		var text = document.getElementById("text");
		if(text.style.fontStyle=="italic") {
			text.style.fontStyle = "normal";
		} else {
			text.style.fontStyle = "italic";
		}
		toggle();
}
function textDecoration() {
		var text = document.getElementById("text");
		if(text.style.textDecoration=="underline") {
			text.style.textDecoration = "none";
		} else {
			text.style.textDecoration = "underline";
		}
		toggle();
}
```

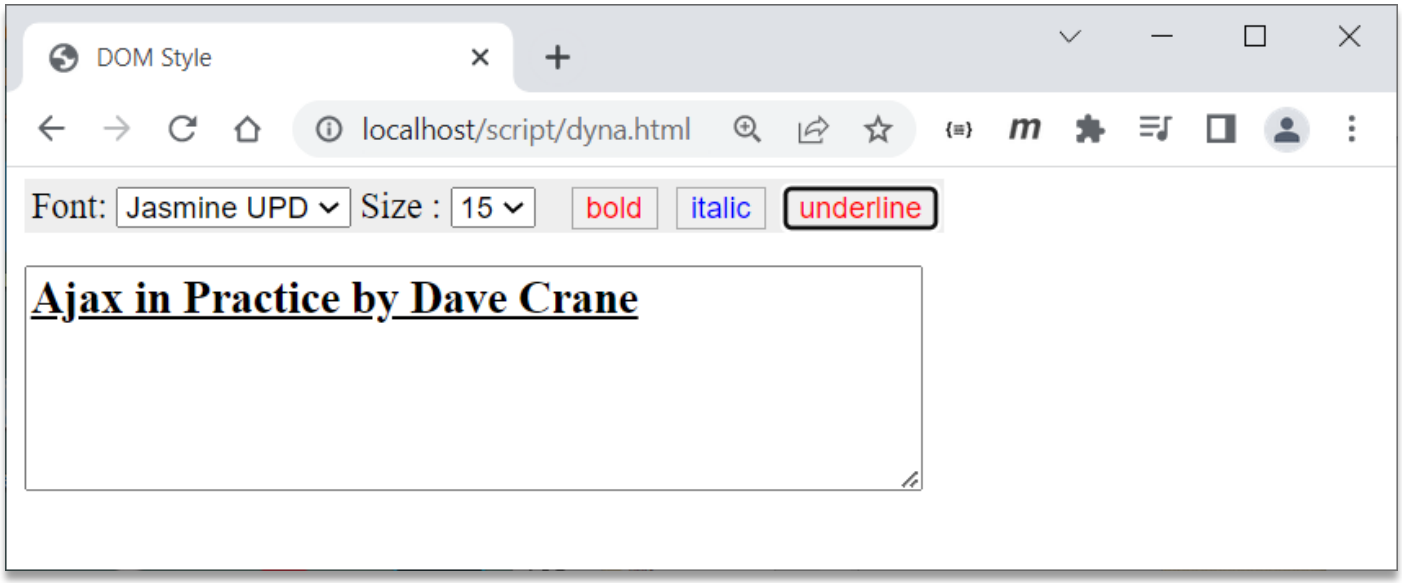
# Example

- CSS – dyna.html (3)

```
function toggle() {
                var src = event.srcElement;
                if(src.style.color!="red") {
                                src.style.color = "red";
                } else {
                                src.style.color = "blue";
                }
}
</script>
</head>
<body>
<span class="span">
                Font: <select id="ffm" onchange="changeFontFamily()">
                                <option value="tahoma">Tahoma</option>
                                <option value="cordia new">Cordia New</option>
                                <option value="jasminUPC">Jasmine UPD</option>
                                                                <select>
                Size : <select id="fsz" onchange="changeFontSize()">
                                <option value="10">10</option>
                                <option value="15">15</option>
                                <option value="20">20</option>
                </select>
                  
                <button onclick="fontBold()">bold</button> 
                <button onclick="fontStyle()">italic</button> 
                <button onclick="textDecoration()">underline</button>
</span><p>
<textarea id="text">Ajax in Practice by Dave Crane</textarea>
</body></html>
```

# Example

- CSS – dyna.html



# Example

- cross-domain – frame.html

```
<html>
        <head>
                <title>Post Message</title>
                <script language="JavaScript">
                function talkToFrameClick() {
                    msgframe.window.postMessage("Hello World", "*");
                }
                window.onmessage = function(e) {
                    console.log("on message in parent : "+e.data);
                }
                </script>
        </head>
        <body>
                <button id="mybutton" onclick="talkToFrameClick()">Talk
To Frame</button>
                <iframe id="msgframe" name="msgframe"
src="frame_msg.html" width="100%" height="100px"></iframe>
        </body>
</html>
```

# Example

- cross-domain – frame_msg.html

```
<html>
        <head>
                <title>Post Message</title>
                <script>
                function sentToParent() {
                    window.parent.postMessage("Hello Message", "*");
                }
                window.onmessage = function(e) {
                    console.log("on message in frame : "+e.data);
                    mytext.value = e.data;
                }
                </script>
        </head>
        <body>
                <input type="text" id="mytext" value="Hello
Message"></input><br/>
                <button id="mybutton" onclick="sentToParent()">Send To
Parent</button>
        </body>
</html>
```

# JSON

- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format
  - It is easy for humans to read and write
  - It is easy for machines to parse and generate
  - It is based on subset of the JavaScript Programming Language
  - JSON is a text format that is completely language independent
  - JSON and ideal data-interchange language

# JSON Structure

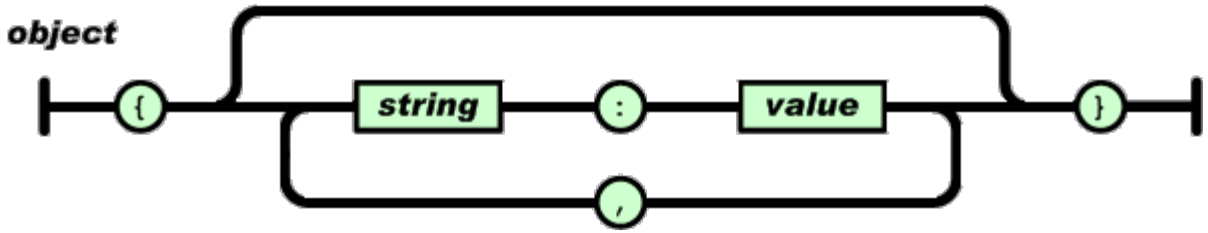
- JSON is built on two structures
  - A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array
  - An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence

# JSON Forms

- JSON take on these forms
  - An *object* is an unordered set of name/value pairs
  - An object begins with { (left brace) and ends with } (right brace)
  - Each name is followed by : (colon) and the name/value pairs are separated by , (comma)

# JSON Forms

- Object form

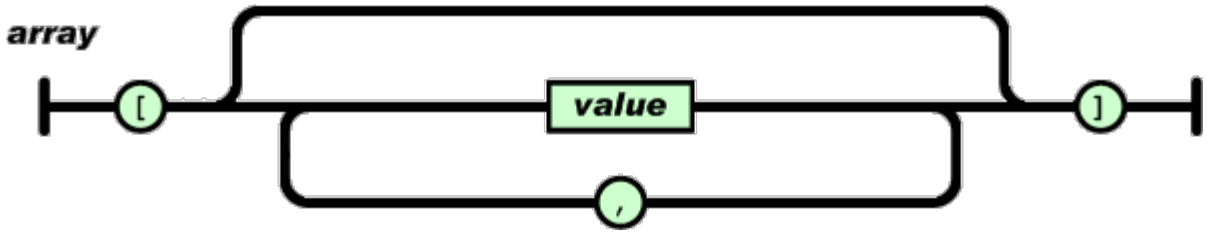


# JSON Forms

- An *array* is an ordered collection of values.
- An array begins with [ (left bracket) and ends with ] (right bracket)
- Values are separated by , (comma)

# JSON Forms

- Array form

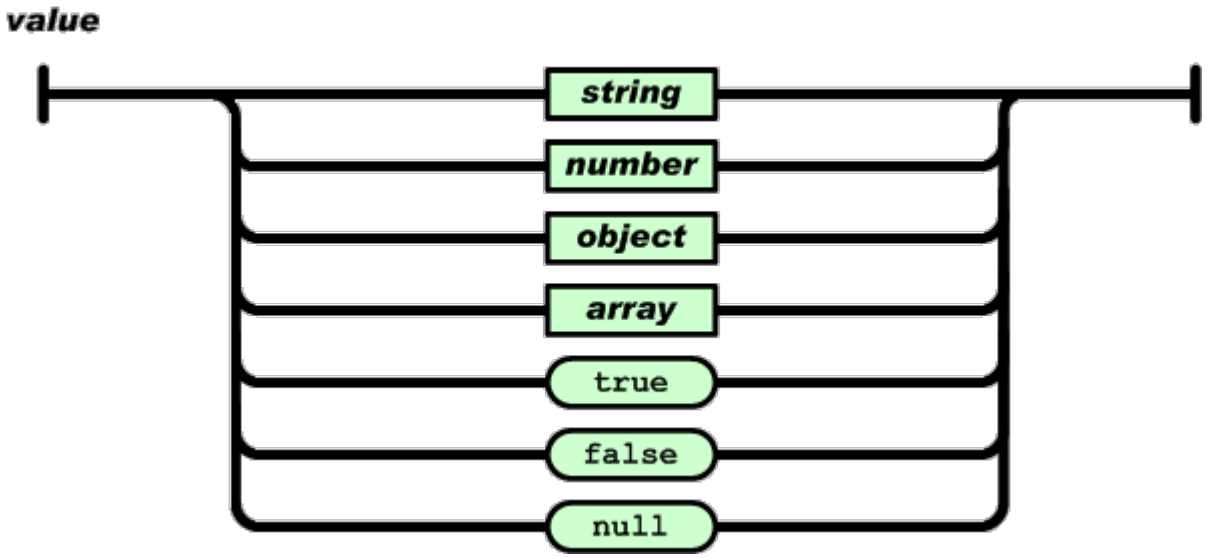


# JSON Forms

- A *value* can be a *string* in double quotes, or a *number*, or true or false or null, or an *object* or an *array*
- These structures can be nested

# JSON Forms

- Value form

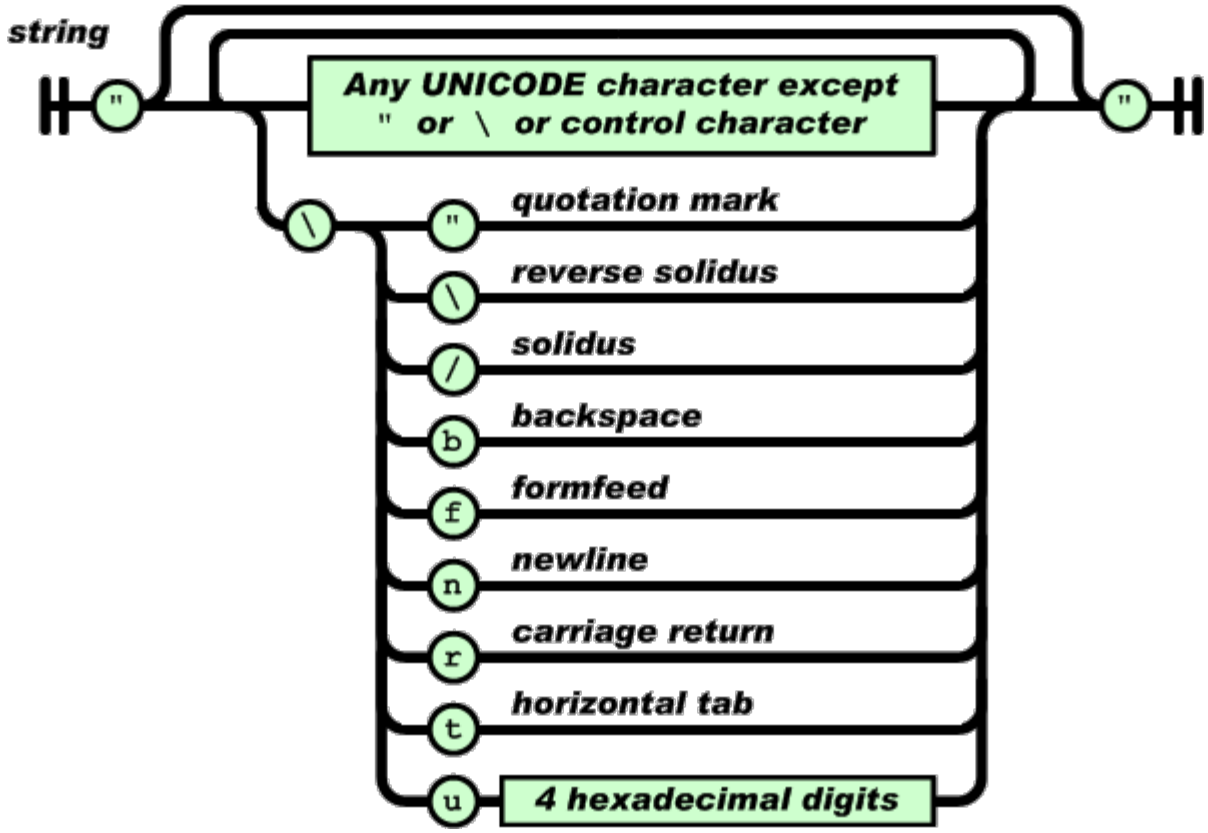


# JSON Forms

- A *string* is a collection of zero or more Unicode characters, wrapped in double quotes, using backslash escapes
- A character is represented as a single character string
- A string is very much like a C or Java string

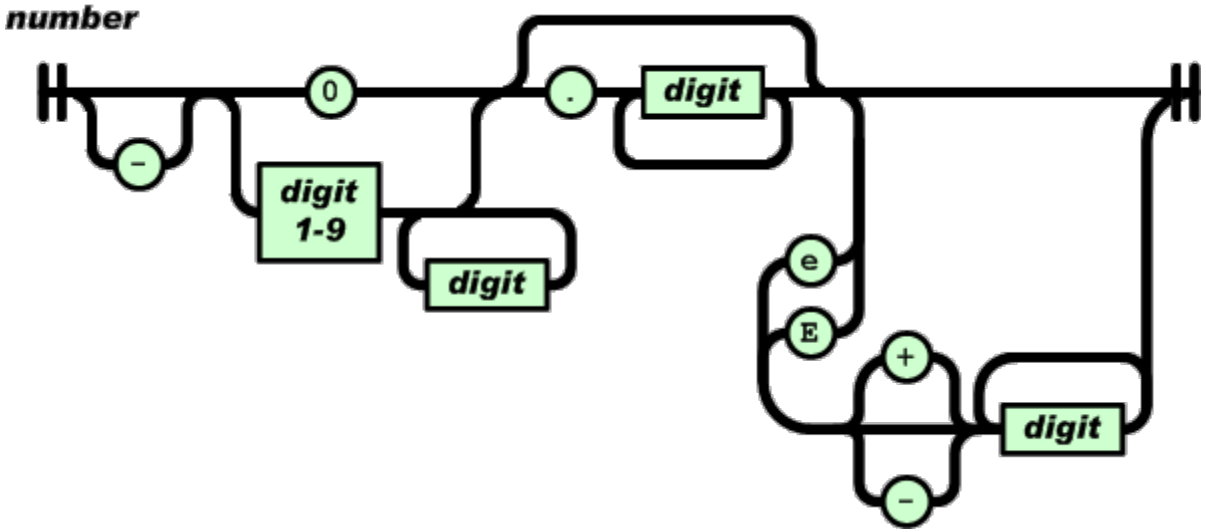
# JSON Forms

- String form



# JSON Forms

- A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used



# JSON In JavaScript

- JSON is a subset of the object literal notation of JavaScript

```
var jsonObject = {"cars": [
                {"year": 2001, "type": "Toyota", "model": "Corolla"},
                {"year": 2002, "type": "Toyota", "model": "Vios"},
                {"year": 2003, "type": "Toyota", "model": "Altis"}
        ]
};
```

  - This example, an object is created containing a single member "cars", which contains an array containing three objects, each containing "year", "type", and "model" members

```
jsonObject.cars[0].model // "Corolla"
```

  - Members can be retrieved using dot or subscript operators

# JSON Conversion

- To convert a JSON text into an object, you can use the eval() function

```
var jsonObject = eval('(' + jsonText + ')');
```

# JSON Data Array

- json01.html

```
<HTML>
<HEAD>
<TITLE> Test JSON</TITLE>
<script language="JavaScript">
function parseClick() {
		var ary = eval('('+myinput.value+')');
		var txt = "";
		for(var i=0;i<ary.length;i++) {
			txt += (ary[i]*5) + '<br>';
		}
		resultLayer.innerHTML = txt;
}
 </script>
 </HEAD>
 <BODY>
		<input type="text" id="myinput" value="[100,200,300,400,500]"></input>
		<input type="button" id="jsonbtn" onclick="parseClick()" value="Parse"></input>
		<div id="resultLayer"></div>
 </BODY>
</HTML>
```

# JSON Data Structure Array

- json02.html - 1

```
<HTML>
<HEAD>
<TITLE> Test JSON</TITLE>
<script language="JavaScript">
function parseClick() {
                var ary = eval('('+mytext.value+')');
                var custs = ary.CUSTOMERS;
                var txt = "";
                for(var i=0;i<custs.length;i++) {
                                txt += custs[i].CUSTOMERID + ' - ' +custs[i].NAME+'
'+custs[i].SURNAME+'<br>';
                }
                resultLayer.innerHTML = txt;
}
 </script>
 </HEAD>
 <BODY>
                <input type="button" id="parsebtn" onclick="parseClick()"
value="Parse"></input><br>
```

# JSON Data Structure Array

- json02.html - II

```
                <textarea id="mytext" cols=50 rows=10>
{
  "CUSTOMERS": [
     { "CUSTOMERID": "100",  "NAME": "John", "SURNAME":"Doe" },
     { "CUSTOMERID": "200",  "NAME": "Jane", "SURNAME":"Air" },
     { "CUSTOMERID": "300", "NAME": "Jack", "SURNAME":"Sparow" }
   ]
}
                </textarea><br>
                <div id="resultLayer"></div>
 </BODY>
</HTML>
```

# JSON Parameters & Arguments

- json03.html

```
<script language="JavaScript">

var str = '{ "CUSTOMERID": “100", "NAME":“John", "SURNAME":“Doe" }’;

function displayCustomer1(custId,custName,custSurname) {
                resultLayer.innerHTML = custId+" - "+custName+" "+custSurname;
}
function displayCustomer2(cust) {
                resultLayer.innerHTML = cust.CUSTOMERID+" - "+cust.NAME+" "+cust.SURNAME;
}
function call1Click() {
                var customer = eval('('+str+')');
                displayCustomer1(customer.CUSTOMERID,customer.NAME,customer.SURNAME);
}
function call2Click() {
                var customer = eval('('+str+')');
                displayCustomer2(customer);
}
function call3Click() {
                displayCustomer2( { CUSTOMERID:'200', NAME:'Supara', SURNAME:’Air'} );
}
 </script>
```

# JSON Converter

- JSON Parser
  - Recognize only JSON text, rejecting all scripts
  - Faster than eval

```
var jsonObject = JSON.parse(jsonText, reviver);
```

- JSON stringifier
  - Does not support cyclic data structures

```
var jsonText = JSON.stringify(jsonObject, replacer);
```

# JSON Converter

- json04.html

```
<HTML>
<HEAD>
<TITLE>Test JSON</TITLE>
<script language="JavaScript">
function parseClick() {
                var jsonObj = {NAME:“John",SURNAME:“Doe"};
                var jsonStr = JSON.stringify(jsonObj);
                StatusLayer.innerHTML = jsonStr;
                var cust = JSON.parse(jsonStr);
                LogLayer.innerHTML = cust.NAME+" "+cust.SURNAME;
}
 </script>
 </HEAD>
 <BODY>
<input type="button" value="Parse" onclick="parseClick()"><br>
<div id="StatusLayer"></div>
<div id="LogLayer"></div>
 </BODY>
</HTML>
```

# AJAX

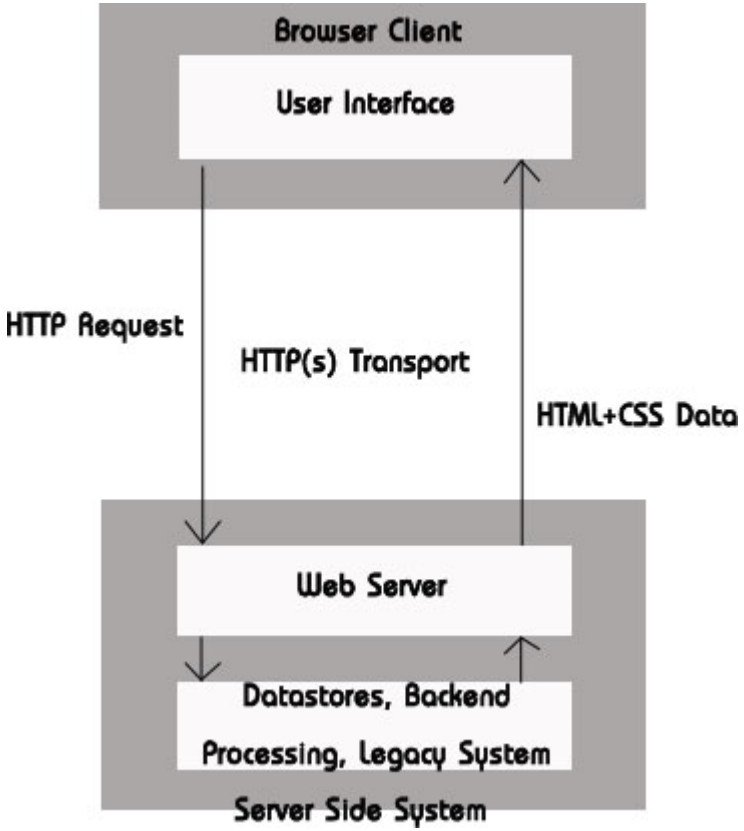
- Ajax is a way of developing Web applications that combines:
  - XHTML and CSS standards based presentation
  - Interaction with the page through the DOM
  - Data interchange with XML and XSLT
  - Asynchronous data retrieval with XMLHttpRequest
  - JavaScript to tie it all together

# Why is AJAX

- When it comes to web applications, AJAX offers two crucial advantages:
  - It is efficient
    - only a part of the web page that needs to be modified is being modified. With traditional server-side scripting you send the entire new page to the browser
  - It is lightweight
    - only small amounts of data (in the XML form) are being exchanged through the Internet, making your web applications very fast

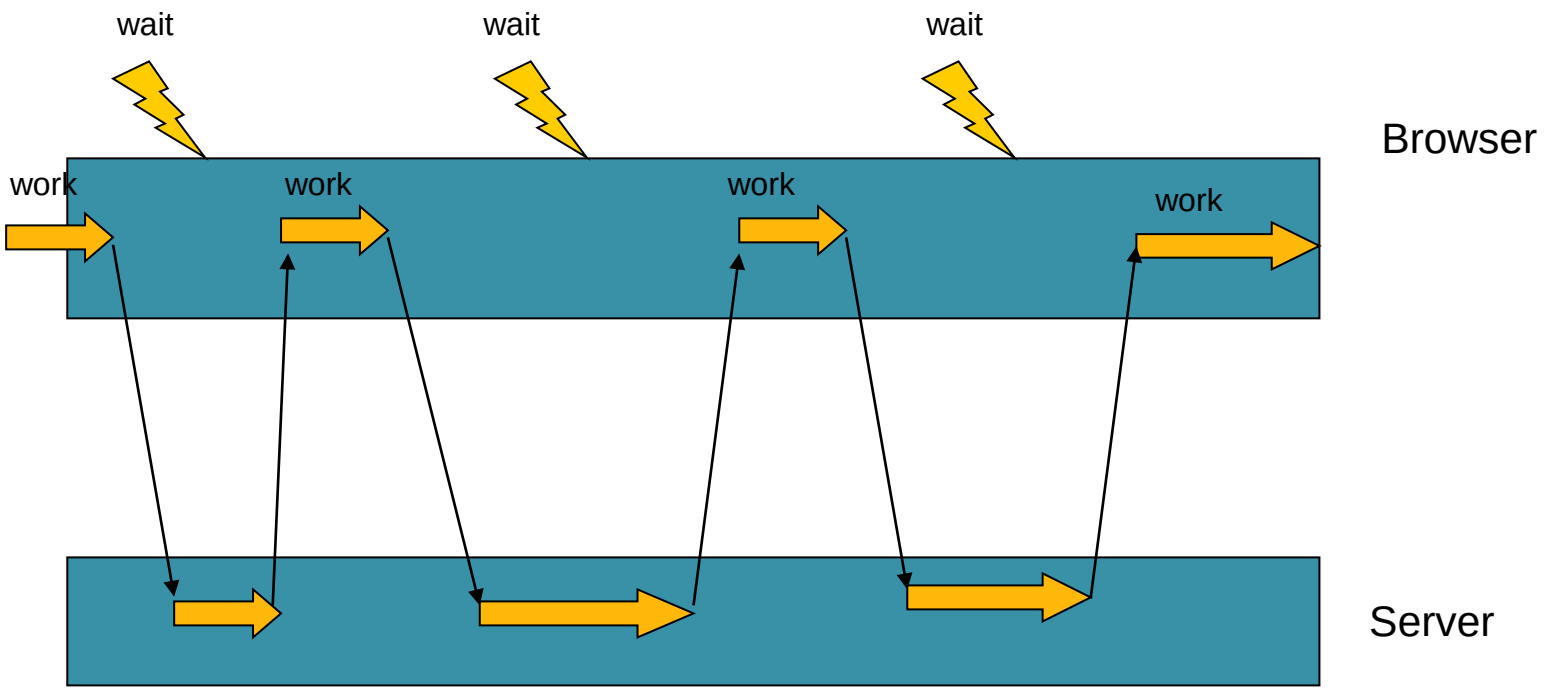
# AJAX

- WEB Interaction
  - Traditional



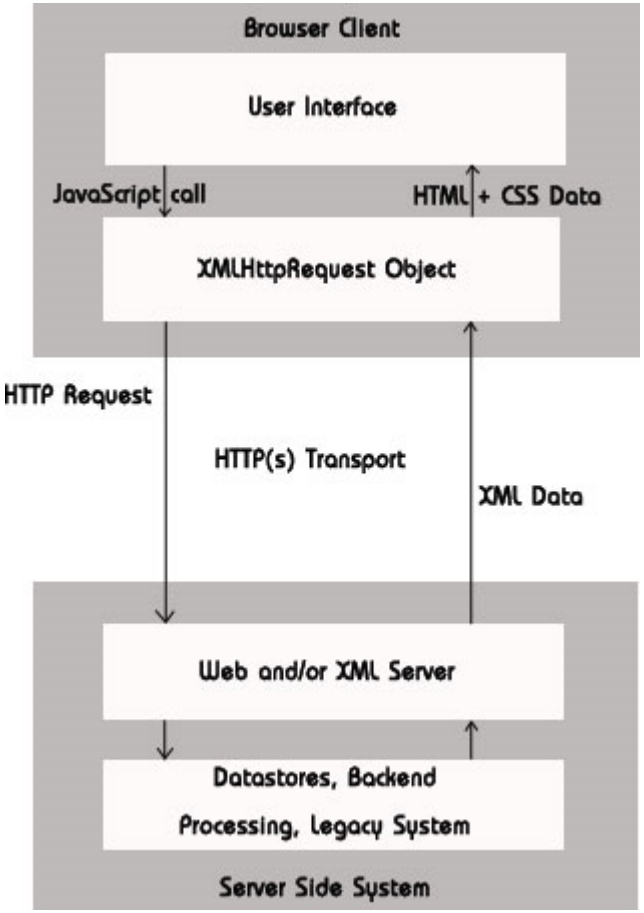
# AJAX

- WEB Interaction
  - Traditional



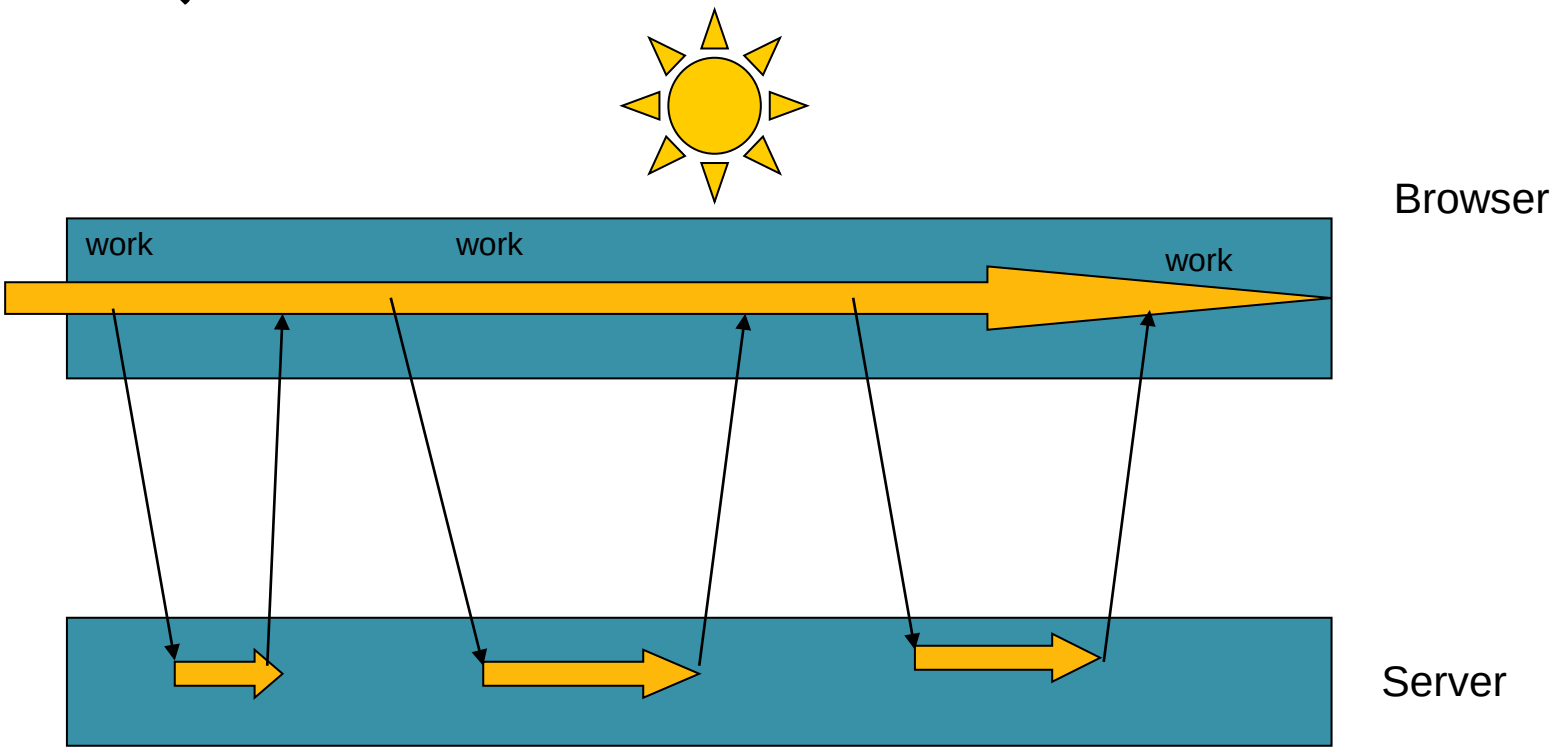
# AJAX

- WEB Interaction
  - Ajax



# AJAX

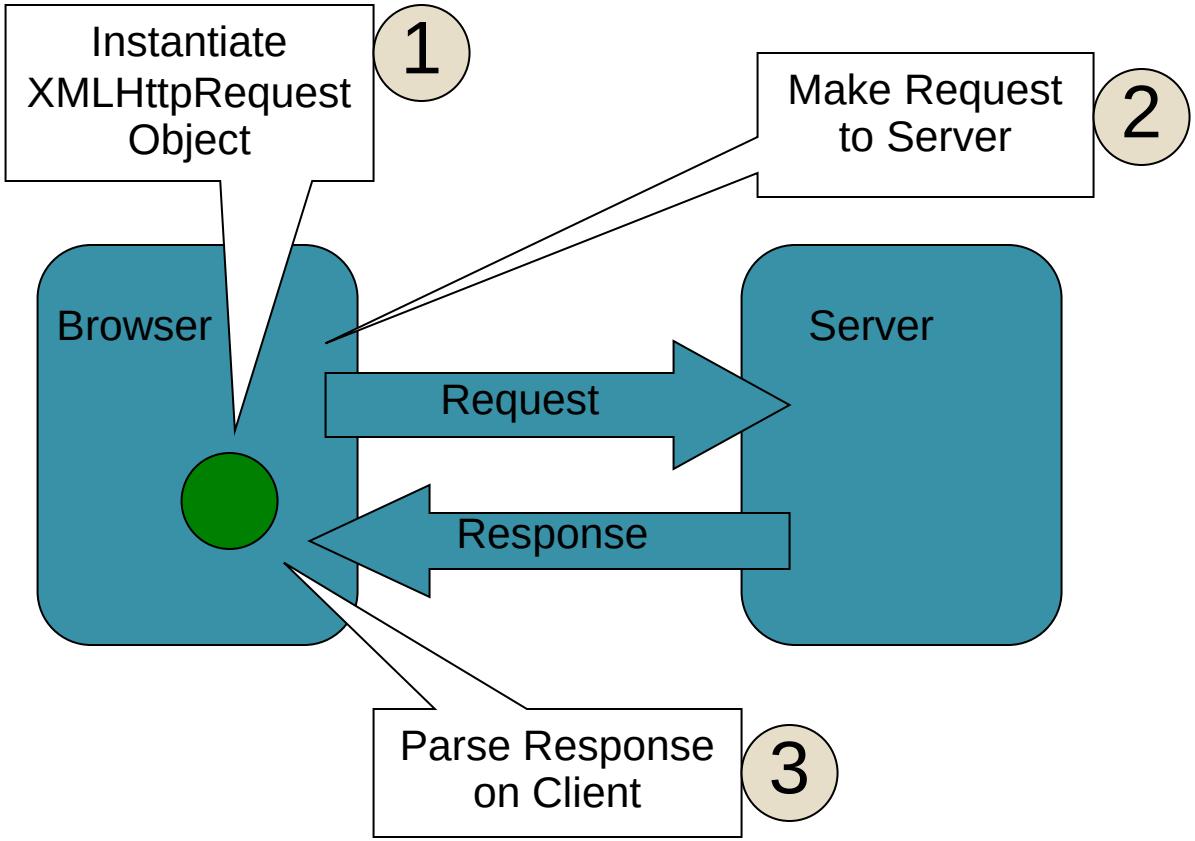
- WEB Interaction
  - Ajax



# AJAX Application Characteristics

- Continuous Feel
  - Ajax offers a smooth ride all the way the server is only telling the screen what changed rather than having it redraw the whole screen from scratch
- Real-Time Updates
  - Ajax application poll the server every few seconds, so it's capable of updating any information directly on the parts of the page that need changing and the rest of the page is unaffected
- Graphical Interaction
  - Ajax represents a transition into the world of GUI controls visible on present-day desktops

# AJAX Application



# AJAX Application

1. Instantiate

```
var req = new XMLHttpRequest();
```

2. Make Request

```
req.open("GET","ajax/getdata.jsp",true);
req.onreadystatechange = parseResponse;
req.send("");
```

3. Parse Response

```
function parseResponse() {
 if (req.readyState == 4) {
   if (req.status == 200) {
     mydiv.innerHTML = req.responseText;
   } else {
     mydiv.innerHTML= " Error: "+ req.status + "\n"
+req.statusText;
   }
 }
}
```

# XMLHttpRequest Object

- A JavaScript Class that lets you make asynchronous HTTP requests from JavaScript
- Make an HTTP request from a JavaScript event
- A call back JavaScript function is invoked at each state of the HTTP request and response

# XMLHttpRequest Object

- IE does it different
  - The above example however won't work on IE
    - req = new ActiveXOjbect("Microsoft.XMLHTTP");

# XMLHttpRequest Object

- Cross Browser Ajax

```
var req;
function loadXMLDoc(url) {
        req = false; // branch for native XMLHttpRequest object
        if(window.XMLHttpRequest) {
                try {
                        req = new XMLHttpRequest();
                } catch(e) {
                        req = false;
                }
        // branch for IE/Windows ActiveX version
        } else if(window.ActiveXObject) {
                try {
                        req = new ActiveXObject("Msxml2.XMLHTTP");
                } catch(e) {
                        try {
                                req = new ActiveXObject("Microsoft.XMLHTTP");
                        } catch(e) {
                                req = false;
                        }
                }
        }
        if(req) {
                req.onreadystatechange = processReqChange;
                req.open("GET", url, true);
                req.send("");
        }
}
```

# XMLHttpRequest Object

- Ajax Factory - I

```
var msHttp = new Array('Msxml2.XMLHTTP.5.0',
                       'Msxml2.XMLHTTP.4.0',
                       'Msxml2.XMLHTTP.3.0',
                       'Msxml2.XMLHTTP',
                      'Microsoft.XMLHTTP',
                       'Msxml.DOMDocument',
                       'Msxml2.DOMDocument',
                       'Microsoft.XMLDOM');
function AJAXFactory(){
                      var req;
                      this.HTTPRequest = function(){
                                        if(window.XMLHttpRequest){
                                                          return new XMLHttpRequest();
                                        }else if(window.ActiveXObject){
                                                          for(var i = 0 ; i < msHttp.length ; i++){
                                                                            try{
                                                                                              return new ActiveXObject(msHttp[i]);
                                                                            }catch(e){}
                                                          }
                                        }
                                        throw new Error("Could not instantiate XMLHttpRequest");
                      };
```

# XMLHttpRequest Object

- Ajax Factory - II

```
		this.sendRequest = function(Url,Prms,HttpMethod,ProcessState){
			if(req==null){
				req = this.HTTPRequest();
			}
			if(req){
				req.open(HttpMethod,Url,true);
				req.onreadystatechange = function() {
ProcessState(req); }
				req.setRequestHeader("Content-Type", "application/x-
www-form-urlencoded");
				req.send(Prms);
			}
			return req;
		}
}
```

# XMLHttpRequest Object

- Methods

| Method | Description |
|---|---|
| `abort()` | Stops the current request |
| `getAllResponseHeaders()` | Returns complete set of headers (labels and values) as a string |
| `getResponseHeader("headerLabel")` | Returns the string value of a single header label |
| `open("method", "URL"[, asyncFlag[, "userName"[, "password"]]])` | Assigns destination URL, method, and other optional attributes of a pending request |
| `send(content)` | Transmits the request, optionally with postable string or DOM object data |
| `setRequestHeader("label", "value")` | Assigns a label/value pair to the header to be sent with a request |

# XMLHttpRequest Object

- Properties

| Property | Description |
|---|---|
| `onreadystatechange` | Event handler for an event that fires at every state change |
| `readyState` | Object status integer:<br>0 = uninitialized<br>1 = loading<br>2 = loaded<br>3 = interactive<br>4 = complete |
| `responseText` | String version of data returned from server process |
| `responseXML` | DOM-compatible document object of data returned from server process |
| `status` | Numeric code returned by server, such as 404 for "Not Found" or 200 for "OK" |
| `statusText` | String message accompanying the status code |

# XMLHttpRequest Object

- reponseXML
  - May not work due to the content type of the web server response
    - Servlet
      - response.setContentType("text/xml; charset=UTF-8");
    - JSP
      - <%@ page contentType="text/xml; charset=UTF-8"%>
    - XML File
      - MIME in web.xml

```
<mime-mapping>
  <extension>xml</extension>
  <mime-type>text/xml</mime-type>
</mime-mapping>
```

# Ajax in 10 Minutes

- ajax01.html - 1

```
<HTML>
<HEAD>
<TITLE> Test AJAX</TITLE>
<meta http-equiv="Content-Type" content="text/html ;charset=UTF-8">
<script language="JavaScript">
var objRequest = createRequestObject();
function createRequestObject() {
        var objTemp = false;
        if(window.XMLHttpRequest) {
                objTemp = new XMLHttpRequest();
        } else if(window.ActiveXObject) {
                objTemp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        return objTemp;
 }
```

# Ajax in 10 Minutes

- ajax01.html - II

```
function getData() {
        if(objRequest) {
                objRequest.open("GET","data.txt");
                objRequest.onreadystatechange = handleResponse;
                objRequest.send(null);
        }
}
function handleResponse() {
        var objDiv = document.getElementById("targetDiv");
        if(objRequest.readyState == 4 && objRequest.status == 200) {
                objDiv.innerHTML = objRequest.responseText;
        }
}
</script>
```

# Ajax in 10 Minutes

- ajax01.html – III

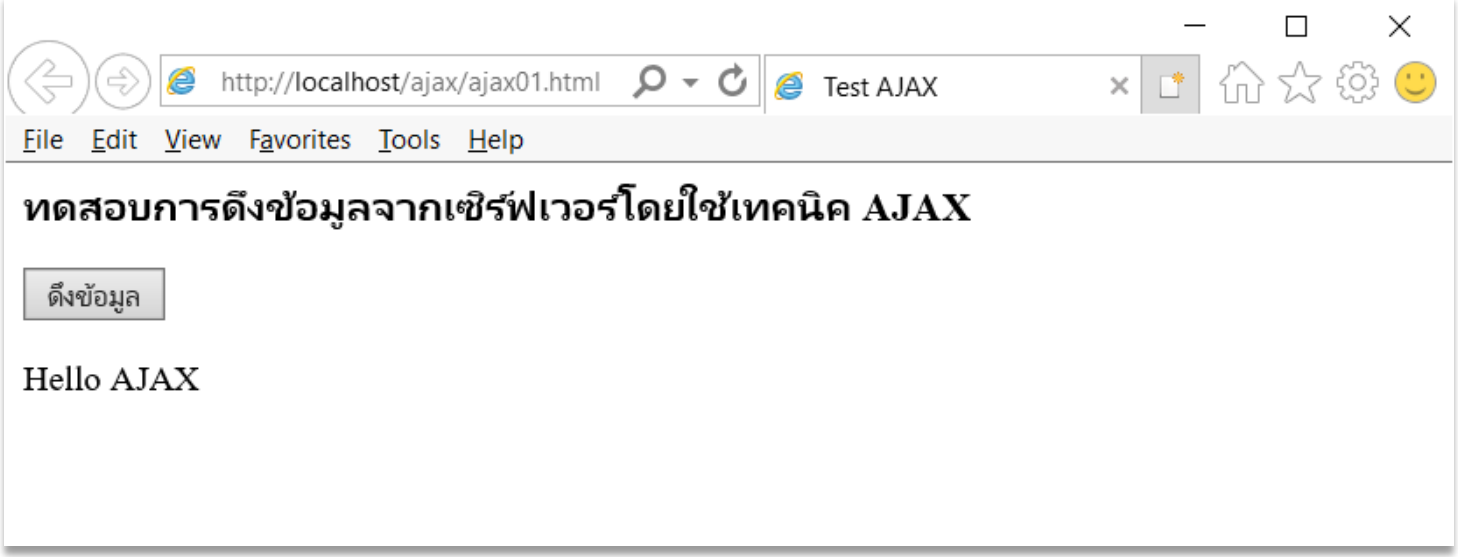
```
</HEAD>
<BODY>
<h3>ทดสอบการดึงข้อมูลจากเซิร์ฟเวอร์โดยใช้เทคนิค AJAX</h3>
<input type="button" value="ดึงข้อมูล" onClick="getData();">
<br><br>
<div id="targetDiv">ข้อมูลจากเซิร์ฟเวอร์จะแสดงที่นี่</div>
</BODY>
</HTML>
```

  - data.txt

```
Hello AJAX
```

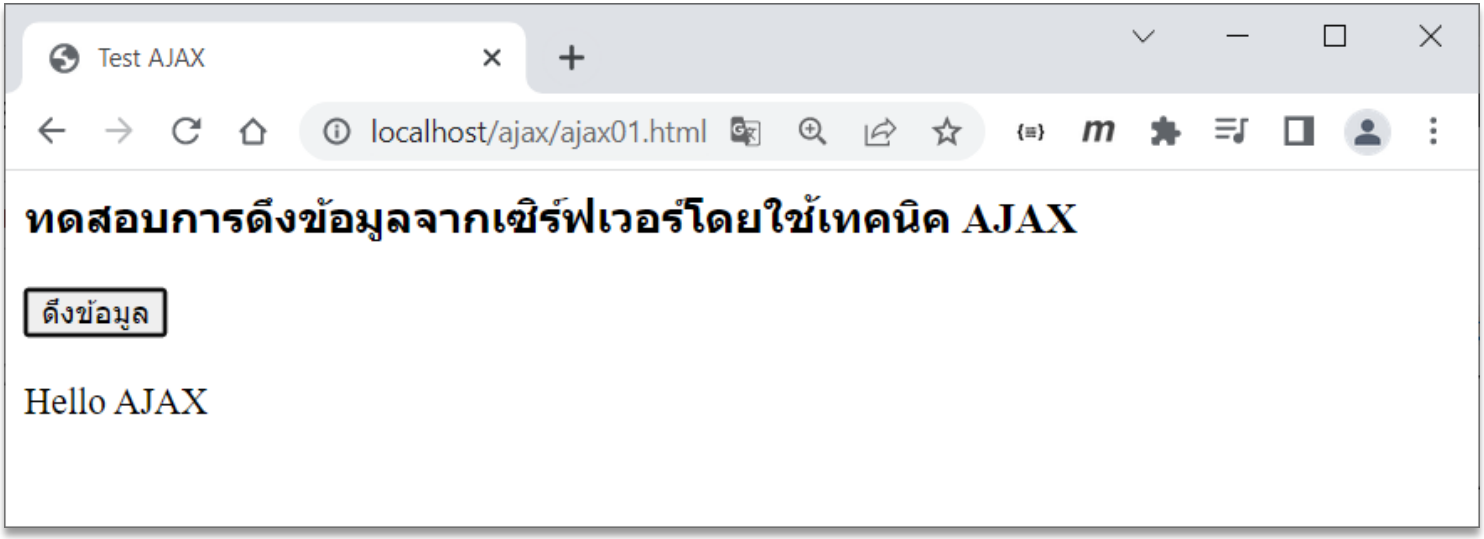
# Ajax in 10 Minutes

- Internet Explorer



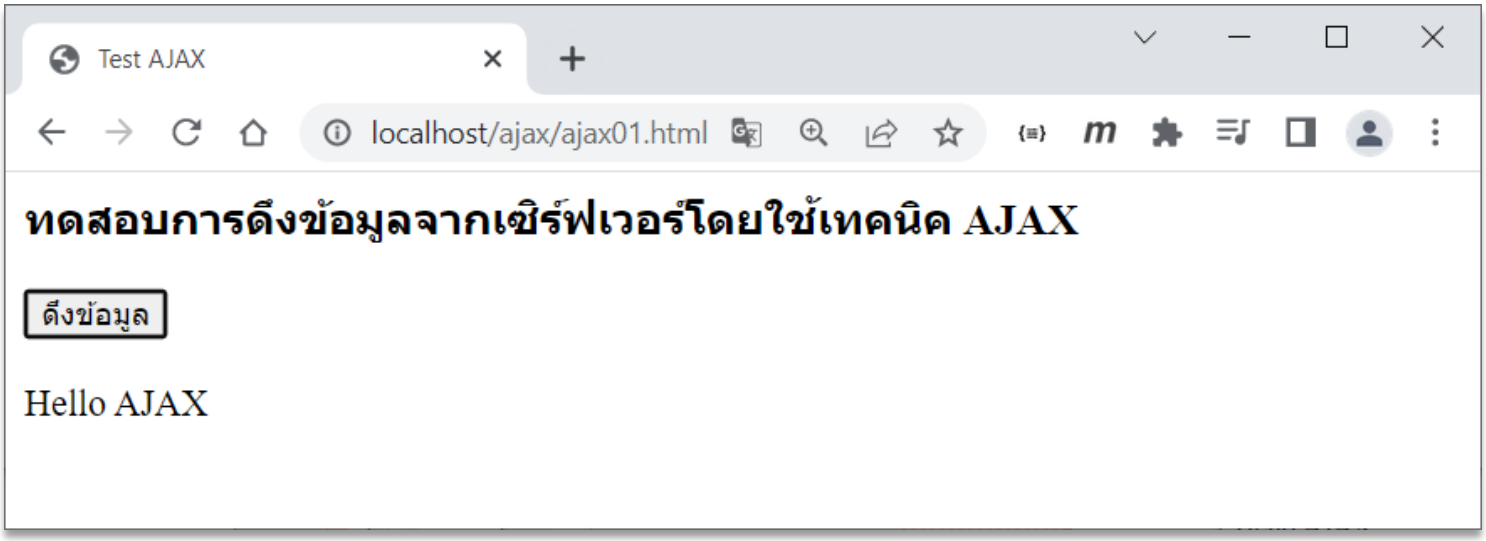
# Ajax in 10 Minutes

- Google Chrome



# Ajax in 10 Minutes

- Mozilla Firefox



# Ajax with Fetch API

- Fetch API provides a JavaScript for accessing and manipulating request and response
- It will seem familiar to anyone who has used XMLHttpRequest
- The Promise returned from fetch

# Ajax with Fetch API

- ajax02.html

```
<HTML>
<HEAD>
<TITLE> Test AJAX</TITLE>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
<script language="JavaScript">
 async function getData() {
            let response = await fetch("data.txt");
            targetDiv.innerHTML = await response.text();
 }
 </script>
 </HEAD>
 <BODY>
 <h3>ทดสอบการดึงข้อมูลจากเซิร์ฟเวอร์โดยใช้เทคนิค AJAX</h3>
 <input type="button" value="ดึงข้อมูล" onClick="getData();">
 <br><br>
 <div id="targetDiv">ข้อมูลจากเซิร์ฟเวอร์จะแสดงที่นี่</div>
 </BODY>
</HTML>
```

# Reference

- http://en.wikipedia.org/wiki/JavaScript
- http://en.wikipedia.org/wiki/DOM_Events
- https://aws.amazon.com/what-is/javascript/
- http://www.w3schools.com/JS/default.asp
- http://www.w3schools.com/jsref/default.asp
- https://cs.brown.edu/courses/bridge/1998/res/javascript/javascript-tutorial.html
- http://www.json.org/
- http://json-lib.sourceforge.net/index.html
- https://www.w3schools.com/js/js_ajax_intro.asp
- http://www.c-point.com/javascript_tutorial/Editor/ajax_tutorial.htm
- https://www.w3schools.com/jsref/api_fetch.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

# Q & A