AJAX

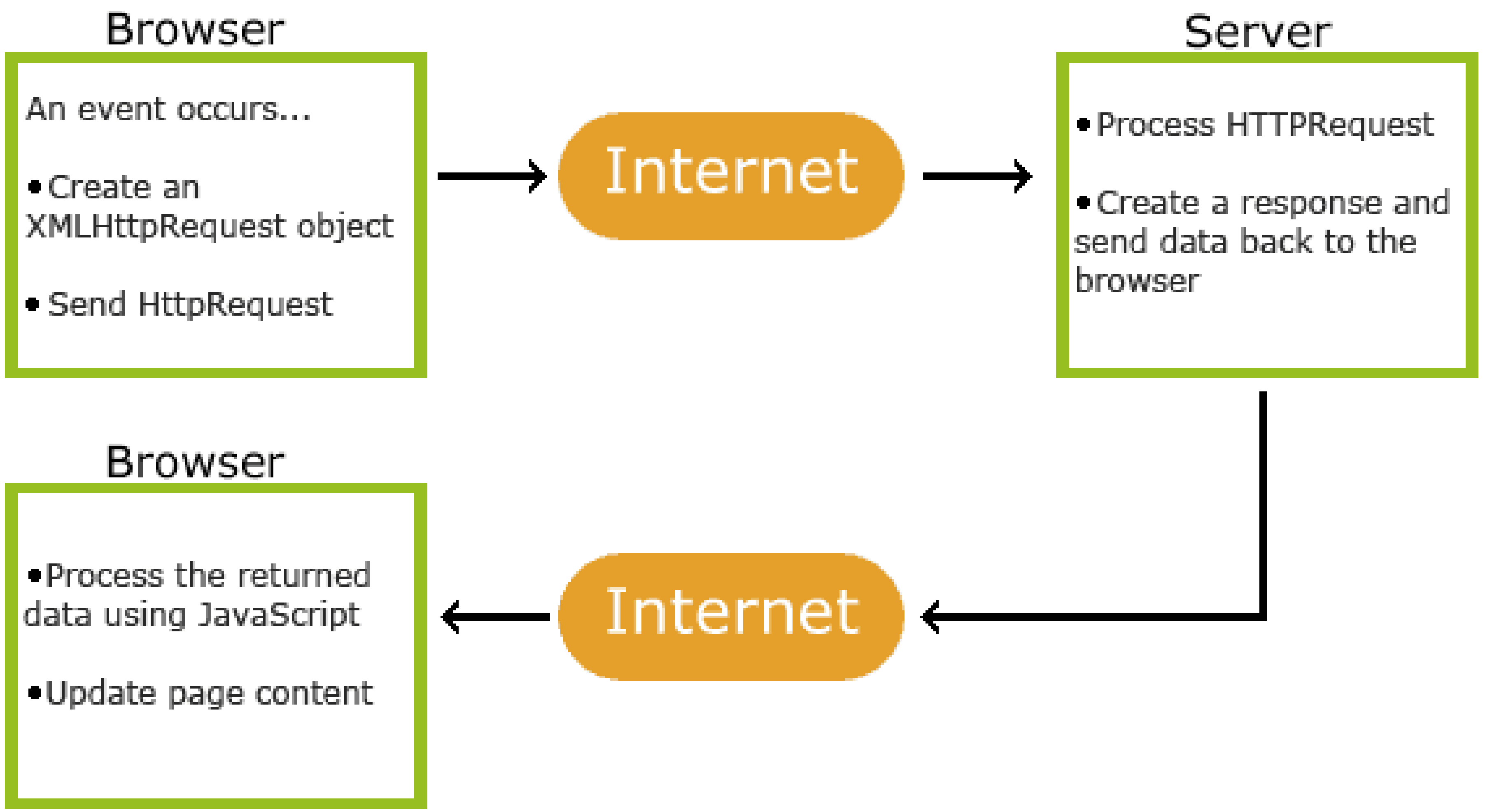
Introduction

- AJAX = Asynchronous JavaScript And XML.
- AJAX is not a programming language.
- AJAX just uses a combination of:
  - A browser built-in XMLHttpRequest object (to request data from a web server)
  - JavaScript and HTML DOM (to display or use the data)

How AJAX Works

- An event occurs in a web page (the page is loaded, a button is clicked)
- An XMLHttpRequest object is created by JavaScript
- The XMLHttpRequest object sends a request to a web server
- The server processes the request
- The server sends a response back to the web page
- The response is read by JavaScript
- Proper action (like page update) is performed by JavaScript

How AJAX Works


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

The XMLHttpRequest Object

- The XMLHttpRequest object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Syntax
  - *variable* = new XMLHttpRequest();
- Eg:
  - var xhttp = new XMLHttpRequest();

ActiveXObject

- Old versions of Internet Explorer (5/6) use an ActiveX object instead of the XMLHttpRequest object
- Eg:
  - *variable* = new ActiveXObject("Microsoft.XMLHTTP");

- Eg:

```
if (window.XMLHttpRequest)
{
      xmlhttp = new XMLHttpRequest();
 }
else
{
xmlhttp = new
ActiveXObject("Microsoft.XMLHTTP");
}
```

XMLHttpRequest Object Methods

Method	**Description**
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method, url, async, user, psw)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true or false *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(string)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	**Description**
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"

Send a Request To a Server

- To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object
  - xhttp.open("GET", "ajax_info.txt", true);
    xhttp.send();

Method	**Description**
open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

GET or POST

- GET is simpler and faster than POST, and can be used in most cases.
- However, always use POST requests when:
  - A cached file is not an option (update a file or database on the server).
  - Sending a large amount of data to the server (POST has no size limitations).
  - Sending user input (which can contain unknown characters), POST is more robust and secure than GET.
- GET Request

```
xhttp.open("GET", "demo_get.php", true);
xhttp.send();
```

- POST Request

```
xhttp.open("POST", "demo_post.php", true);
xhttp.send();
```

POST forms

- To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method

```
xhttp.open("POST", "ajax_test.php", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

- The url - A File On a Server: The url parameter of the open() method, is an address to a file on a server
  - EG: xhttp.open("GET", "ajax_test.php", true);
-

AJAX - Server Response

- The onreadystatechange Property
  - The readyState property holds the status of the XMLHttpRequest.
  - The onreadystatechange property defines a function to be executed when the readyState changes.
  - The status property and the statusText property holds the status of the XMLHttpRequest object.

Using a Callback Function

- A callback function is a function passed as a parameter to another function.

- If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task.

- The function call should contain the URL and what function to call when the response is ready.

```
loadDoc("url-1", myFunction1);
loadDoc("url-2", myFunction2);
function loadDoc(url, cFunction) {
  var xhttp;
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      cFunction(this);
    }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}
function myFunction1(xhttp) {
}
function myFunction2(xhttp) {
}
```

Server Response Properties and Methods

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

JSON - JavaScript Object Notation

Introduction

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a text format for storing and transporting data
- JSON is "self-describing" and easy to understand
- JSON is a lightweight text-based open standard designed for human-readable data interchange.
- The JSON syntax is derived from JavaScript object notation, but the JSON format is text only. Code for reading and generating JSON exists in many programming languages.

JSON syntax

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

```
{
  "book": [

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

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

  ]
}
```

Cont..

- JSON supports the two data structures −
- Collection of name/value pairs − This Data Structure is supported by different programming languages.
- Ordered list of values − It includes array, list, vector or sequence etc.

JSON - DataTypes

Type & Description
Number double- precision floating-point format in JavaScript
String double-quoted Unicode with backslash escaping
Boolean true or false
Array an ordered sequence of values
Value it can be a string, a number, true or false, null etc
Object an unordered collection of key:value pairs
Whitespace can be used between any pair of tokens
null empty

JSON.parse()

- A common use of JSON is to exchange data to/from a web server.
- When receiving data from a web server, the data is always a string.
- Parse the data with JSON.parse(), and the data becomes a JavaScript object.
- Imagine we received this text from a web server:
  - '{"name":"John", "age":30, "city":"New York"}‘
- Use JSON.parse() to convert text into a JavaScript object:
  - const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');

JSON.stringify()

- A common use of JSON is to exchange data to/from a web server.
- When sending data to a web server, the data has to be a string.
- Convert a JavaScript object into a string with JSON.stringify().
- Example

```
const obj = {name: "John", age: 30, city: "New York"};
const myJSON = JSON.stringify(obj);
```

Example

```
<h2>Store and retrieve data from local storage.</h2>
<p id="demo"></p>
<script>
// Storing data:
const myObj = { name: "John", age: 31, city: "New York" };
const myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
let text = localStorage.getItem("testJSON");
let obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
</script>
```

JSON - Objects

- Creating Simple Objects
- JSON objects can be created with JavaScript.
- Creation of an empty Object −
  - *var JSONObj = {};*
- Creation of a new Object −
  - *var JSONObj = new Object();*
- Creation of an object with attribute bookname with value in string, attribute price with numeric value. Attribute is accessed by using '.' Operator −
  - *var JSONObj = { "bookname ":"VB BLACK BOOK", "price":500 };*

Creating Simple Objects

```
<html>
  <head>
    <title>Creating Object JSON with JavaScript</title>
    <script language = "javascript" >
      var JSONObj = { "name" : "manipal.edu", "dept"  : "Computer Applications" };
      document.write("<h1>JSON with JavaScript example</h1>");
      document.write("<br>");
      document.write("<h3>Website Name = "+JSONObj.name+"</h3>");
      document.write("<h3>Year = "+JSONObj.dept+"</h3>");
    </script>
  </head>
  <body>
  </body>
</html>
```

JSON PHP

- PHP has some built-in functions to handle JSON.
- Objects in PHP can be converted into JSON by using the PHP function json_encode()

```
<?php
    $myObj->name = "John";
    $myObj->age = 30;
    $myObj->city = "New York";
    $myJSON = json_encode($myObj);
    echo $myJSON;
?>
```

The Client JavaScript

```
<h2>Get JSON Data from a PHP Server</h2>
<p id="demo"></p>
<script>
const xmlhttp = new XMLHttpRequest();

xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML = myObj.name;
}
xmlhttp.open("GET", "demo_file.php");
xmlhttp.send();
</script>
```

PHP Array

```
<h2>Get JSON Data from a PHP Server</h2>
<p>Convert the data into a JavaScript array:</p>
<p id="demo"></p>
<script>
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML
= myObj[2];
}
xmlhttp.open("GET", "demo_file_array.php");
xmlhttp.send();
</script>
```

```
<?php
$myArr
= array("John", "Mary", "Peter", "Sally"
);

$myJSON = json_encode($myArr);

echo $myJSON;
?>
```