

Web Programming Technologies

Harshita Maheshwari

Session-13

Topics to be covered...

- What is JSON?
- Why use JSON
- JSON Syntax
- JSON Value
- JSON Object
- JSON Array
- JSON vs. XML
- How and when to use JSON
- Reading Data from JSON
- Creating JSON Text From JavaScript

JavaScript Object Notation (JSON):

- Is an open standard light-weight format that is used to store and exchange data.
- Is an easier and faster alternative to XML.
- Is language independent format that uses human readable text to transmit data objects.
- Consists of objects of name/value pairs.
- Files have the extension .json.

Syntactically, JSON is similar to the code for creating JavaScript objects. Due to this similarity, standard JavaScript methods can be used to convert JSON data into JavaScript objects.

JSON is NOT...

- Overly Complex
- A “document” format
- A markup language
- A programming language

Why Use JSON

- Straightforward syntax
- Easy to create and manipulate
- Supported by all major JavaScript frameworks
- Supported by most backend technologies

- JSON uses name/value pairs to store data.
- Commas are used to separate multiple data values.
- Objects are enclosed within curly braces.
- Square brackets are used to store arrays.
- JSON keys must be enclosed within double quotes.

Syntax:

```
{"fName":"Ronald", "lName":"Smith", "Contact":"121 12345"}
```

String:

Sequence of 0 or more Unicode characters

Wrapped in "double quotes"

Backslash escapement

Number:

Integer, Real, Scientific, No octal or hex, No NaN or Infinity – Use null instead.

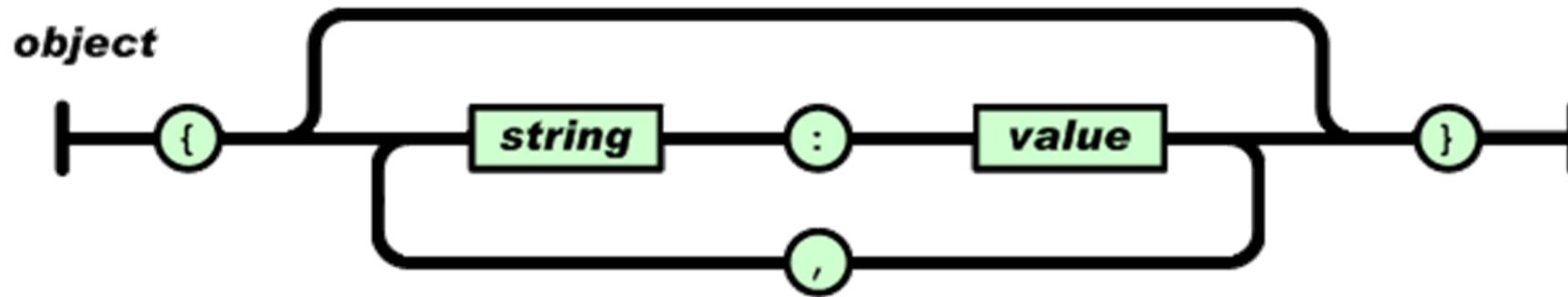
Booleans: true or false

Null: A value that specifies nothing or no value.

"Name": "Jane"	\\Storing string value
"isAlive": true	\\Storing boolean value
"age": 23	\\Storing integer value
"spouse": null	\\Storing null

JSON Object

- Unordered sets of name/value pairs
- Begins with { (left brace)
- Ends with } (right brace)
- Each name is followed by : (colon)
- Name/value pairs are separated by , (comma)



```

{
  "fName": "Jane",
  "lName": "Doe",
  "isAlive": true,
  "age": 23,
  "children": [],
  "spouse": null
}

```

JSON Array

- An ordered collection of values
- Begins with [(left bracket)
- Ends with] (right bracket)
- Name/value pairs are separated by , (comma)

```
{
  "fName": "Jane",
  "lName": "Doe",
  "isAlive": true,
  "age": 23,
  "ContactNumber": [
    {"type": "Mobile", "Number": "+9198765" },
    {"type": "Office", "Number": "+9124456" }
  ],
  "children": [],
  "spouse": null
}
```

Similarities

- Both are human-readable, that is, self-describing.
- Both represent hierarchical structure, that is, values within values.
- Both can be accessed and parsed by almost every programming language.
- Both can be accessed and fetched with an XMLHttpRequest object.

Dissimilarities

- XML needs an XML parser, whereas, a standard JavaScript method can be used to parse JSON.
- There is no need of end tag in JSON.
- JSON is much shorter as compared to XML.
- It is easy to read and write JSON.
- JSON can be used with arrays.

XML

```
<students>
  <student>
    <fName>Jenny</fName>
    <lName>Watson</lName>
  </student>
  <student>
    <fName>Dean</fName>
    <lName>Smith</lName>
  </student>
</students>
```

JSON

```
{"students": [
  {"fName": "Jenny", "lName": "Watson"},
  {"fName": "Dean", "lName": "Smith"}
]}
```

How & When to use JSON

- Transfer data to and from a server
- Perform asynchronous data calls without requiring a page refresh
- Working with data stores
- Compile and save form or user data for local storage

Reading Data From JSON

- A most common usage of JSON objects is to read/fetch data from a Web server in JSON format, and display it on an HTML Web page.
- To read data from a JSON object, you can use the **JSON.parse()** method provided by JavaScript.
- Syntax: `var obj = JSON.parse(text);`

Example:

```
<script>
var x = '[      { "code": "1001", "name": "ram" },
                { "code": "1002", "name": "shyam" },
                { "code": "1003", "name": "seeta" }
            ]';
var r = JSON.parse(x);
for (var i in r)
document.write(r[i].code+" " +r[i].name+"<br/>");
</script>
```

Creating JSON Text From JavaScript

JavaScript provides you the `JSON.stringify()` method that allows you to convert JavaScript value to a JSON string.

Syntax: `var obj = JSON.stringify(value);`

```
<script>
    var x = [
        { code: "1001", name: "ram" },
        { code: "1002", name: "shyam" },
        { code: "1003", name: "seeta" }
    ];
    var r = JSON.stringify(x);
    document.write(r);
</script>
```

Session-14



- ❖ Synchronous web communication
- ❖ Asynchronous web communication
- ❖ Web Application and Ajax
- ❖ Ajax Features
- ❖ Where should use Ajax
- ❖ Core Component
- ❖ Process Cycle
- ❖ XMLHttpRequest Object
- ❖ Prototype Ajax Model
- ❖ Handling Ajax request using jQuery

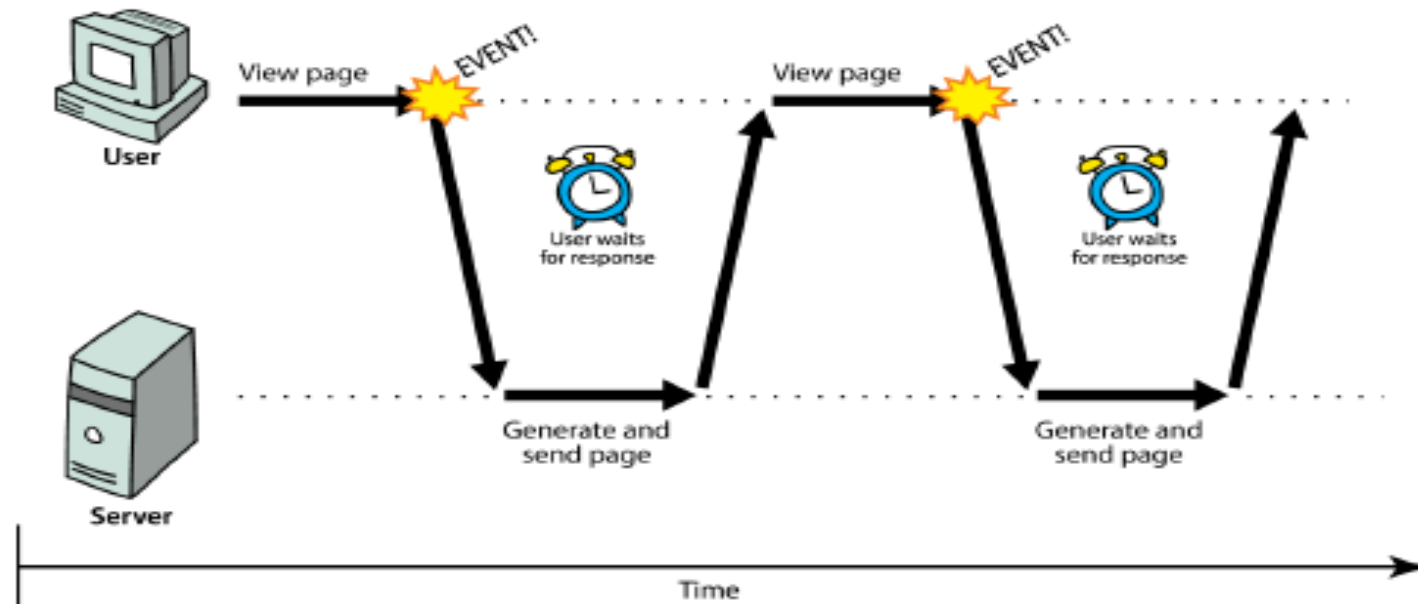
Standard Client Server Application



These are standard client server applications where client has to wait for the response from the server to execute other task on the webpage.

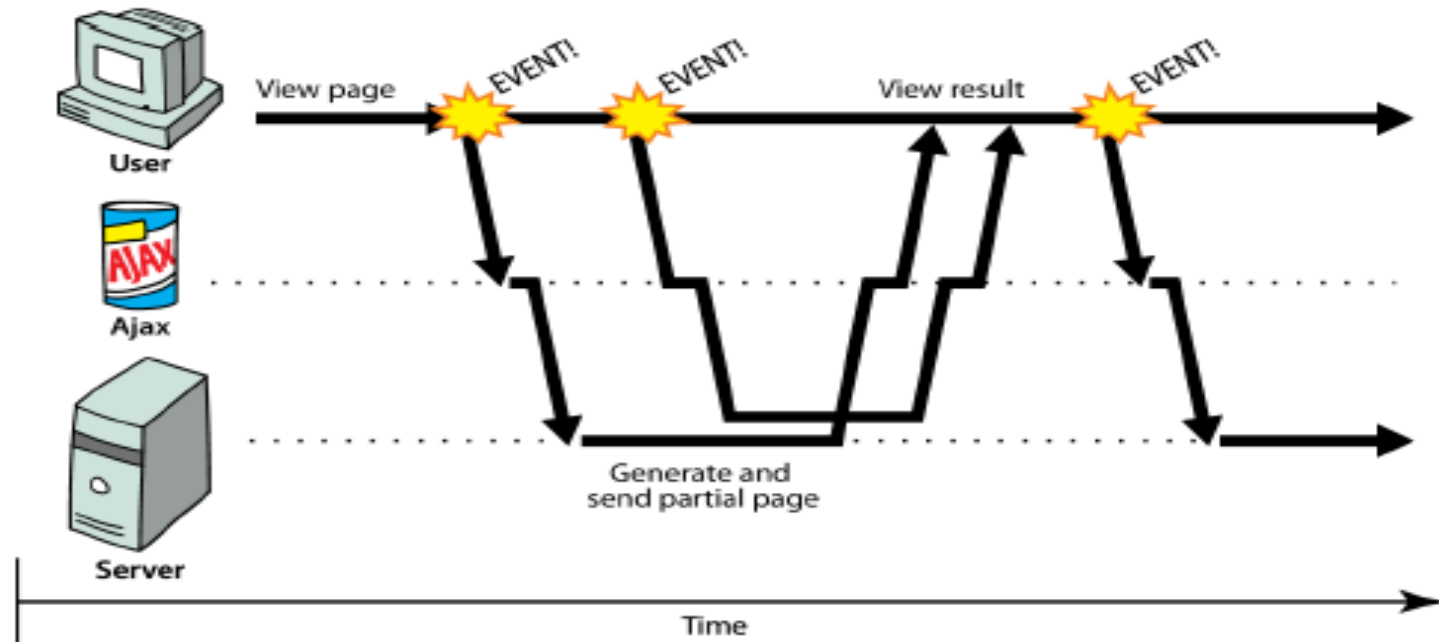
synchronous:

- user must wait while new pages load
- The typical communication pattern used in web pages (click, wait, refresh)



Exporting with `JSON.stringify()` while data loss

Interacting with page while data loads



Ajax stands for Asynchronous JavaScript and XML

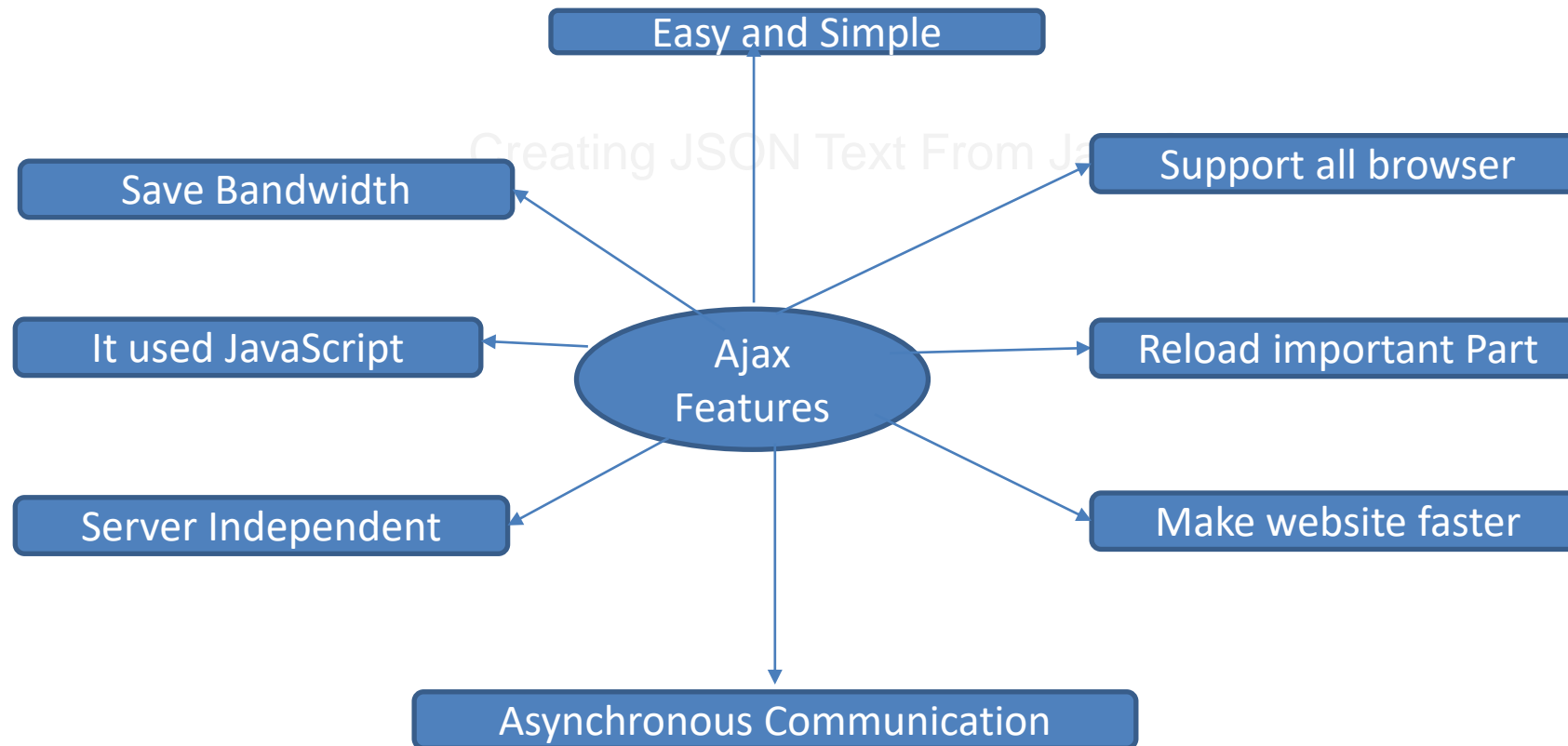
Asynchronous:- Asynchronous means that we are exchanging data to/from the server in the background without having to refresh the page.

AJAX was made popular in 2005 by Google, with Google Suggest.

What about the response in Ajax Application

Responses from the server in AJAX are handled in the form of callbacks.

A callback is a special function which is used in AJAX so that server can respond to the client when it is ready to send data to the client.



Where should use Ajax?

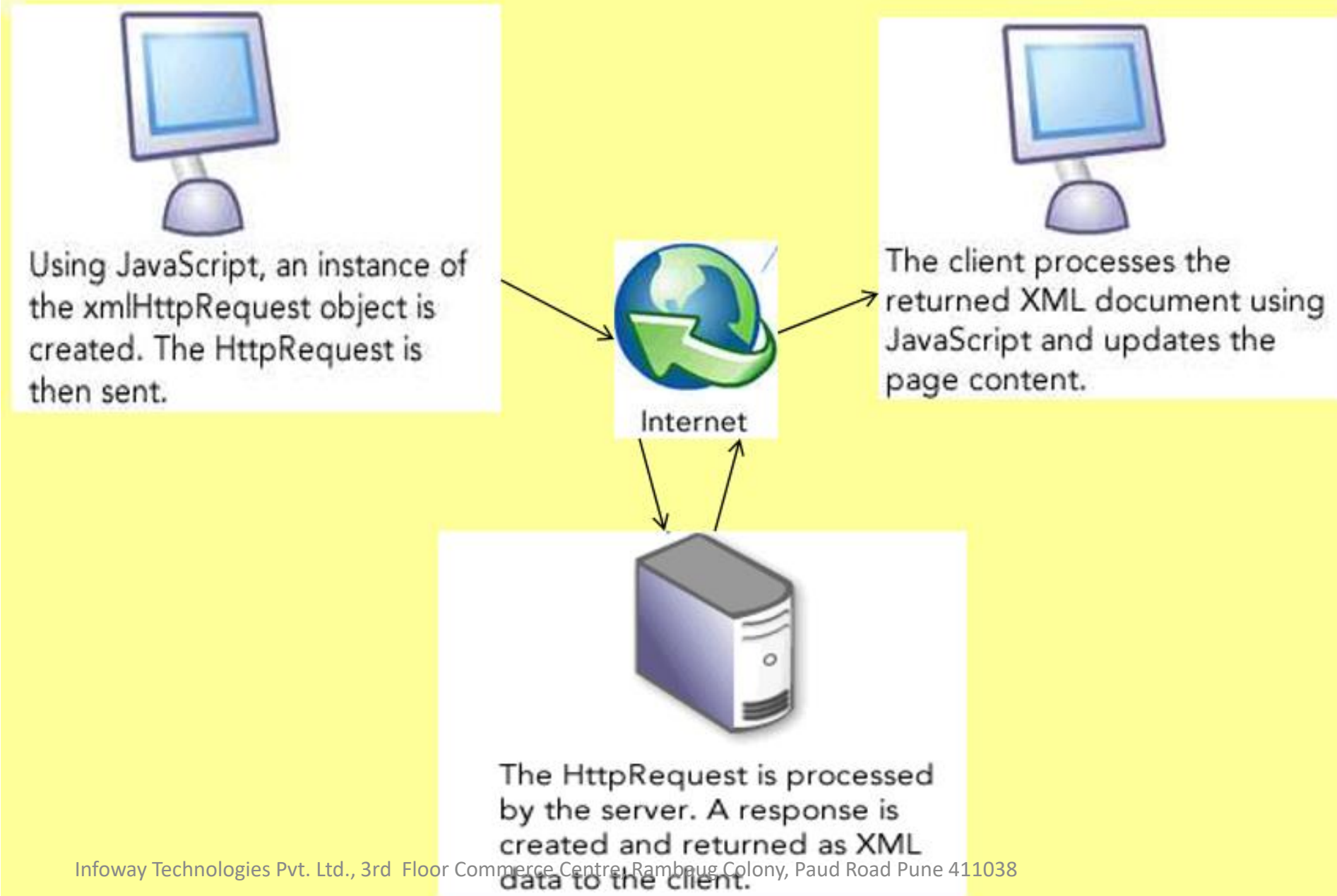
- Form Validation
- Sort or Filter
- Vote or Rating
- Chat Websites
- Blog Comments
- Captcha

Creating JSON Text From JavaScript

The Core Components

- HTML & CSS - for presenting.
- JavaScript - for local processing.
- Document Object Model (DOM) – to access data inside the page or to access elements of an XML file on the server.
- XMLHttpRequest object – to read/send data to the server asynchronously.

The Process Cycle



Step-1

Create a XMLHttpRequest Object

Creating JSON Text From JavaScript

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a server behind the scenes.

This means that it is possible to update parts of a web page without reloading the whole webpage.

Syntax:

Variable=new XMLHttpRequest();

var xhttp= new XMLHttpRequest();

Step-2

Making a request to the server

The XMLHttpRequest object is used to exchange data with 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);

Method used in request
it can be get or post

This is the url of the
file that we want to load
into our web page

true signifies that we are making
asynchronous request if we set it to false
then it will not to be ajax request

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 **statusText** property holds the status of the XMLHttpRequest object.

readyState: Holds the status of XMLHttpRequest

0: request not initialized

1: server connection established

2: request received

3: processing request

4: request finished and response ready

Status: 200 "OK"

403 "forbidden"

404 "Page not found"

```
Xhttp.onreadystatechange=function(){  
If(this.readyState==4 && this.status==200){  
Document.getElementById("demo").innerHTML=this.responseText;  
}  
};
```





We are executing this anonymous function when this readystatechange event trigger or readystate value changes if the value of readystate is 4 and status is 200, we are just modifying the inner html of id demo with the response that is returned from the server.

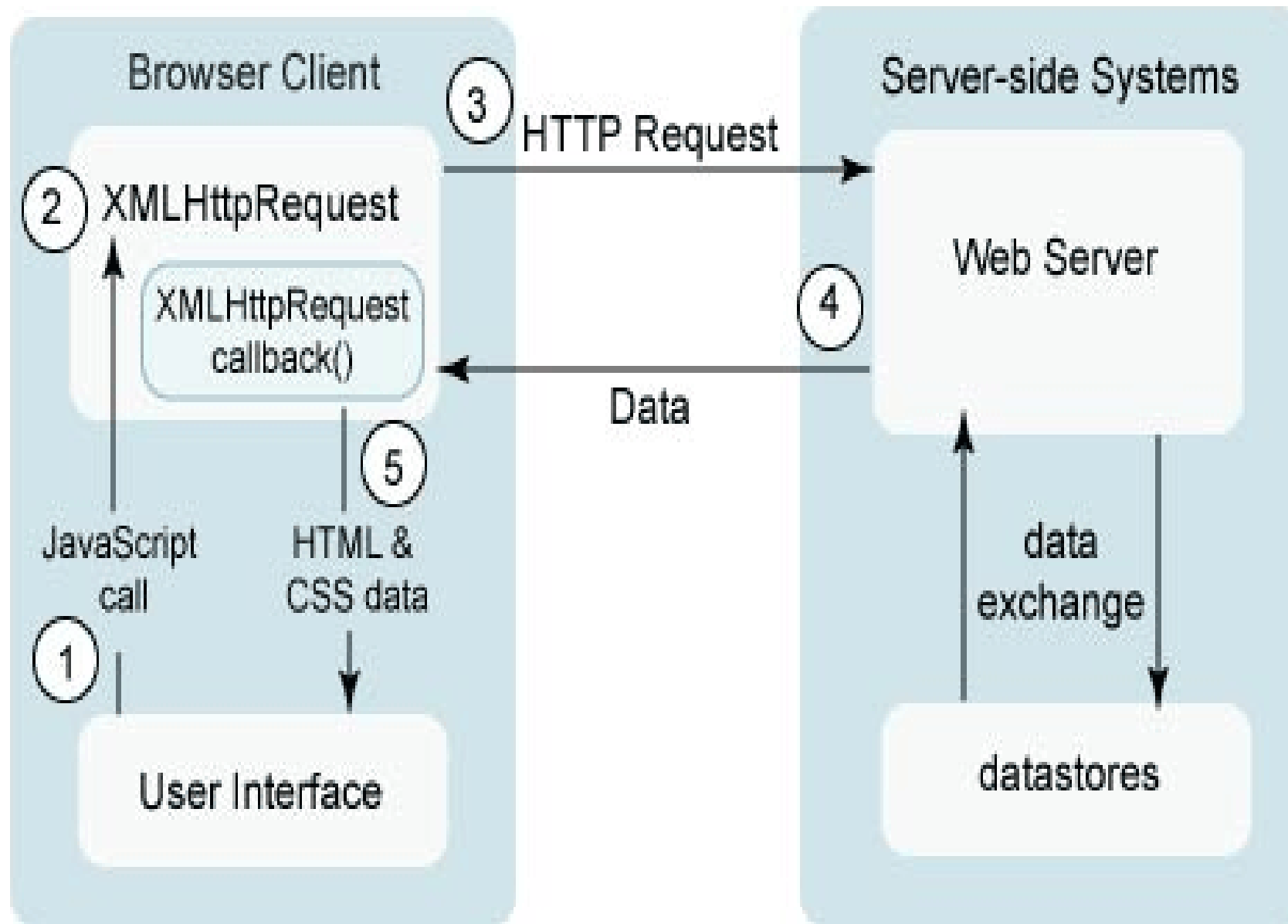
Send the request

```
Xhttp.send();
```

Creating JSON Text From JavaScript

Complete AJAX example

```
var xhttp=new XMLHttpRequest();  1
xhttp.onreadystatechange=function(){
  If(this.readyState==4 && this.status==200){  2
    Document.getElementById("demo").innerHTML=this.responseText;
  }
};
xhttp.open("GET","ajax_info.txt",true);  3
xhttp.send();  4
```



Can I use AJAX with jQuery so that I have to write less code and update parts of a Web page?

Handling AJAX Requests using jQuery

jQuery library provides you with various methods, known as jQuery AJAX methods, that allow you to make a call to the AJAX code. These methods allow you to perform various tasks.

The following list depicts the jQuery AJAX methods:

load()

get()

post()

ajax()

The load() method is used to load or fetch data from a Web server into a selected HTML element.

Syntax :

```
$(selector).load(URL[,data][,complete])
```

URL: Is used to specify the URL from where you want to load the data. The URL can be used to refer to any resource, such as text file or html file.

data: Is used to pass an object of a key/value pair along with the request to the server.

complete: Is used to refer to a callback method that will be executed after the successful execution of load() method.

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("demo.txt");
    });
});
</script>
```

```
<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>
```

```
<button>Get External Content</button>
```

jQuery \$.get() Method

The `get()` method is used to load data from a Web server using the HTTP GET request.

Syntax: `$(selector).get(url[,data][, callback],[datatype])`

`datatype`: Is used to specify the type of data, such as text, JSON, or XML, that is expected in return from the server.

```
$("#button").click(function(){  
    $.get("demo.html", function(data, status){  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});
```


jQuery \$.post() Method

The post()method is:

- Similar to the get()method.
- Used to load data from a Web server using the HTTP POST request.
- Used when the requested is large in amount.
- Used to send the data in an encrypted format.

Syntax: \$(selector).post(url[,data][, callback],[datatype])

```

$("button").click(function(){
    $.post("demo_post.php",
    {
        name: "Infoway Technologies",
        city: "Pune"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});

```

jQuery ajax() Method

The ajax()jQuery method:
Can be used to call the AJAX requests.
Helps in partial-page updates.

Syntax : \$(selector).ajax(options)

options: Is an optional parameter that helps in configuring the AJAX calls by using key/value pairs.

Option	Description
<i>async</i>	<i>A boolean value that indicates whether to execute the request asynchronously.</i>
<i>complete</i>	<i>A callback function that executes whenever the request finishes.</i>
<i>datatype</i>	<i>A string defining the type of data, such as XML, HTML, JSON, or script, that is expected back from the server.</i>
<i>success</i>	<i>A callback function that is executed if the request succeeds.</i>
<i>type</i>	<i>A string defining the HTTP method to be used for the request (GET or POST).</i>
<i>URL</i>	<i>A required option that refers to the string containing the URL to which the request is sent.</i>

```
$("#button").click(function(){
    $.ajax({
        url: "demo.txt",
        type: 'GET',
        success: function(result){
            $("#div1").html(result);
        }
    });
});
```

jQuery getJSON() Method

The getJSON() method is used to get JSON data using an AJAX HTTP GET request.

Syntax: `$(selector).getJSON(url,data,success(data,status,xhr))`

`success(data,status,xhr):`

Optional. Specifies the function to run if the request succeeds

`data` - contains the data returned from the server.

`status` - contains a string containing request status ("success", "notmodified", "error", "timeout", or "parsererror").

`xhr` - contains the XMLHttpRequest object

```

$("button").click(function() {
    $.getJSON("demo.js", function(result) {
        $.each(result, function(i, field) {
            $("div").append(field + " ");
        });
    });
});

```

END

Thank You