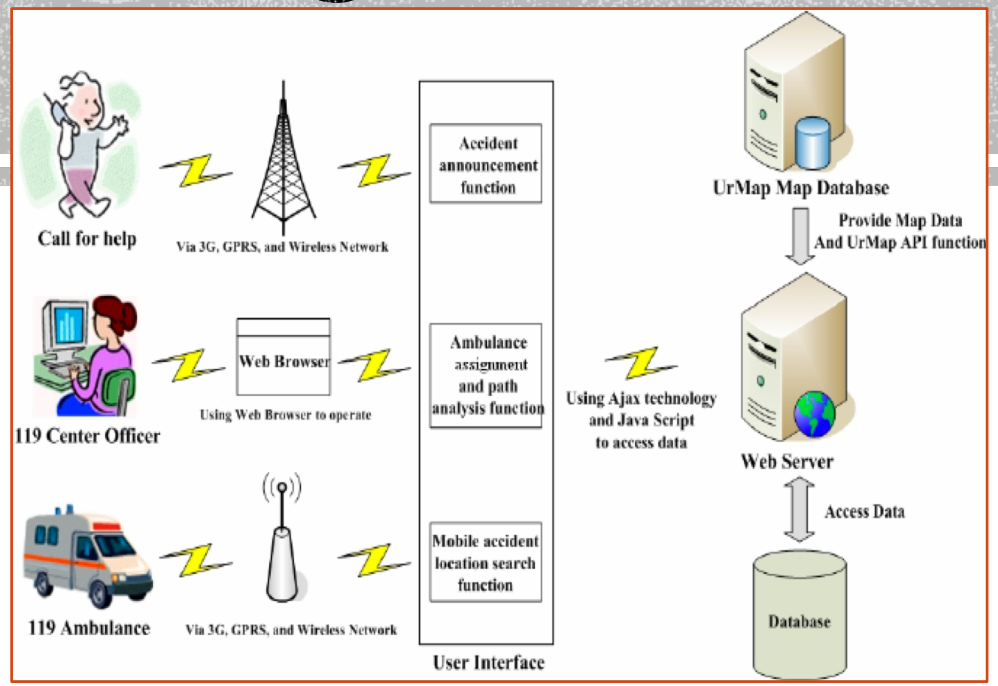
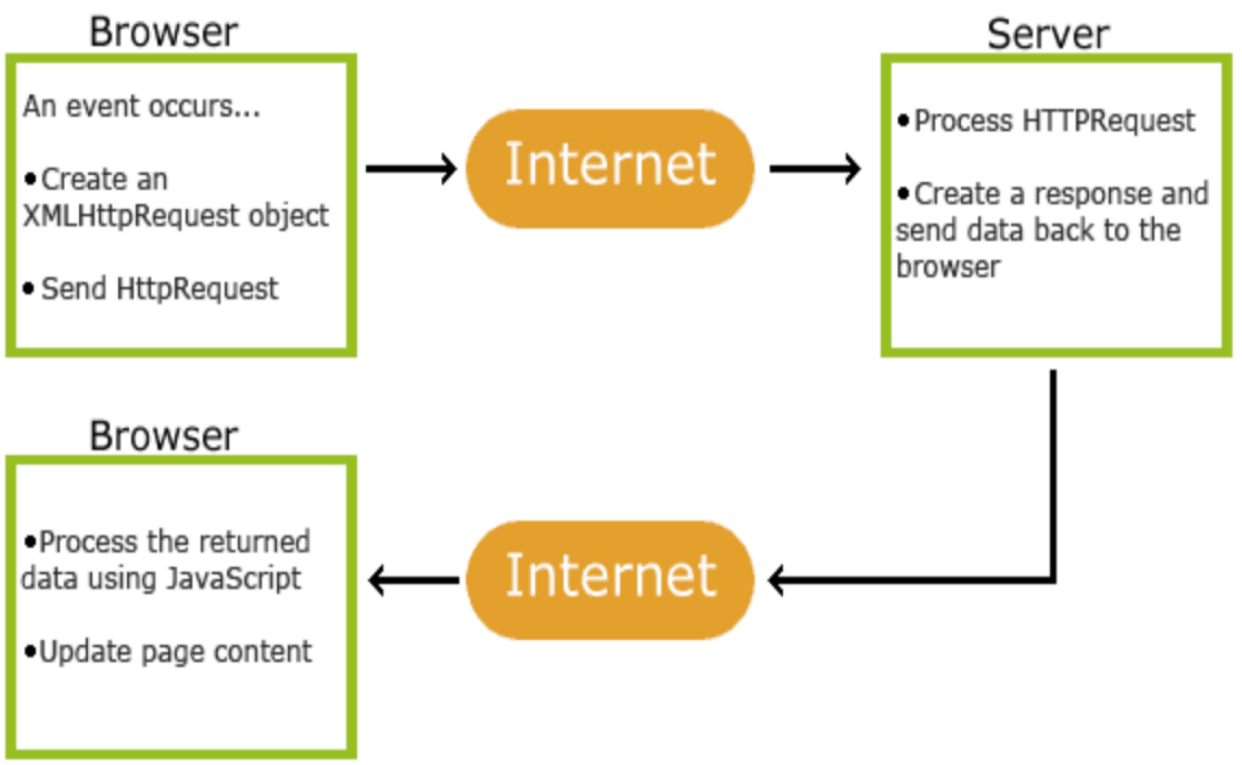
AJAX


Call for help
Via 3G, GPRS, and Wireless Network
Accident announcement function
UrMap Map Database
Provide Map Data And UrMap API function
119 Center Officer
Web Browser
Using Web Browser to operate
Ambulance assignment and path analysis function
Using Ajax technology and Java Script to access data
Web Server
Access Data
119 Ambulance
Via 3G, GPRS, and Wireless Network
Mobile accident location search function
Database
User Interface

1

INTRODUCTION

- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
- Not a programming language.
- Uses a combination of:
  - A browser built-in XMLHttpRequest object (to request data from a web server)
  - JS and HTML DOM (to display or use the data)
- AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.
- Allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes.
- Possible to update parts of a web page, without reloading the whole page.
- Request/ Receive data from a server - after the page has loaded
- Send data to a server - in the background



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

THE XML HTTP REQUEST OBJECT

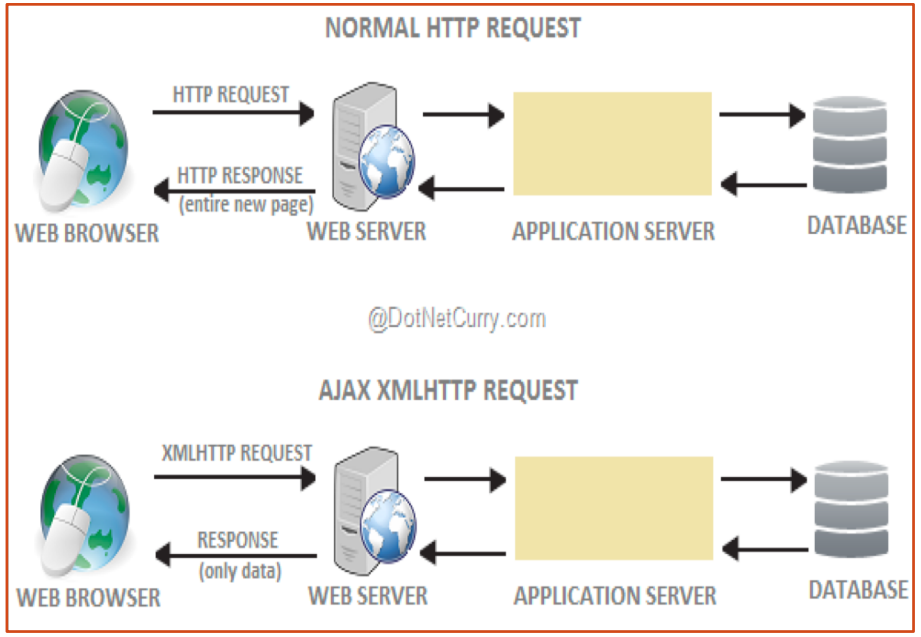
- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.
- All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.
- For security reasons, modern browsers do not allow access across domains.
- This means that both the web page and the XML file it tries to load, must be located on the same server.
- We can create an object using:

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

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 (asynchronous) or false (synchronous) *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

XML HTTP REQUEST 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" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")



INVOKING AJAX

GET

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();
```

POST

```
xhttp.open("POST", "demo_post2.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

CALLBACK

ASYNC

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

SYNC

```
xhttp.open("GET", "ajax_info.txt", false);
```

AJAX & JQUERY (1)

- The jQuery load() method is a simple, but powerful AJAX method.
- The load() method loads data from a server and puts the returned data into the selected element.
- The optional callback parameter specifies a callback function to run when the load() method is completed.
- The callback function can have different parameters:
  - responseTxt - contains the resulting content if the call succeeds
  - statusTxt - contains the status of the call
  - xhr - contains the XMLHttpRequest object

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
            if(statusTxt == "success")
                alert("External content loaded successfully!");
            if(statusTxt == "error")
                alert("Error: " + xhr.status + ": " + xhr.statusText);
        });
    });
});
</script>
</head>
<body>

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

<button>Get External Content</button>

</body>
</html>
```

AJAX & JQUERY (2)

- Two commonly used methods for a request-response between a client and server are:
  - **GET** - Requests data from a specified resource
  - **POST** - Submits data to be processed to a specified resource
- GET is basically used for just getting (retrieving) some data from the server. GET method may return cached data.
- POST can also be used to get some data from the server. POST method NEVER caches data and often used to send data along with the request.

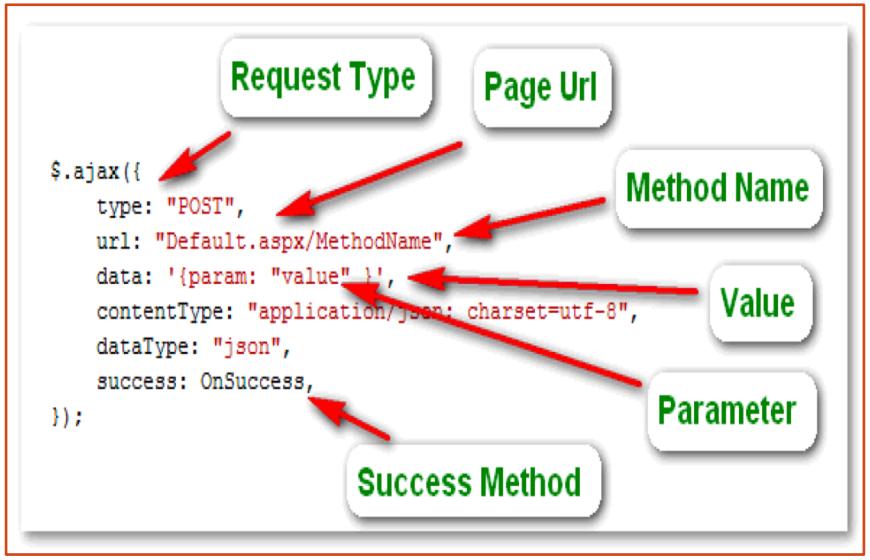
```
$.get("demo_test.asp", function(data, status){
    alert("Data: " + data + "\nStatus: " + status);
});
```

```
$.post("demo_test_post.asp",
{
  name: "Donald Duck",
  city: "Duckburg"
},
function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
});
```

AJAX & JQUERY (3)

```
$.ajax({url: "demo_test.txt", success: function(result){
    $("#div1").html(result);
}});
```

async	A Boolean value indicating whether the request should be handled asynchronous or not. Default is true
beforeSend(*xhr*)	A function to run before the request is sent
complete(*xhr,status*)	A function to run when the request is finished (after success and error functions)
data	Specifies data to be sent to the server
error(*xhr,status, error*)	A function to run if the request fails.
password	Password to be used in an HTTP access authentication request.
success(*result,status,xhr*)	A function to be run when the request succeeds
timeout	The local timeout (in milliseconds) for the request
type	Specifies the type of request. (GET or POST)
url	Specifies the URL to send the request to. Default is the current page
username	Specifies a username to be used in an HTTP access authentication request



AJAX & ANGULARJS

- The AngularJS $http service makes a request to the server, and returns a response.
- The .get method is a shortcut method of the $http service. There are several shortcut methods:
  - .delete()
  - .get()
  - .head()
  - .jsonp()
  - .patch()
  - .post()
  - .put()
- The response from the server is an object with these properties:
  - .config the object used to generate the request.
  - .data a string, or an object, carrying the response from the server.
  - .headers a function to use to get header information.
  - .status a number defining the HTTP status.
  - .statusText a string defining the HTTP status.

```
$http.get("welcome.htm")
.then(function(response) {
    $scope.myWelcome = response.data;
});
```

```
$http({
    method : "GET",
    url : "welcome.htm"
}).then(function mySuccess(response) {
    $scope.myWelcome = response.data;
}, function myError(response) {
    $scope.myWelcome = response.statusText;
});
```