

# Internet Fundamentals & Introduction to Web Technologies

Course: IT (044615)

Lecture: 10

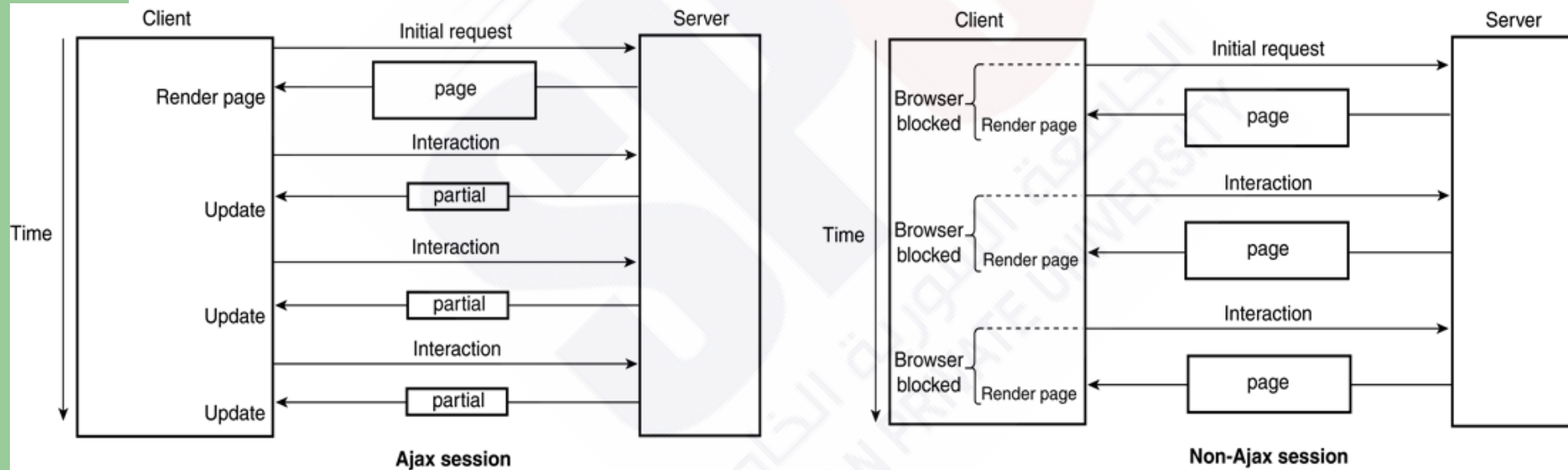
**Introduction to Ajax**

Dr. Ramez Hajislam

# Overview of Ajax

- Ajax is not an API or a programming language
- Ajax aims to provide more responsive web applications
- In normal request/response HTTP cycles, the browser locks waiting for the response and an entire page must be displayed
- With Ajax, asynchronous requests may be made and responses are used to update part of a page
  - User can continue to interact with a page while the request is in progress
  - Less data needs to be transmitted
  - Page update is quicker because only a part of a page is modified

# Traditional and Ajax browser/server interactions



# The Application

- The example application uses the customer information part of the popcorn application
- As the user enters the zip code information, a request for the corresponding city and state will be made to the server
- If successful, the information will be filled in the text widgets

# A display of the popcornA.html document

**Welcome to Millenium Gymnastics Booster Club Popcorn Sales**

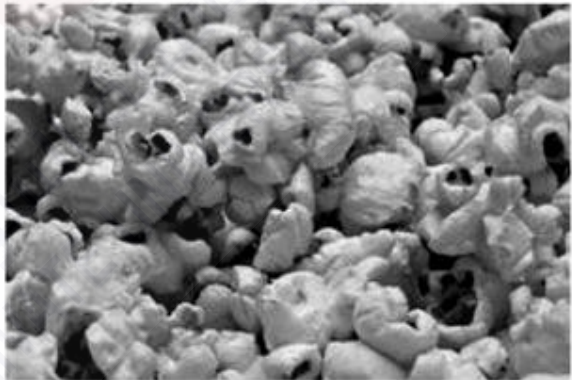
Buyer's Name:

Street Address:

Zip code:

City:

State:



# Display of the form after the zip code has been entered

**Welcome to Millenium Gymnastics Booster Club Popcorn Sales**

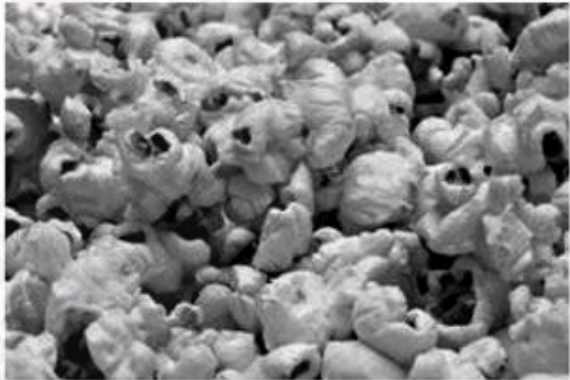
Buyer's Name:

Street Address:

Zip code:

City:

State:



# Display showing the city and state provided implicitly

**Welcome to Millenium Gymnastics Booster Club Popcorn Sales**

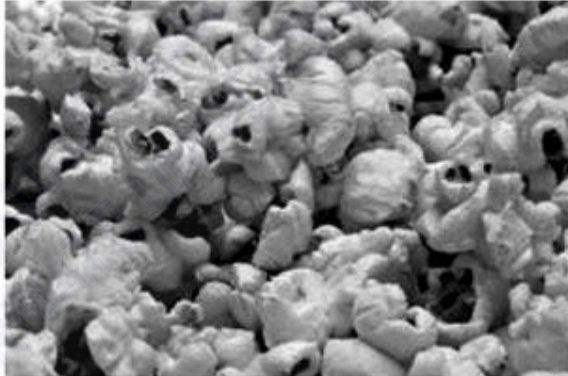
Buyer's Name:

Street Address:

Zip code:

City:

State:



# The Form Document

- The trigger for the request is a blur event on the zip code widget
- `this.value` is used by the handler to get the zip code value entered
- All relevant widgets have the `id` attribute set for easy access in the JavaScript



# The Form Document

```
...  
<tr>  
  <td> zip Code </td>  
  <td> <input type="text" name="zip" onblur=  
    "getPlace(this.value"/></td>  
</tr>  
<tr>  
  <td> city</td>  
  <td> <input type="text" name="city" id = "city"/></td>  
</tr>  
<tr>  
  <td> State</td>  
  <td> <input type="text" name="state" id = "state"/></td>  
</tr>  
...
```

# The Request Phase

- Two functions
  - blur event handler
  - Response processor
- An `XMLHttpRequest` object is used to create the request
- A callback is a function called when a response is received
  - Function `receivePlace` is the callback
  - The function name is assigned to a property of `XMLHttpRequest`
- The `open` method sets up the request
  - Method parameter, either “GET” or “POST”
  - URL parameter with zip code in the URL
  - A parameter signifying asynchronous or not
- The `send` method sends the request
- The `getPlace` method implements this

# The Request Phase

```
// function getPlace  
  
Function getPlace(zip) {  
    var xhr = new XMLHttpRequest();  
    xhr.onreadystatechange = receivePlace;  
    xhr.open("GET", "getCityState.php?zip=" +  
        zip, true);  
    xhr.send(null);  
}
```

# The Response Document

- The response from the server is created by looking up the zip code
  - A local hash of zip codes is used for simplicity
- A string with the city and state is sent as the response
- The example is in PHP

# The Response Document

```
<?php
// getCityState.php
$cityState = array("81611" => "Aspen, Colorado",
                  "81411" => "Bedrock, Colorado",
                  ...
                  );

header("Content-Type: text/plain");
$zip = _GET("zip");
if (array_key_exists($zip, $cityState))
    print $cityState[$zip];
else
    print " , ";
?>
```

# The Receiver Phase

- The function that parses the response must have access to the XMLHttpRequest object
- This cannot be global since there may be multiple outstanding requests at any time
- The callback becomes an anonymous function which is defined in the getPlace method and keeps references to the XMLHttpRequest object held in a local variable
- The response handler only acts if the readyState is 4, meaning the response is complete

# The Receiver Phase

```
// function receivePlace

function receivePlace() {
    if (xhr.readyState == 4 && xhr.status == 200) {
        var result = xhr.responseText;
        var place = result.split(', ');
        if (document.getElementById("city").value == "")
            document.getElementById("city").value = place[0];
        if (document.getElementById("state").value == "")
            document.getElementById("state").value = place[1];
    }
}
```

# Cross-Browser Support

- Older Microsoft browsers uses a different approach to getting the request object
- Testing the existence of `window.XMLHttpRequest` differentiates the browsers
- In the older browsers  

```
new  
ActiveXObject ("Microsoft.XMLHTTP")
```
- creates the object needed