Module 5

Communicating with a Remote Server

Module Overview

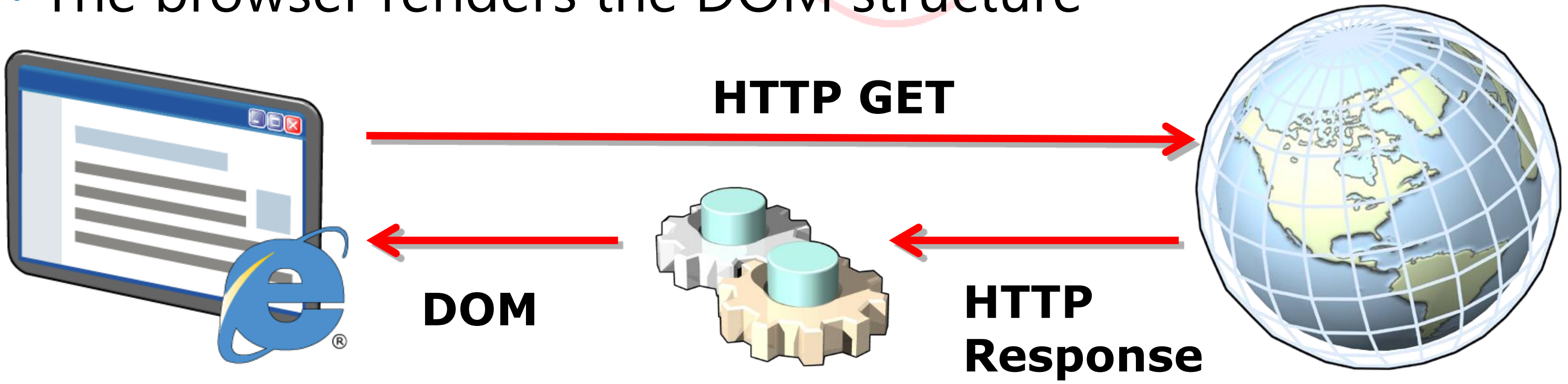
- Sending and Receiving Data by Using the XMLHttpRequest Object
- Sending and Receiving Data by Using the jQuery Library

Lesson 1: Sending and Receiving Data by Using the XMLHttpRequest Object

- How a Browser Retrieves Web Pages
- Using the XMLHttpRequest Object to Access Remote Data
- Handling HTTP Errors
- Consuming the Response
- Handling an Asynchronous Response
- Transmitting Data with a Request

How a Browser Retrieves Web Pages

- A web browser issues HTTP GET requests to fetch a web page to display
  - The response is parsed into a DOM structure
  - The browser renders the DOM structure



- Elements with a **src** attribute can initiate further HTTP GET requests
- JavaScript code can trigger HTTP GET requests

Using the XMLHttpRequest Object to Access Remote Data

- To send an HTTP request:

1. Create a new **XMLHTTPRequest** object
2. Specify the HTTP method and URL
3. Set the request header
4. Send the request

```
var request = new XMLHttpRequest();
var url = "http://contoso.com/resources/...";
request.open( "GET", url );
request.send();
```

- Requests are asynchronous by default
  - To block and wait for a response:

```
request.open( "GET", url , false);
```

Handling HTTP Errors

- Check the status code of the **XMLHttpRequest** object to verify that the request has been sent:

```
var request = new XMLHttpRequest();
request.open("GET", "/luckydip/enter");
request.send( );
…
if( request.status != 200 ) {
    alert( "Error " + request.status + " - " + request.statusText );
}
```

- Wrap your code in a **try…catch** block to handle any unexpected network errors

Consuming the Response

- Determine the type of data in the response
- Read the response data from the **responseText** property

```
var request = new XMLHttpRequest();
…
var type = request.getResponseHeader();
    switch( type ) {
        case "text/xml" :
            return request.responseXML;
        case "text/json" :
            return JSON.parse(request.responseText);
        default :
            return request.responseText;
}
```

Handling an Asynchronous Response

- Create an event handler for the **readystatechange** event
- Check that the **readyState** of the **XMLHttpRequest** object is set to 4

```
request.onreadystatechange = function () {
    if (request.readyState === 4) {
        var response = JSON.parse(request.responseText);
        ...
    }
};
```

Transmitting Data with a Request

- To send data to a server:

1. Serialize the data
2. Set the **Content-Type** property of the request header
3. Transmit the data by using the HTTP **POST** method

```
var data = JSON.stringify(...);
var request = new XMLHttpRequest();
var url = ...;
request.open("POST", url );
request.setRequestHeader("Content-Type", "text/plain" );
request.send(data);
```

Lesson 2: Sending and Receiving Data by Using the jQuery Library

- Using the jQuery Library to Send Asynchronous Requests
- Using the jQuery ajax() Function
- Serializing Forms Data by Using jQuery
- Demonstration: Communicating with a Remote Data Source

Using the jQuery Library to Send Asynchronous Requests

- The jQuery library provides asynchronous methods for sending requests and handling the response:

```
var response;

$.get(' http://contoso.com/resources/...', function(data) {
  response = data;
}).error(function() {
  alert("error occurred during get operation");
});
```

```
$.getJSON( url, body, callback );
```

```
$('#container').load( url, body, callback );
```

Using the jQuery ajax() Function

- The jQuery **ajax()** function provides additional properties and finer control over HTTP requests

```
$.ajax({
        url: '/luckydip/enter',
        type: 'GET',
        timeout: 12000,
        dataType: 'text'
}).done(function( responseText ){
    $('#answer').text( responseText );
}).fail(function() {
    alert('An error has occurred – you may not have been entered');
});
```

Serializing Forms Data by Using jQuery

- To include forms data in a request, use the **data** property:

```
$.ajax({
        url: '/luckydip/enterWithName',
        type: 'POST',
        timeout: 12000,
        dataType: 'text',
        data: {
            firstName: myForm.fname.value,
            lastName: myForm.lname.value
});
```

- To retrieve input data directly from a form, use the **serializeArray()** function

Demonstration: Communicating with a Remote Data Source

In this demonstration, you will learn about the tasks that you will perform in the lab for this module.

Lab: Communicating with a Remote Data Source

- Exercise 1: Retrieving Data
- Exercise 2: Serializing and Transmitting Data
- Exercise 3: Refactoring the Code by Using the jQuery ajax Method

Estimated Time: 60 minutes

Lab Scenario

You have been asked modify the Schedule page for the ContosoConf website. Previously, the session data was provided as a hard-coded array of data and the JavaScript code for the page displayed the data from this array. However, session information is not static; it may be updated at any time by the conference organizers and stored in a database. A web service is available that can retrieve the data from this database, and you decide to update the code for the Schedule page to use this web service rather than the hard-coded data currently embedded in the application.

In addition, the conference organizers have asked if it is possible for conference attendees to be able to indicate which sessions they would like to attend. This will enable the conference organizers to schedule popular sessions in larger rooms. The Schedule page has been enhanced to display star icons next to each session. An attendee can click a star icon to register their interest in that session. This information must be recorded in a database on the server, and you send this information to another web service that updates the corresponding data in the database.

A session may be very popular, so the web service will return the number of attendees who have selected it. You will need to handle this response and display a message to the attendee when they have selected a potentially busy session.

Module Review and Takeaways

- Review Question(s)