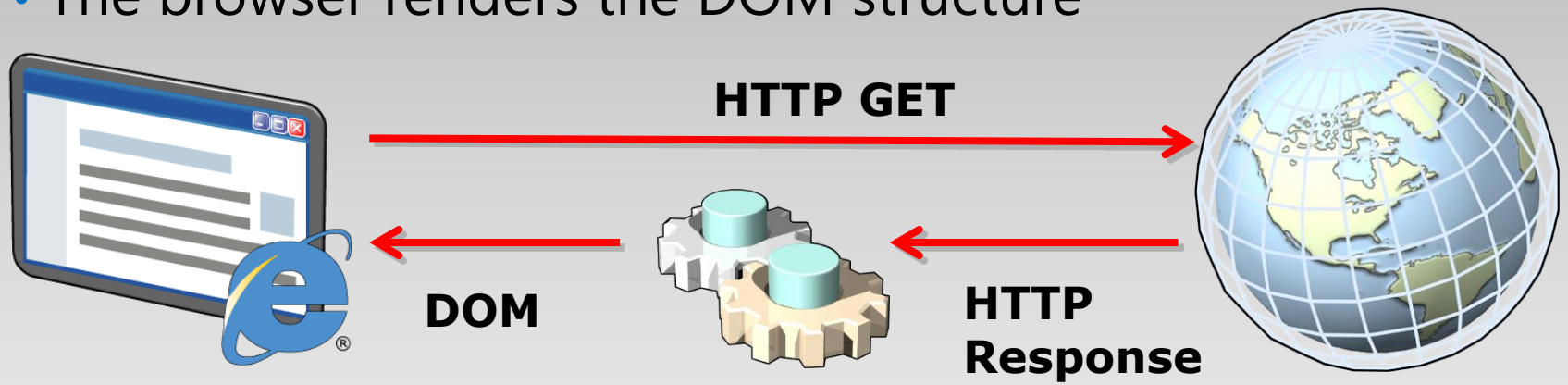
AJAX

Agenda

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

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 URL and HTTP method
3. Send the request

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

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

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

```
request.open( String method, String url, Boolean async);
```

HTTP **GET** or **POST**

- **GET** to request a known resource without parameters
- **POST** to send parameterized data to the server, most commonly form data
- **HEAD** specifies that the response should only be header information
  - For example, to get a summary of a large resource ahead of actually downloading it.

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);
        ...
    }
};
```

- **0**: The **XMLHttpRequest** object is has not been opened.
- **1**: The **XMLHttpRequest** object has been opened.
- **2**: The **XMLHttpRequest** object has sent a request.
- **3**: The **XMLHttpRequest** object has started to receive a response.
- **4**: The **XMLHttpRequest** object has finished receiving a response.

Transmitting Data with a Request

- To send data to a server:

1. Serialize the data – JSON format
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", "application/json");
request.send(data);
```

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;
}).fail(function() {
  alert("error occurred during get operation");
});
```

```
$.getJSON //data is passed in JSON format rather than as text
```

```
$('#container').load(url, body, callback); // body and callback is optional
```

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').textContent( 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