JavaScript

Devdatta Kulkarni

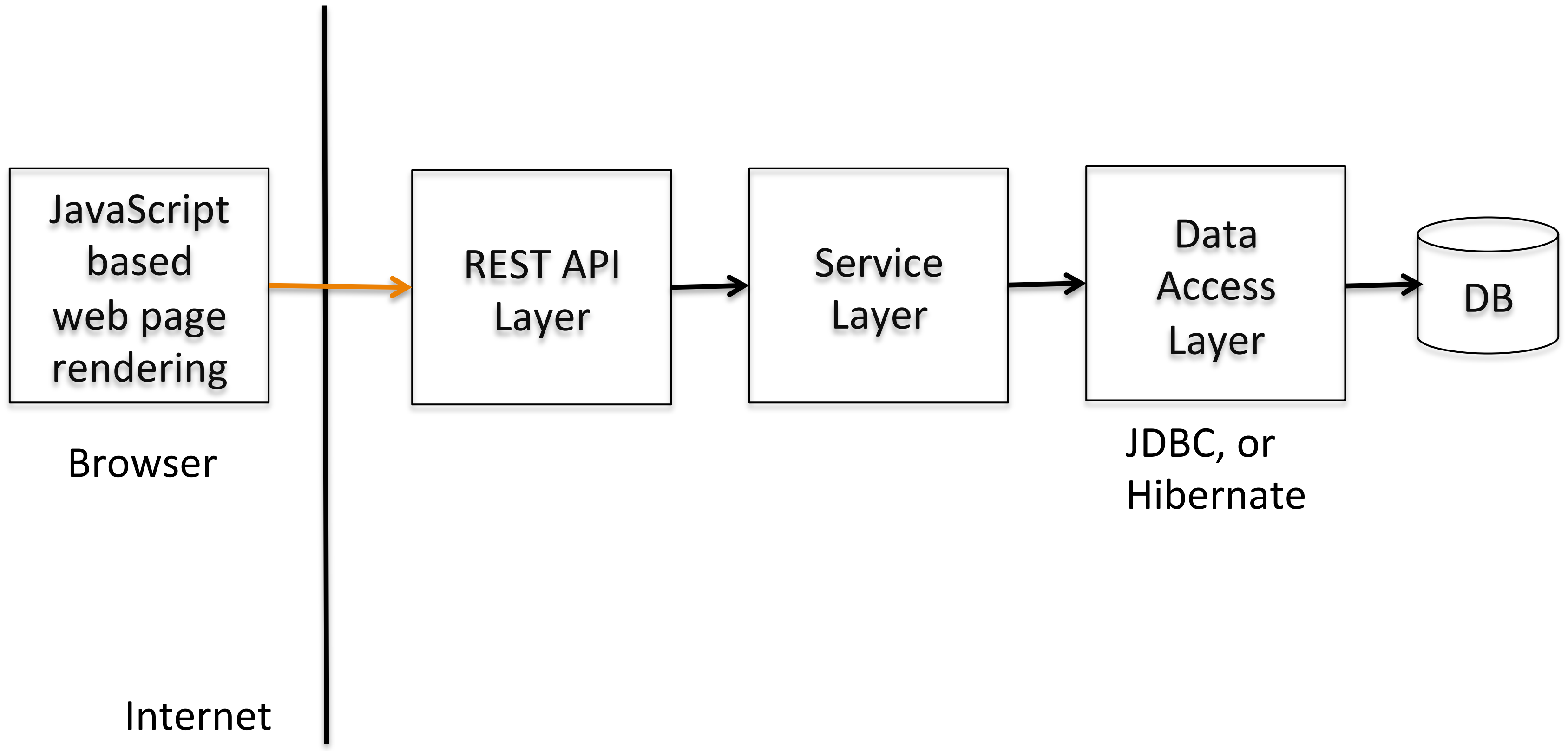
Agenda for April 25, 2016

- Last class
  - JavaScript
- Today
  - Assignment 6
  - JavaScript
  - Cloud computing
- Announcements
  - Next class will be covered by Venket
    - I am attending OpenStack conference happening in Austin this week

JavaScript

- What is it?
  - Scripting language that controls how web browsers render web pages
- Purpose
  - Make web pages *dynamic*
    - Client side error checking of forms
    - Taking different actions based on *events* happening on the browser
    - Dynamically updating content on the browser
- Where is the script specified?
  - Within <script></script> elements which is included within
    - the <head></head> element
    - the <body></body> element

Modern Web Applications



JavaScript

- Tutorial to follow:
  - http://www.w3schools.com/js/
- Where to add the scripts?
  - src/main/webapp/*.html
  - src/main/webapp/*.js
  - Show Example

JavaScript Example

- https://github.com/devdattakulkarni/ModernWebApps/tree/master/JavaScript-Example
- http://localhost:8080/js-example/test.html
- http://localhost:8080/js-example/ajaxexample.html

JavaScript main concepts

- Document Object Model (DOM) associated with the web page
- In HTML, JavaScript statements are "instructions" to be "executed" by the web browser.
- Event handling model
  - Event bubbling (inner most to outer most)
  - Event capturing (outer most to inner most)
- Asynchronous JavaScript (AJAX)
  - To interact with REST API
  - Cross-origin resource sharing (CORS)

JavaScript

- Script
- Comments
  - //
  - /* */
- Case Sensitive

  lastName = "Doe";
  lastname = "Peterson”;

  lastName and lastname are different variables
- Hyphens are not allowed in JavaScript. Hyphen is reserved for subtractions.

DOM

- The “document” object
  - This object represents the web page
  - http://www.w3schools.com/js/js_htmldom.asp

- Accessing an element in the DOM
  - Use getElementById method
    - document.getElementById(“<some_id>”)
      - <some_id> is the value of the id attribute of some element in the HTML page

- Accessing contents of an element from the DOM
  - Use the “innerHTML” property

Finding HTML Elements

- document.getElementById(*id*)
  - Find an element by element id
- document.getElementsByTagName(*tagName*)
  - Return a list of elements by tag name
- document.getElementsByClassName(*className*)
  - Return a list of nodes by class name

Changing HTML Elements

- *element*.innerHTML
  - Change the inner HTML of an element

- *element.attribute*
  - Change the attribute of an HTML element

Adding and Deleting Elements

- document.createElement(*elementName*)
  - Create an HTML element
- *parentNode*.appendChild(*childNode*)
  - Appends the *childNode* as the last child to the *parentNode*
- *parentNode*.removeChild(*childNode*)
  - Removes the *childNode* from the *parentNode*
- *parentNode*.replaceChild(*new, current*)
  - Replace *current* node with *new* node

HTML DOM Events

- A JavaScript can be executed when an event occurs on the web page
- Examples of HTML events:
  - When a user clicks the mouse
  - When a web page has loaded
  - When an image has been loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted

HTML DOM EventListener

- Adding an event listener
  - document.getElementById("myBtn").addEventListener("click", displayDate);
  - document.getElementById("myBtn").onclick = displayDate;
- The addEventListener() method attaches an event handler to the specified element.
- It does not overwrite existing event handlers.
- Many event handlers can be attached to one element

HTML DOM EventListeners

- *element*.addEventListener(*event, function, useCapture*);
  - The first parameter is the type of the event (like "click" or "mousedown").
  - The second parameter is the function we want to call when the event occurs.
  - The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.
    - Default is ‘false’, which means the event bubbling model will be used

Event Propagation Model

- Event Propagation Model
  - Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's ``click'' event should be handled first?
- Event Bubbling
  - In *bubbling,* the inner most element's event is handled first and then the outer element's
- Event Capturing
  - In *capturing,* the outer most element's event is handled first and then the inner element's

BOM

- Browser Object Model
  - http://www.w3schools.com/js/js_window.asp
- The “window” object represents the browser’s window
  - All global JavaScript objects, functions, and variables automatically become members of the window object.
  - Global variables are properties of the window object.
  - Global functions are methods of the window object.
  - The document object (of the HTML DOM) is a property of the window object

window.document.getElementById("header"); and
document.getElementById("header"); are same

AJAX

- http://www.w3schools.com/ajax/
- AJAX
  - Asynchronous JavaScript and XML.
- The XMLHttpRequest Object
  - All modern browsers support the XMLHttpRequest object (IE5 and IE6 use an ActiveXObject).
  - The XMLHttpRequest object is 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.

AJAX - Details

- Send a Request to Server
  - open(*method,url,async*)
    - *method*: the type of request: GET or POST
      url: the location of the file on the server
      async: true (asynchronous) or false (synchronous)
- Server Response
  - responseText get the response data as a string
    responseXML get the response data as XML data
- The onreadystatechange event
  - http://www.w3schools.com/ajax/ajax_xmlhttprequest_onreadystatechange.asp

Same Origin Policy

- http://tools.ietf.org/html/rfc6454

- A JavaScript running on a web browser is able to interact with web resources arising from the *same origin as that of the script*
- Same origin:
  - Two URIs are part of the same origin (i.e., represent the same security principal) if they have the same *scheme*, *host*, and *port*
  - Scheme: http/https

Same Origin Policy

- Following have same origin
  - http://example.com/
  - http://example.com:80/
  - http://example.com/path/file
- Different origin from each other
  - http://example.com/
  - http://example.com:8080/
  - http://www.example.com/
  - https://example.com:80/
  - https://example.com/
  - http://example.org/
  - http://ietf.org/

Same origin policy

- Applies only to AJAX requests
- Does not apply to loading scripts or images
  - http://stackoverflow.com/questions/5707363/same-origin-policy-and-external-scripts

Frameworks and libraries

- Google's AngularJS
  - MVC framework for JavaScript
  - http://www.w3schools.com/angular/default.asp
- Facebook's ReactJS
  - Only the "view" layer
  - Has the notion of a virtual DOM; allows partial DOM updates
- Twitter's Bootstrap
  - Framework for web UI development
  - http://www.w3schools.com/bootstrap/
- JQuery
  - Library for building JavaScript based applications

References

- Browser security
  - https://code.google.com/p/browsersec/wiki/Main
- How Ad Servers work?
  - http://www.adopsinsider.com/ad-serving/how-does-ad-serving-work/

Cross-site scripting

- https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29