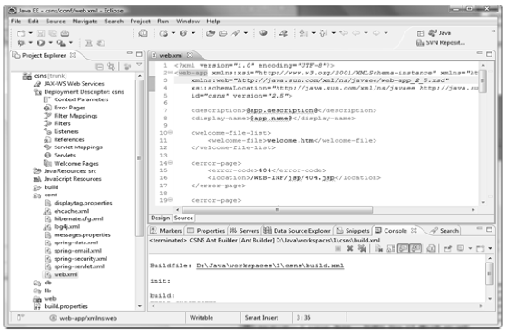
CS520 Web Programming
Introduction to Ajax and jQuery

Chengyu Sun
California State University, Los Angeles

The Desktop Experience



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="ht
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.su
    id="csns" version="2.5">

    <description>@app.description@</description>
    <display-name>@app.name@</display-name>

    <welcome-file-list>
        <welcome-file>welcome.html</welcome-file>
    </welcome-file-list>

    <error-page>
        <error-code>404</error-code>
        <location>/WEB-INF/jsp/404.jsp</location>
    </error-page>

    <error-page>
```

```
<terminated> CSNS Ant Builder [Ant Builder] D:\Java\workspaces\1\csns\build.xml

Buildfile: D:\Java\workspaces\1\csns\build.xml

init:

build:
```

The Desktop Advantage

- Large selection of GUI components
- Interactive
  - Rich event model
- Responsive
  - Partial redraw

HTML Event Models

- HTML 4 Event Model
  - HTML 4.01 Specification - http://www.w3.org/TR/REC-html40/interact/scripts.html#h-18.2.3
  - Limited but widely supported
- Standard Event Model
  - DOM Level 2 HTML Specification - http://www.w3.org/TR/DOM-Level-2-Events/events.html
- Browser specific event models

Events and Event Handler

- Events
  - onfocus, onblur, onkeypress, onkeydown, onkeyup, onclick, ondbclick, onmousedown, onmouseup, onmousemove, onmouseover …
- Specify event handler
  - <element event="code">
  - For example:

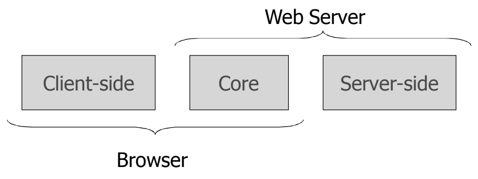
```
<button onclick="clickHandler();">click</button>
```

Example: Event Handling

- `j1.html`
  - Uses X Library from http://cross-browser.com/
  - *Event handler*
    - *Written in JavaScript*
    - *Modify the HTML document*

JavaScript

- Interpreted language
- Originally developed by Netscape
- Syntax is similar to Java



Core JavaScript

- Mainly covers language syntax, which is similar to Java
- Some "un-Java-like" language features
  - Object creation
  - Functions as first-class citizens

Object Creation – Approach 1

```
var car = new Object();
car.make = 'Honda';
car.model = 'Civic';
car.year = 2001;

var owner = new Object();
owner.name = 'Chengyu';

car.owner = owner;
```

- A JavaScript object consists of a set of properties which can be added dynamically

Object Creation – Approach 2

```
var car = {
  make: 'Honda',
  model: 'Civic',
  year:  2001,
  owner: {
    name: 'Chengyu'
  }
};
```

- Object Literal

Object Creation – Approach 3

```
var car = {
  'make': 'Honda',
  'model': 'Civic',
  'year':  2001,
  'owner': {
    'name': 'Chengyu'
  }
};
```

- JSON (JavaScript Object Notation)

Functions as First-class Citizens

- In JavaScript, functions are considered objects like other object types
  - Assigned to variables
  - Assigned as a property of an object
  - Passed as a parameter
  - Returned as a function result
  - *Function literals* (i.e. functions without names)

Function Examples

```
function foo() {
  alert('foo');
}
```

Regular function creation

```
bar = function() {
  alert('bar');
}
```

- Function literal
- Function assignment

```
setTimeout( bar, 5000 );
```

Function as parameter

```
setTimeout( function() {
  alert('foobar');},
  5000 )
```

Function literal as parameter

Client-Side JavaScript

- Embed JavsScript in HTML
  - <script>
    - `type="text/javascript"`
    - `language="JavaScript"`
    - `src="path_to_script_file"`
- Run inside a browser

Processing an HTML Document

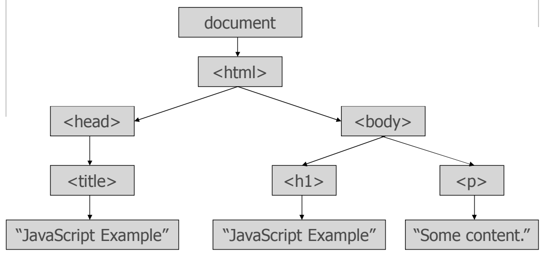
```
<html>
<head><title>JavaScript Example</title></head>
<body>
    <h1>JavaScript Example</h1>
    <p>Some content.</p>
</body>
</html>
```

- As a text file – very difficult
- As an object

Document Object Model (DOM)

- Representing documents as objects so they can be processed more easily by a programming language

DOM Representation



Manipulate a Document

- Find Elements
- Modify Elements
- Create Elements

Find Elements

- document.getElementById()
- document.getElementsByName()
- document.getElementsByTagName()

Modify Elements ...

- HTMLElement properites and methods
  - IE
    - innerHTML
    - innerText
    - insertAdjacentHTML()
    - insertAdjacentText()
  - Netscape/Mozilla
    - innerHTML
  - Element-specific

... Modify Elements

- node
  - setAttribute(), removeAttribute()
  - appendChild(), removeChild()
  - insertBefore(), replaceChild()

Create Elements

- document
  - createElement()
  - createTextNode()

Example: Document Manipulation

- `j2.html`
  - Read and display the text input
  - *Display "Hello <name>"??*
  - *Add text input to table??*

Create Desktop-Like Web Applications

- Interactivity
  - HTML events
  - JavaScript for event handling
  - DOM for document manipulation
- Responsiveness??

Communicate with Server

- The *synchronous* request-response model is still a limiting factor in responsiveness
- Solution: XMLHttpRequest
  - A JavaScript object
    - Send request and receive response
  - Response can be handled *asynchronously*
    - Do not need to wait for the response

Understand Asynchronous …

- Synchronous

```
send( request );

// wait for response

process( response );

// do other things
…
```

- Asynchronous

```
send( request );

// don't wait for response

process( response );

// do other things
…
```

What's the problem??
What's the solution??

… Understand Asynchronous

```
// callback function
function foo( response )
{
  process( response );
}
```

Same as handling events like click.

- Asynchronous

```
// set a callback function
// which will be called
// when the response comes
// back
… …

send( request );

// do other things
…
```

An XMLHttpRequest Example

- `a1.html`
  - A client scripts sends an XMLHttpRequest
  - A servlet responses with a random number
  - When the message arrives on the client, a *callback function* is invoked to update the document

About the Example

- clickHandler()
- newXMLHttpRequest()
- updateDocument()
- getReadyStateHandler()

XMLHttpRequest - Properties

- onreadystatechange
- readyState
  - 0 – uninitialized
  - 1 – loading
  - 2 – loaded
  - 3 – interactive
  - 4 – complete
- status
- statusText
- responseBody
- responseStream
- responseText
- responseXML

XMLHttpRequest - Methods

- abort()
- getAllResponseHeaders()
- getResponseHeader(header)
- open(method, url, asyncFlag, username, password)
  - asyncFlag, username, password are optional
- send(messageBody)
- setRequestHeader(name, value)

So What is Ajax?

- Asynchronous JavaScript and XML
  - http://www.adaptivepath.com/ideas/essays/archives/000385.php
  - JavaScript + XMLHttpRequest
- Characteristics of Ajax
  - Non-blocking – the server response is handled asynchronously with a callback function
  - Partial page update using JavaScript

More About AJAX

- XMLHttpRequest used to be an IE specific feature that received little attention
- It's all started by Google Maps
- The beginning of "Web 2.0"

Key Elements of an Ajax Operation

Client	Server
Event Event handler • Create a XMLHttpRequest • Attach a callback function • Send the request Callback function • Process the response • Update the HTML Page	Process the request Send back a response

Problems of Plain JavaScript + XmlHttpRequest

- Each browser has their own JavaScript implementation
  - Code that works on some browsers may not work on others
- Lack of pre-made GUI components
- Implementing Ajax operations is quite tedious

JavaScript/Ajax Frameworks and Libraries

- http://en.wikipedia.org/wiki/List_of_Ajax_frameworks
  - Cross-browser compatibility
    - New JavaScript API, e.g. X Lib, JQuery
    - New language, e.g. ZK, Taconite
  - Pre-made, Ajax-enabled GUI component
  - Simplify the implementation of Ajax operations

One Library to Rule Them All - jQuery

- jQuery - http://jquery.com/
- jQuery UI - http://jqueryui.com/
- The market share of jQuery
  - http://trends.builtwith.com/javascript/javascript-library

A jQuery Example

- `j3.html`
  - Usage
  - jQuery wrapper
  - Selectors
  - Elements
  - Events and event handling
  - DOM manipulation

Use jQuery Library

- http://jquery.com/download/
  - Local copy vs. CDN hosted
  - `1.x` vs `2.x`

jQuery Wrapper

- `jQuery()` or `$()`
  - Return a collection of matched elements either found in the DOM based on passed argument(s) or created by passing an HTML string.

```
$( "input[name='firstName']" )

$( "#who" )          $( "#t1" )
```

Selector

Basic Selectors

- By id
  - `$(“#foo”)`
- By tag name
  - `$(“div”)`
- By CSS class
  - `$(“.foo”)`
- By attribute
  - `$(“[name]”)`
  - `$(“[name=‘joe’]”)`

Combine Selectors

- Select all the `<div>` elements with CSS class `foo` and an attribute `bar`

`$(“div.foo[bar]”)`

- Select all the `<div>` elements, and all the elements with CSS class `foo`, and all the elements with an attribute `bar`

`$(“div,.foo,[bar]”)`

Other Selectors and Filters

- Form selectors
- Hierarchy selectors
- Filters

What Can We Do With An Element

- Get and set
  - `html()`
  - `attr()`
  - `prop()`
  - `val()`
- Manipulate CSS
  - `addClass()`
  - `removeClass()`
  - `toggleClass()`
  - `hasClass()`

Property `tagName`

```
<input name="username" value="cysun" />
```

Attribute `name`   `val()`

Typical Event and Event Handling in jQuery

Event   Event Handler

```
$("#click").click( function(){
      ... ...
});
```

Unobtrusive JavaScript:
separate style, behavior, and structure.

<button id="click" ~~onclick="display();~~"> Click Me</button>

Document Ready Event

- Triggered when the DOM hierarchy of the HTML document is fully constructed

```
$( document ).ready( handler )
```

```
$().ready( handler )
```
(not recommended)

`$( handler )`

Other Events

- Mouse events
  - `.click()`
  - `.dbclick()`
  - …
- Keyboard events
  - `.keyup()`
  - `.keydown()`
  - `.keypress()`
  - …
- Form events
  - `.change()`
  - `.submit()`
  - …
- Browser events
  - `.resize()`
  - …
- Document events

DOM Manipulation

- Insertion
  - Around (i.e. parent)
  - Inside (i.e. children)
  - Outside (i.e. sibling)
- Removal
- Replacement

Example:

```
$("#t1").append("<tr><td>John</td><td>Doe</td></tr>");
```

Example: jQuery Tic Tac Toe

- j4.html

	o	x
	x	
o		

AJAX with jQuery

- http://api.jquery.com/category/ajax/
- **`$.ajax`**`( url [, settings])`
  - `url`: request URL
  - `data`: data to be sent to the server
  - `success`: a function to be called if the request succeeds
- Example: `a2.html`

Example: Who's Online (I)

Who's Online

• cysun
• admin

- `WhosOnlineService` bean
  - In-memory storage like the "application scope" in servelt programming
- `Login`, `Logout`
- `WhosOnline` controller and view

Who's Online (II)

- Use an AJAX request to get the list of online users as a JSON object, then use the JSON object to populate the list
  - JSON
  - Jackson
  - JSON view in Spring
  - More jQuery

Java JSON Library

- Serialize and de-serialize Java objects
- Jackson
  - http://wiki.fasterxml.com/JacksonHome
  - Maven dependency: `jackson-databind`
- Gson
  - https://code.google.com/p/google-gson/

Jackson and Spring

- Add a `BeanNameViewResolver`
- Add a `MappingJackson2JsonView`
- Model objects passed to the view will be automatically serialized to JSON

Who's Online (III)

- Automatically update the Who's Online list
  - *How??*

Repeated Requests

- `Refresh` response header, or
- `setInterval(function, interval)` in JavaScript

Asynchronous Request Processing

- Introduced in Servlet 3.0 specification
- Supported by Spring 3.2+
  - http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-async

Enable Asynchrous Request Processing

- web.xml
  - Add `<async-supported>` to servlets and filters
  - Add `<dispatcher>` to filter mapping
- <servlet>-context.xml
  - Add `<mvc:async-support>` to `<mvc:annotation-driven>`

DeferredResult …

- http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/context/request/async/DeferredResult.html

… DeferredResult …

Controller code:

```
@RequestMapping("/whosonline.deferred.json")
@ResponseBody
public DeferredResult<String> wosDeferred()
{
    DeferredResult<String> deferredResult =
        new DeferredResult<String>();
    return result;
}
```

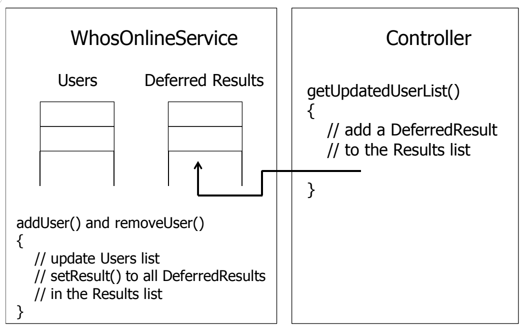
Some other code:

```
…
deferredResult.setResult(data);
```

… DeferredResult

- Controller can return immediately so the associated servlets and filters can finish and their resources released
- The response remains open until `setResult(data)` is called, at which point `data` is sent back to the client as the response body (per `@ResponseBody`)

How Will Who's Online Work



Use Jackson's ObjectMapper

- http://fasterxml.github.io/jackson-databind/javadoc/2.3.0/com/fasterxml/jackson/databind/ObjectMapper.html
  - JSON to Java
    - `readValue( input, type )`
  - Java to JSON
    - `writeValue( output, object )`

Potential Issues

- Concurrency issues
  - E.g. multiple users login/logout at the same time
- Speed issues
  - E.g. requests coming in faster than we can setResult()
- Various exceptions
  - E.g. connection timeout

Some Possible Solutions

- Use thread-safe data structures in `java.util.concurrent`
- Keep track of which client has already been served