JQUERY AND AJAX

Dynamic HTML (DHTML)

- Manipulating the web page's structure is essential for creating a highly responsive UI

- Two main approaches
  - Manipulate page via plain JS
  - Manipulate page using JS + library (e.g., jQuery)

Document Object Model (DOM)

- Fancy name for the web page's structure

- Web page is basically a tree structure
  - One node per HTML element
  - Each node can have attributes

Rewriting using innerHTML attribute

```
<span id="stuff"></span>
<form><input id="inpt" onchange="doit()"></form>
<script>
function doit() {
  document.getElementById("stuff").innerHTML =
  document.getElementById("inpt").value;
}
</script>
```

Rewriting the contents of a span. NOTE: There is a security problem in the code above. See next slide.

Assigning the .innerText instead

```
<span id="stuff"></span>
<form><input id="inpt" onchange="doit()"></form>
<script>
function doit() {
  document.getElementById("stuff").innerText =
  document.getElementById("inpt").value;
}
</script>
```

Rewriting the contents of a span. NOTE: There is a browser-compatibility problem in the code above. See next slides.

Welcome to jQuery

- jQuery is one of many available libraries that
  - Provide functions for manipulating the web page
    - With fairly good performance
  - Help to keep your JS code clean
    - Indirectly help to protect security (somewhat)

- Those are the benefits of using such a library
- The downside is that you have an extra dependency and need to learn a new library

Getting started with jQuery

- Download a copy of the jquery JS file and store it on your hard drive
- Reference the JS file in your HTML
- Access the jQuery functions via the $ object

Simple example

```
<script src="jquery-1.8.2.min.js"></script>
<span id="stuff"></span>
<form><input id="inpt" onchange="doit()"></form>
<script>
function doit() {
  $("#stuff").text($("#inpt").val());
}
</script>
```

Rewriting the contents of a span. No security problems or cross-browser compatibility problems.

Warning: You need clean HTML

- If you want jQuery to perform reliably…
  - Always include <html></html> tag
  - Always put this line before your <html> tag

    <!DOCTYPE html>
    - This tells the browser to operate in "standards" mode.
  - Always include "" around your attribute values

    blah blah

Examples of things you can do with jQuery

- Read the contents of DOM nodes (tag)
- Modify the contents of DOM nodes
- Modify the appearance of DOM nodes
- Create and attach new DOM nodes
- Remove DOM nodes
- Run a function right when the page is ready
- Add and remove event handlers
- Retrieve content from a web server
- Send content to a web server

Example: Modifying DOM appearance

```
<!DOCTYPE html><html><head>
<script src="jquery-1.8.2.min.js"></script>
<style>
.nice {background-color: orange; color: white;}
</style></head><body>
<div id="clickme" onclick="toggle()">Click me!</div>
<script>
function toggle() {
  var els = $("#clickme");
  if (!els.hasClass('nice'))
    els.addClass('nice');
  else
    els.removeClass('nice');
}
</script>
```

Example: Creating new nodes

```
<!DOCTYPE html><html><head>
<script src="jquery-1.8.2.min.js"></script>
</head><body>
<div id="mydiv" onclick="addstuff()">Add stuff</div>
<script>
function addstuff() {
  for (var i = 0; i < 10; i++) {
    $('#mydiv').append('<div class="kid">'+i+'</div>');
  }
}
</script>
```

Example: Removing nodes

```
<!DOCTYPE html><html><head>
<script src="jquery-1.8.2.min.js"></script>
</head><body>
<div id="mydiv" onclick="addstuff()">Add stuff</div>
<script>
function addstuff() {
  var kids = $(".kid"); // this creates a "wrapped set" around all nodes with class=kid
  if (!kids.length) {
    for (var i = 0; i < 10; i++)
      $('#mydiv').append('<div class="kid">'+i+'</div>');
  } else {
    kids.remove();
  }
}
</script>
```

Example: Running code on page ready

```
<!DOCTYPE html><html><head>
<script src="jquery-1.8.2.min.js"></script>
</head><body>
<div id="mydiv" onclick="addstuff()">Add stuff</div>
<script>
function addstuff() {
  var kids = $(".kid");
  if (!kids.length) {
    for (var i = 0; i < 10; i++)
      $('#mydiv').append('<div class="kid">'+i+'</div>');
  } else {
    kids.remove();
  }
}
$(addstuff);
</script>
```

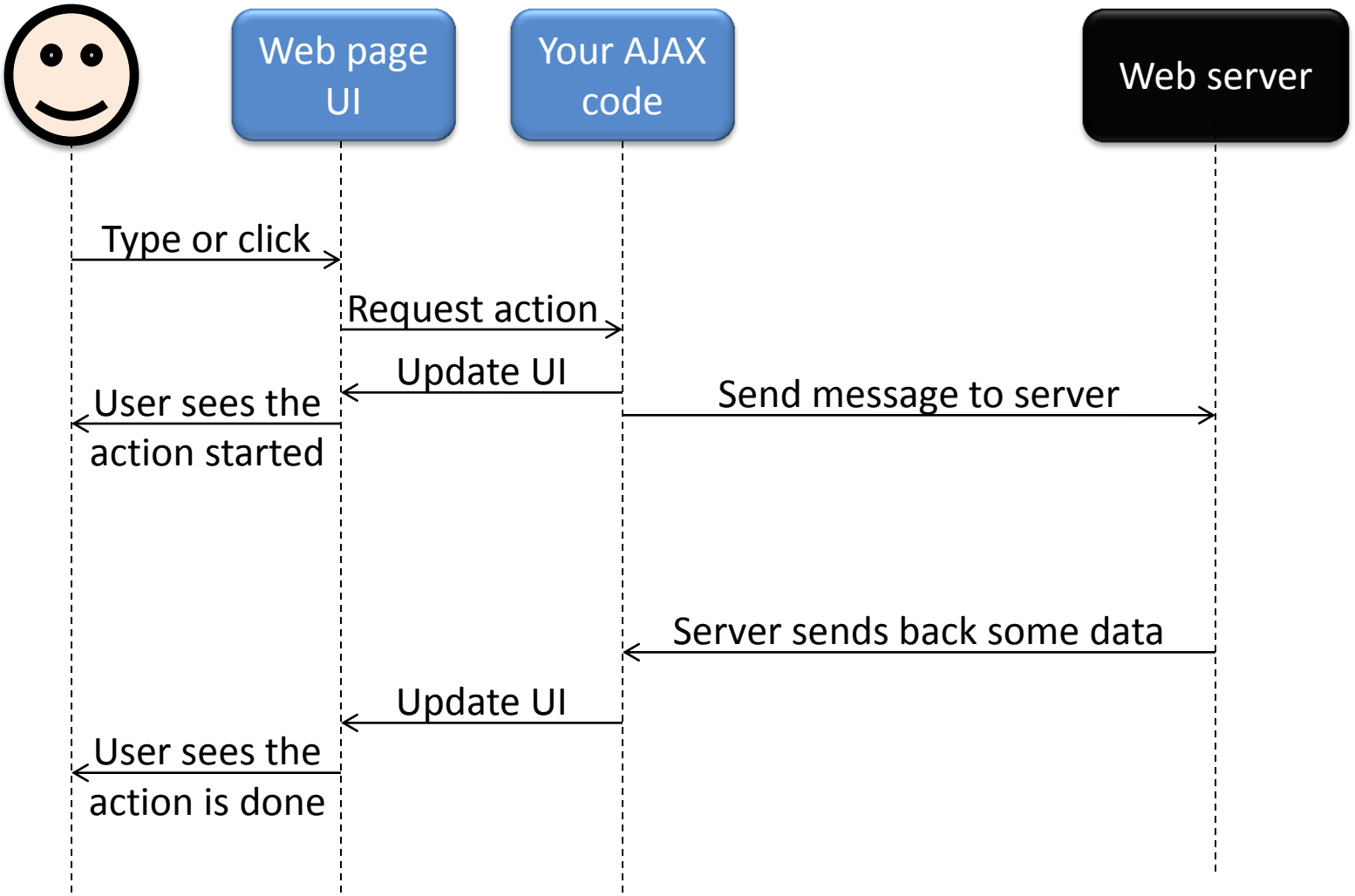
Example: Manipulating event handlers

```
<!DOCTYPE html><html><head>
<script src="jquery-1.8.2.min.js"></script>
<style>
.nice {background-color: orange; color: white;}
</style></head><body>
<div id="clickme">Click me!</div>
<script>
function toggle() {
  var els = $("#clickme");
  if (!els.hasClass('nice'))
    els.addClass('nice');
  else
    els.removeClass('nice');
}
$("#clickme").click(toggle);
</script>
```

Coolest part of jQuery: Simplifies AJAX

- Old school (synchronous full page refresh)
  - Click a link, wait for page to load, submit a form, wait for page to load, click a link, wait for page…
- New school (asynchronous partial refresh)
  - Click a link, part of page quickly changes, fill out a form, page immediately responds while server gets data, etc.
    - More complicated, but much more usable

How asynchronous partial refresh works



How it works in detail

- User types or clicks: need an event handler
- UI requests some action: need a JS function
- UI shows it started: need a DIV to update
  - Should be clear, so user sees it started
- Send message to server: need AJAX code
- Server eventually replies: need callback JS
- UI shows it finished: need a DIV to update
  - Should show the result to the user

A very simple web page and XML

```
<!DOCTYPE html><html><head>
<script src="jquery-1.8.2.min.js"></script>
</head><body>
<div id="clickme" onclick="startAjax()">Click me</div>
<script>
function startAjax() {
  $("#clickme").text("Calling server");
  $.ajax({url:"somefile.xml",
     success:callbackFunction, error:errorFunction}
  );
}
function callbackFunction(data,info) {
  $("#clickme").text("result:"+data);
}
function errorFunction(data,info) {
  $("#clickme").text("error occurred:"+info);
}
</script>
</body></html>
```

somefile.xml

```
<?xml version="1.0"?>
<root>
  <entry name="blah">ok</entry>
</root>
```

Key things to note

- There's an element with an onclick handler
- And the onclick handler calls

  $.ajax({url:myurl, success:jsFn, error:jsFn});
- And each JS function looks like

  function myjsfunction(data, info) {…}
- Inside the JS function, update the UI using

  $("#myelementid").text("whatever");

So where can you load data from?

- In general, you can only load data from the same web site that your main html came from
  - This is called the "same origin policy"

- When you're working from the file system…
  - Firefox 13 & Internet Explorer 9 let you load files
  - Chrome 22 does not let you load other files
  - Other versions & other browsers may vary!

So what is this XML you speak of?

- Basically a tree-like structure, similar to the document object model you get from HTML
  - In fact, some of the same W3C official standards apply to both XML-based and HTML-based DOMs
  - There is an XML-based HTML standard called XHTML, which is basically well-formed HTML.
- First you have the XML declaration
- And then you have the tree of tags.

Another example of XML

```
<?xml version="1.0"?>
<rss version="0.92">
  <channel>
    <title>Books I Love</title>
    <link>http://www.moreinfo.com/booksilove.html</link>
    <description>Gosh, I sure love books</description>
    <item>
      <title>The $100 Startup</title>
      <link>http://www.amazon.com/dp/0307951529</link>
    </item>
    <item>
      <title>The Art of Non-Conformity</title>
      <link>http://www.amazon.com/dp/0399536108</link>
    </item>
  </channel>
</rss>
```

Once you have XML, what can you do?

- $(data) gives you a wrapped set
- You can select nodes within the set with
  - .find("tagname")
  - .find("tagname:first") to get just the first
  - .find("#myid") to get an XML node by id
  - .find("tag1 tag2") to get tags inside tags
- And then get the text inside nodes using
  - .text()

For example, to grab and concatenate all the title elements in the document…

```
<!DOCTYPE html><html><head><script src="jquery-1.8.2.min.js"></script></head><body>
<div id="clickme" onclick="startAjax()">Click me</div>
<script>
function startAjax() {
  $("#clickme").text("Calling server");
  $.ajax({url:"somefile.xml",     success:callbackFunction, error:errorFunction}
}
function callbackFunction(data,info) {
  var titles = $(data).find("title");
  if (titles && titles.length)
    $("#clickme").text("result:"+titles.text());
  else
    errorFunction(data, "No titles");
}
function errorFunction(data,info) {
  $("#clickme").text("error occurred:"+info);
}
</script></body></html>
```

XML is kind of wordy, though

```
<?xml version="1.0"?>
<rss version="0.92">
  <channel>
    <title>Books I Love</title>
    <link>http://www.moreinfo.com/booksilove.html</link>
    <description>Gosh, I sure love books</description>
    <item>
      <title>The $100 Startup</title>
      <link>http://www.amazon.com/dp/0307951529</link>
    </item>
    <item>
      <title>The Art of Non-Conformity</title>
      <link>http://www.amazon.com/dp/0399536108</link>
    </item>
  </channel>
</rss>
```

What if we could just use JS notation?


```
{"version":"0.92",
  "channels":[ {
    "title":"Books I Love",
    "link":"http://www.moreinfo.com/booksilove.html",
    "description":"Gosh, I sure love books",
    "items":[ {
      "title":"The $100 Startup",
      "link":"http://www.amazon.com/dp/0307951529",
    },
    {
      "title":"The Art of Non-Conformity",
      "link":"http://www.amazon.com/dp/0399536108"
    }
  }]
}
```

Well, that is what we call JavaScript Object Notation (JSON)

- Essentially identical to declaring JS arrays
  - Either associative arrays or sequential arrays
  - Except that you have to be sure to quote the property names

```
{"name":"Jimmy", "age":54,
        "son":{"name":"Sam", "age":20}
}
```

A very simple web page and JSON

```
<!DOCTYPE html><html><head>
<script src="jquery-1.8.2.min.js"></script>
</head><body>
<div id="clickme" onclick="startAjax()">Click me</div>
<script>
function startAjax() {
  $("#clickme").text("Calling server");
  $.ajax({url:"somefile.json",
     success:callbackFunction,error:errorFunction,
     dataType: 'json'} /* request json -> JS object */
  );
}
function callbackFunction(data,info) {
  $("#clickme").text("result:"+data); /*data is JS object */
}
function errorFunction(data,info) {
  $("#clickme").text("error occurred:"+info);
}
</script></body></html>
```

somefile.json

```
{"name":"Jimmy", "age":54,
"son":{"name":"Sam", "age":20}
}
```

What if the server sends garbage?

- Be sure to provide $.ajax() an error handler.
- Be sure to check for null before using data.
- You probably should even use try/catch

```
function callbackFunction(data,info) {
  try {
    if (!data || !data.name)
      errorFunction(data, "no data");
    else
      $("#clickme").text("result:"+data.name);
  } catch (someException) {
    errorFunction(data, someException+"");
  }
}
```

How to send data to the server (or use GET for idempotent requests)

```
$.ajax({type:'POST',
  url:"blahblahblah.php",
  success:callbackFunction,error:errorFunction,
  dataType: 'json',
  data:{name1:value1,name2:value2}
});
```

Side comments about JSON…

- There are also libraries for reading and writing objects to/from JavaScript in other languages
  - For example, for writing Java Objects to JSON

- In JavaScript, you also can convert strings to/from objects even without hitting a server
  - var obj = JSON.parse(str)
  - var str = JSON.stringify(obj)