The DOM

Class 15

Cookies

- a cookie is a name-value pair
- created by the server
- labeled with server's domain name
- sent in HTTP headers from server to browser with a page
- stored by the browser
- sent back to server by browser
- two versions:
    - session cookies never "expire"; deleted when browser closes
    - persistent cookies have an explicit expiration date

Characteristics

- similar in concept to hidden input fields
- for convenience only
- easily blocked
- easily modified
- cannot harm browser (but can be used to track private actions)
- cannot harm server (unless stolen and used as part of attack)

Example

http://borax.truman.edu/315/c15/cookie.php

note the "lag" due to the request – server – response cycle

also note that we can navigate away and come back, and the cookie is still there

- cookies can be viewed in the browser
- easiest in Chrome; auto-updated
- menu → more tools → developer tools → application

Manage Users with Cookies

A Cookie-Powered Log-In

1. a page has some public information and a Login link
2. the link sends the browser to a login page
3. if successful, the user is redirected to home and can now see private info
4. if unsuccessful, the user must try again

`http://borax.truman.edu/315/c14/`

- this is not secure
- can be used for simple, low-stakes access control

A Document

- an HTML page in a browser is a model of a physical document

- a server emits HTML code
- the browser renders this onto the screen

- so far, the only control over how things appear on the screen is HTML + CSS

- once rendered, the page is static, fixed
- the only way to “change” the page is to go back to the server and get some different HTML
- but then it’s a totally new, different page, not the same page changed

Javascript

- like PHP, JS is a lightweight interpreted language
- JS can actually modify an existing page
- we use JS for event driven web page behavior
- JS acts in response to some event

User events

- mouse click
- key press
- mouse movement

System events

- page loads
- timers

PHP vs JS

- PHP runs on the server
- PHP runs once and is done
- JS is on the client (i.e., browser)
- JS is continuously available as long as the page is in the browser

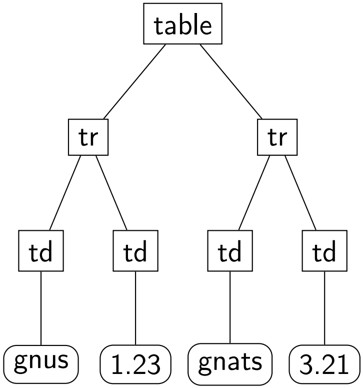
The DOM

to understand JS, you must embrace the DOM

- an API for HTML (and XML) documents
- defines the logical structure of a document
- defines the way a document is accessed and manipulated

Logical Organization

```
<table>
  <tr>
    <td>gnus</td>
    <td>1.23</td>
  </tr>
  <tr>
    <td>gnats</td>
    <td>3.21</td>
  </tr>
</table>
```



An Object Model

the DOM identifies

- the interfaces and objects used to represent and manipulate a document
- the semantics of these interfaces and objects, including both behavior and attributes
- the relationships and collaborations among these interfaces and objects

The Window

- the window object is the root of everything in a browser
- the window contains all other objects
- five main attributes (which are themselves objects)
- they are so important they are given global scope

history	contains the URLs visited by the user (within this browser window)
location	contains information about the current URL
navigator	contains information about the web browser currently in use
screen	information about the device screen that displays the window
document	the HTML document within the window on the screen

The Navigator Object

properties

- appname
- appversion
- cookieenabled
- language
- platform
- useragent

this is how a web server knows what kind of browser you're using

The Document

the DOM defines the relationship between HTML and the document

window.document (or just document, since it has global scope) contains nodes

- all HTML elements are element nodes
- all HTML attributes are attribute nodes
- the text inside an HTML element is a text node
- each comment is a comment node

Document Attributes

- body: the body element
- cookie: a string representation of all cookies supplied to this page
- referrer: the URL of the document the user was viewing before this one
- title: shown in the title bar
- URL: the complete URL of the current page

DOM Events

the DOM also defines events

- the DOM allows event handlers to be registered to an element
- event handlers are functions (e.g., JS functions)
- the function is executed when an event occurs (e.g., when a user clicks a button)

example events

- onclick
- onmousedown
- onkeypress
- onload
- onsubmit

Separation

remember from the very first lecture:

a fundamental concept of modern systems is the separation of:

- content
- appearance
- behavior

The Old Bad Way

90% of the examples of JS on the web are wrong

multiply example

this works, but is very wrong

Unobtrusive JS

- embedding JS behavior into HTML content is unacceptable
- instead we separate behavior and content with separated unobtrusive JS

Unobtrusive JS

Old way, everything in HTML:

```
<button id="doit_button" onclick="foobar();">Do It</button>
```

Modern separated way, the HTML:

```
<button id="doit_button">Do It</button>
```

in JS:

```
let doit_button = document.getElementById('doit');
doit_button.onclick = foobar;
```

JS Placement

- where does the JS code above go?
- when is it loaded?

- event registration must happen last, after page is fully loaded
- window.onload event happens upon completion of page load

multiply example 2

Two Common Errors

1. event vs property capitalization
   - event names are all lowercase: onload, onclick
   - property names are often camelCase: appName, scrollHeight
2. use of parens with functions
   - call and define a function with parentheses
   - refer to a function without parentheses

When Things Go Wrong

for all JS problems, use the console

- `"use strict";`
- misspelled.html

DOM Values

we got the value of the input control using

```
let multiplicand = document.getElementById('multiplicand').value;
```

an object's value is always a string.

when we multiplied:

```
let product = multiplicand * multiplier;
```

JS did automatic type conversion from string to number.

automatic type conversion is bad

add example

Strings to Numbers

two strategies:

```
let str1 = document.getElementById('num1').value;
```

- parseInt (or parseFloat)
  ```
  let input1 = parseInt(str1);
  ```
- Number constructor
  ```
  let input1 = Number(str1)
  ```

they differ when there are problems:

- `parseInt("24px")` gives 24
- `Number("24px")` gives NaN

- `parseInt("2e1")` gives 2
- `Number("2e1")` gives 20

The Safe Way

- whether you use parseInt() or Number()
- use regular expression first to make sure you know whether conversion will succeed