latex input: mmd-article-header Title: Notes on Web Programming Author: Ethan C. Petuchowski Base Header Level: 1 latex mode: memoir Keywords: Theory, Distributed Computing, Modern Technology CSS: http://fletcherpenney.net/css/document.css xhtml header: copyright: 2014 Ethan C. Petuchowski latex input: mmd-natbib-plain latex input: mmd-article-begin-doc latex footer: mmd-memoir-footer

Session

8/23/14

Name of the abstraction of keeping state during a user's interaction with your site.

1. There's no *specific* way this abstraction must be implemented, but one may use
    - Local/Session Storage database in the browser
    - Cookies
2. Sessions are useful for saving user preferences that apply across pages
    1. User logs in -> session is created
    2. Remembering how user has sorted a table, and their preferred date format, etc.

PUT

6/20/14

Find more in the *Networking and Network Programming Notes*

- Used for **modifying data on the server**
- **Idempotent** -- doing the same thing a second time has no effect

Web Cache

5/8/14

Caching Tutorial

Introduction

- **Sits between Web servers and clients, and watches requests come in, and *saves responses being returned***
    - **If it sees the same request come after, it *returns the saveed response***

- This **reduces latency** because it may be *physically closer* to the client then the server(s)
- It also **reduces network traffic** because *less wires are used* to relay the interaction
- The **browser cache** is the browser's own cache, which lives on your hard drive
    - It checks once a session for each resource
    - Makes the `back` button much faster
    - Makes loading pictures that are used on multiple pages throughout the site much faster
- A **proxy cache** is set up for your corporate network or by your ISP (possibly on the firewall)
    - Your requests, and those of 1,000s of people around you, are routed there unbenknownst to you, and y'all are sharing saved content amongst yourselves
- A **gateway cache** is set up by a Webmaster to increase performance of her site
    - Requests are routed to it by a load balancer, often by a Content Delivery Network (CDN) like *Akamai*

Downsides

- They can make it hard to see who's actually using the site
- They can serve **stale** content

Rules caches follow

- There is a response header that instructs the browser not to cache a resource, and they'll follow it
- Authenticated requests (e.g. `HTTPS`) are not cached
- Cached resources must have an experiation date
    - If it has expired, the origin server will *validate* whether the copy is still good
        - If it is *valid*, the server only has to say "use cache" instead of resending the whole resource
    - Unless the Net connection is down, in which case the expired resource will simple be served

Controlling caches

- Few browsers honor `<meta>` tags indicating expiration dates

You really have to use *HTTP headers*

- `Expires: Fri, 30 Oct 1998 14:19:41 GMT`
- `Cache-control: max-age=[seconds]`

- `Cache-control: no-cache` -- ask the origin server for validation every time
- etc.

Using ETags

- **ETags** are unique identifiers generate by the server and changed every time the resource does
    - For static content, your server has an easy time generating and moderating these automatically

Make your site cache-friendly

- The same content should always have the same URL
- Give things that won't change a large `max-age` header
- Note once more that SSL'd pages are not cacheable
- Generally only `GET` responses are cached
- Use the `Content-Length` response header
- Images and other large files are the most important
- "If you must know every time a page is accessed, select ONE small item on a page (or the page itself), and make it uncacheable, by giving it a suitable headers."
    - Don't actually do that though
- It's possible to facilitate shared-caching of pages that are password protected with `Cache-Control: public, no-cache`

Indexed DB

4/28/14

A document store that can be thought of as the NoSQL answer to Web SQL

- Reference blog post

> The gist of the story is this: in 2009 or so, native iOS and Android apps were starting to give the web a run for its money, and one area where the W3C recognized some room for improvement was in client-side storage. So Apple and Google hacked up the Web SQL Database API, which basically acknowledged that SQLite was great, mobile devs on iOS and Android loved it, and so both companies were happy to ship it in their browsers.

> However, Microsoft and (especially) Mozilla balked, countering that the SQL language is not really a standard, and having one implementation in WebKit didn’t meet the “independent implementations” requirement necessary to be considered a serious spec.

> So by 2010, Web SQL was abandoned in favor of IndexedDB, which is a document store that can be thought of as the NoSQL answer to Web SQL. It was designed by Nikunj Mehta at Oracle (of all places), and by 2014 every major browser, including IE 10 and Android 4.4, has shipped a version of IndexedDB, with Safari expected to join later this year.

> Now what we have instead [of Web SQL] is IndexedDB, which basically lets you store key/value pairs, where the values are JavaScript object literals and the keys can be one or more fields from within that object. It supports gets, puts, deletes, and iteration.

Post/Redirect/Get (PRG)

4/26/14

Quoted from Wikipedia

Implements bookmarks and the refresh button in a predictable way that prevents creating duplicate form submissions.

When a web form is submitted to a server through an HTTP `POST` request, a web user that attempts to refresh the server response in certain user agents can cause the contents of the original HTTP `POST` request to be resubmitted, possibly causing undesired results, such as a duplicate web purchase.

To avoid this problem, many web developers use the PRG pattern — instead of returning a web page directly, the `POST` operation returns a redirection command. The `HTTP 1.1` specification introduced the `HTTP 303 ("See other")` response code to ensure that in this situation, the web user's browser can safely refresh the server response without causing the initial HTTP `POST` request to be resubmitted.

A commonly used alternative to the PRG pattern is the use of a `nonce` to prevent duplicate form submissions.

User agents (such as browsers) store only the URI of an HTTP request as a bookmark. Because of this, an HTTP `POST` request that results in a response based on the body of the HTTP `POST` request cannot be bookmarked. By using the PRG pattern, the URI of the HTTP `GET` request can safely be bookmarked by a web user.

Padding vs. Margin

- **`padding` is the *inner* space of an element**
- **`margin` is the *outer* space of an element.**

4/28/14

Nice Tutorial

- Good illustration here
- Use margin to separate the block from things outside it
- Use padding to move the contents away from the edges of the block.

<div> vs <p>

4/13/14

These are both elements you can put text into

```
<div> text here </div>
<p> here too </p>
```

By default

- `<p>` tags have `margin: 17px` but no padding or border
- `<div>` tags have *no* margin, padding, or border.

File Reader API

Lets web applications asynchronously read the contents of files (or raw data buffers) stored on the user's computer.

Apparently this is pretty new and didn't used to be possible for security reasons.

Docs on MDN

4/12/14

Example based on *A Software Engineer Learns HTML5, JavaScript and jQuery*

First the **HTML**

```
<section id =" csvImport">
    <div>
        <label for="task">Import tasks from a CSV file</label>
        <input type="file" id="importFile" name="importFile"/>
    </div>
</section>
```

Now the **JavaScript** which is notified when the `<input>` `"file"` in the HTML is filled.

```
function useFile(event) {
  var reader = new FileReader() // this is the key
  reader.onload = function(evt) {
    // stuff to do on load
  }
  reader.onerror = function(evt) { /* etc. */ }
  reader.readAsText(event.target.files[0]) // 1st file they selecte
  // this is from the API in the docs above
  // or readAsArrayBuffer, readAsBinaryString, readAsDataUrl, etc.
}
```

Application Cache

Basically makes your web app available offline.

4/12/14

You create a manifest file with the following outline

```
# files to download and save for future use
CACHE MANIFEST
file.html
scripts/jquery-v.r.s.n.js
scripts/myapp.js

# files to use when offline
FALLBACK:
file.html backupPlan.html
etc. etc2.

# network addresses the browser is allowed to access
# to retrieve the files you specified
NETWORK:
/     # only the origin server
*     # *any* server
```

The files are only reloaded from your server when *something (anything [e.g. comment])* changes in your manifest file.

To tell the `tasks.html` file to download the manifest when it is loaded, add an attribute to its `html` tag:

```
<html lang="en" manifest="tasks.appcache">
```

Web Storage API

Offline DB for your app provided by the browser.

4/9/14

This is the simplest form of offline storage and supports name/value pairs where the name and value must both be strings. Although this is a relatively simple storage mechanism, and does come with a number of limitations, all major web browsers support it.

Basics

- Provides 15 MB per origin, compared to 4 kB available to cookies.
- Unlike cookies, data is not transmitted to server in every HTTP request, and the server can't write directly to it
- A hash-map data model, where keys and values are both strings
- If you pass in a non-string, `toString` is called on it, so if you want to turn it to/from JSON, use `JSON.stringify(obj)` and `JSON.parse(str)`

Local vs. Session Storage

- **localStorage** -- persists indefinitely, data is available to all pages from the same origin as the page they are saved from
- **sessionStorage** -- lasts for the duration of the session
  - intended to allow separate instances of the same web application to run in different windows without interfering with each other

API

```
sessionStorage.setItem('key', 'value');
sessionStorage.getItem('key');

sessionStorage.key2 = 'abcd' // shorthand for setItem
sessionStorage.key2          // shorthand for getItem

localStorage.setItem('key', JSON.stringify({name: 'value'}));
JSON.parse(localStorage.getItem('key')).name;
localStorage.removeItem('key');

// clear all data stored by this origin
localStorage.clear()
```

API

CRUD

TODO

- Create
- Read
- Update
- Delete

Cookies

4/9/14

Cameron, Dane (2013-10-30). A Software Engineer Learns HTML5, JavaScript and jQuery: A guide to standards-based web applications (p. 145). Cisdal Publishing. Kindle Edition:

- Traditionally the only mechanism for storing data on the client in a way that survives page refreshes was cookies.
- A cookie is a simple text based name/value pair that is stored on the client.
- The server can set a cookie on the client by including it in an HTTP response.
- Every-time the browser sends an HTTP request to a page on the domain from this point forward, the cookie (both its name and value) will be included in the request.
- Cookies can also be created and interacted with (with some restrictions) using JavaScript.

Cookies have significant limitations that make them inappropriate for storing large amounts of data:

1. They are included on each request to the server, so unless you want the entire task list to be included on each request, cookies are not a good solution.
2. The maximum size of each cookie is approximately 5 kilobytes.
3. The maximum number of cookies that a single domain can store is 20 (this can be browser dependent). This means that the total amount of cookie storage available to a domain is less than 100 kilobytes.

It has therefore been evident for some time that HTML needs some form of offline data storage if it is to allow for rich and dynamic web applications that can exist independent of a web server. Even applications that are continually connected to a web server can significantly improve user experiences if they could cache larger quantities of data on the browser.

Web storage API (see above): this is the simplest form of offline storage and supports name/value pairs where the name and value must both be strings. Although this is a relatively simple storage mechanism, and does come with a number of limitations, all major web browsers support it.

W3C

4/7/14

Wikipedia Page

- **World Wide Web Consortium**
- Main international **standards organization** for the World Wide Web
- Founded (in 1994) and currently **led by Tim Berners-Lee**
- HQ at MIT CSAIL
- Made up of 379 member organizations which maintain full-time staff developing WWW standards
  - This includes Apple, Facebook, Boeing, etc.
  - Their main criticism is that they're made up of multinational companies
- Also engages in education and outreach, develops software and serves as an open forum for discussion about the Web
- Tries to get all browser vendors to implement the consortium's chosen set of core components

WebRTC

Introduction

4/6/14

From MDN: WebRTC

- The RTC in WebRTC stands for Real-Time Communications
- Enables audio/video streaming and data sharing between browser clients (peers).
- Provides any browser with the ability to share application data and perform teleconferencing peer to peer, without the need to install plug-ins or third-party software.
- WebRTC components are accessed with JavaScript APIs
- Currently in development
  - Network Stream API -- an audio or video data stream
  - PeerConnection API -- allows two or more users to communicate browser-to-browser
  - DataChannel API -- enables communication of other types of data for
    - real-time gaming
    - text chat

        - file transfer
        - and so forth

Audio for the Web

AudioContext

4/6/14

Based on MDF Ref

- The AudioContext interface represents an audio-processing graph built from audio modules linked together, each represented by an AudioNode.
- You may want to use it to create the nodes you need
    - E.g. `AudioContext.createGain()` creates a `GainNode` (see below about `GainNodes`)

AudioNode

4/6/14

Quotes from MDF Ref, an excellent-looking documentation for this thing.

- "This is an experimental technology because this technology's specification has not stabilized."
- The `AudioNode` interface represents an audio-processing module, like an audio source (e.g. an HTML `<audio>` or `<video>` element), an audio destination, or an intermediate processing module (e.g. a filter like `BiquadFilterNode`, or a volume control like **`GainNode`**).
- An `AudioNode` has inputs and outputs. Each of them have a given amount of channels. An AudioNode with zero input and exactly one output is called a source node.
- The processing varies from one `AudioNode` to another but, in general, a node reads its inputs, does some audio-related processing, and generates new values on its output.
- Different nodes can be linked together to build a processing graph. Such a graph is contained in an `AudioContext`. Each `AudioNode` participates in exactly one such context.
- An `AudioNode` can be target of events, therefore it implements the `EventTarget` interface.

Note these aspects

- `AudioNode.context` *(read only attribute)* -- links to the associated `AudioContext` (see above), that is the object representing the processing graph

the node is participating in.
- `AudioNode.connect(AudioNode)` *(method)* -- connect one output of this node to one input of another node.

GainNode

4/6/14

From MDF Ref

The GainNode interface represents a change in volume. It is an AudioNode, that is an audio-processing module, that causes a given gain to be applied to the input data before its propagation to the output. A GainNode always has exactly one input and one output, both with the same amount of channels.

The gain is a unitless value, eventually changing with time, that is multiplied to each corresponding sample of all inputs channels. If modified, the new gain is applied using a de-zippering algorithm in order to prevent unaesthetic 'clicks' from appearing in the resulting audio.

WebSockets

3/21/14

Wikipedia

- **WebSocket** -- a *protocol* providing *full-duplex* communications channels over a *single TCP connection*.
- **Full-duplex** -- data can go both directions across the connection at the same time
- First standardized in 2011
- Facilitates live content and real-time games
- Enables an ongoing back and forth between browser and server
- Happens over TCP port 80
- Server or client can easily close it.
- Huge performance and architecture advantages over AJAX or Long Poll
- Uses URI schems `ws:` and `wss:` for unencrypted and encrypted connections respectively
- Handshake looks like a HTTP `GET` request/response sequence, but it's actually an a protocol-upgrade request.

Client request:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Server response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

WebSocket.org

- "HTML5 Web Sockets" provides a dramatic improvement from the old, convoluted hacks that are used to simulate a full-duplex connection in a browser
- Older techniques for achieving this included *long-polling* and *streaming*, often wrapped into **Comet**-based "pushes"
- Simply put, HTTP wasn't designed for real-time, full-duplex communication, it is a half-duplex model
- WebSockets are in the HTML5 specification
- Once a WS connection is established by handshake (see above), both text and binary frames can be sent full-duplex, in either direction at the same time.
- The data is minimally framed with just two bytes.

LostTechies.com

- Because the connection is persistent, the server can now initiate communication with the browser. The server can send alerts, updates, notifications.
- This adds a whole new dimension to the types of applications that can be constructed.
- WebSockets are a boon to the *Internet of Things* (good point...)
- "In point of fact, the entire AJAX protocol could be built using Websockets technology. This makes Websockets a literal superset of AJAX."

AJAX

3/21/14

Asynchronous JavaScript and XML

Wikipedia

- Has a specific field for sending arbitrary text, and another for XML
- Nowadays, people just use the text one for sending JSON
- Create connection to server on each request, sends request (with possible extra data), and gets response from server. Then connection closes.
- It is *single request > response for each AJAX call*.
- JavaScript and the `XMLHttpRequest` object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.
- Term first appeared in 2005

Downsides:

- *In pre-HTML5 browsers*, hitting the back buttun didn't have the intended effect
- *In pre-HTML5 browsers*, bookmarking didn't have the intended effect
- If user's connection is slow, it might do things at the wrong time
- Most web-crawlers don't execute JavaScript
- It makes it harder to make web-page accessible to screen-reader technologies
- It leads to trickier callback-ridden code that is harder to maintain, debug, and test

Example using jQuery:

```
$.get('send-ajax-data.php', function(data) {
    alert(data);
});
```

Port

3/21/14

Wikipedia

- The purpose of ports is to uniquely identify different applications or processes running on a single computer and thereby enable them to share a single physical connection to a packet-switched network like the Internet.
- Data packets are routed across the network to a specific destination IP address, and then, upon reaching the destination computer, are further routed to the specific process bound to the destination port number.
- Used by *Transport Layer* protocols, such as *TCP* and *UDP*
- 16-bit *port number*
- Some example **Well-known ports**:

    - 22 -- SSH
    - 53 -- DNS
    - 80 -- HTTP
    - 194 -- IRC
    - 465 -- SMTP Secure

Socket

- IP Address + Port
- A single bidirectional connection between two network applications (can even be on the same computer)

Socket.IO

3/21/14

- "Realtime application framework for Node.JS, with HTML5 WebSockets and cross-browser fallbacks support."
- "Aims to make realtime apps possible in every browser and mobile device, blurring the differences between the different transport mechanisms."

Comet

3/21/14

- Old-school way of doing full-duplex over HTTP, replaced by WebSockets
- A long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it.
- Comet is an umbrella term, encompassing multiple techniques for achieving this interaction.
- All these methods rely on features included by default in browsers, such as JavaScript
- **Long-polling** -- the browser sends a request to the server and the server keeps the request open for a set period. If a notification is received within that period, a response containing the message is sent to the client. If a notification is not received within the set time period, the server sends a response to terminate the open request.
- **Streaming** -- the browser sends a complete request, but the server sends and maintains an open response that is continuously updated and kept open indefinitely (or for a set period of time). The response is then updated whenever a message is ready to be sent, but the server never signals to complete the response, thus keeping the connection open to deliver future messages.

Backbone.js

Data models for JavaScript?

Angular.js

RabbitMQ

Scalable, asynchronous messenging

Vocab

- **Origin** -- combination of *protocol*, *hostname*, and *port number*