Web and Mobile Technologies

A Quick Overview

History

- Web was invented by Tim Berners-Lee working at CERN
  - Combined ideas of hyper-text, TCP and domain name system
  - Intent was to share information amongst researchers
- First web-site went online:

  December 20, 1990

HTTP

- HTTP - Hypertext Transfer Protocol
- Specifies how to fetch and send documents
- Popular servers: Apache, Nginx, IIS
- Two base concepts:
  - GET -- retrieve a document
  - POST -- push a document

Parts of a URL

- URI (Uniform Resource Identifier) is a descriptor of the location of content or a resource:

scheme:[//autority]path[?query][#fragment]

- URL (Uniform Resource Locator) is a type of URI used on the web (and elsewhere)

http://fred@example.com:80/place/?q=a&x=3;#subtitle

GET vs POST

- Both:
  - Use URL
  - Send headers and possibly query strings to server
  - Expect a response document
- Post also includes document content sent to the server

Typical GET

- Response body:
  - Document with content
  - Commonly HTML
  - Content may point at other content to fetch: images, style sheets, Javascript...

Typical GET

- GET: `https://slashdot.org`
- Request headers:

Host:	slashdot.org
User-Agent:	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:75.0) Gecko/20100101 Firefox/75.0
Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language:	en-US,en;q=0.5
Accept-Encoding:	gzip, deflate, br
Connection:	keep-alive
Upgrade-Insecure-Requests:	1
Cache-Control:	max-age=0

Typical GET

- **Response headers:**

	HTTP/2 200 OK
server:	nginx/1.14.0 (Ubuntu)
date:	Tue, 14 Jan 2019 19:31:05 GMT
content-type:	text/html; charset=utf-8
slash_log_data:	shtml
cache-control:	no-cache
pragma:	no-cache
x-xrds-location:	https://slashdot.org/slashdot.xrds
strict-transport-security:	max-age=31536000
X-Firefox-Spdy:	h2

Typical POST

- POST: `https://example.com`
- Request headers:

	`POST /test HTTP/1.1`
`Host:`	`example.com`
`Content-Type:`	`application/x-www-form-urlencoded`
`Content-Length:`	`27`

- Request body:

```
field1=value1&field2=value2
```

Response Codes

- Response from server includes a code indicating success/ failure
- Common codes:
  - 200 -- ok
  - 301 -- page moved permanently
  - 302 -- page moved temporarily (YMMV)
  - 401 -- unauthorized
  - 403 -- Forbidden
  - 404 -- Not found
  - 500 -- Internal Server Error
  - 502 -- Bad gateway

Response Bodies

- A response from a server contains a code, some headers and some content -- the "response body"
- There are multiple variations of response body:
  - Static
  - Dynamic
    - Server Side scripting
    - Containing client-side scripting
    - Single Page Applications
  - REST (SOAP, others)

Static Sites

- Straight HTML
- May include images, CSS, Javascript
- No server side work, server just responds with contents of directory that maps to the URL
- Static site generators use templates to create content
- Fast
- Secure

Dynamic Sites

- Server runs software to generate pages
  - Started with CGI protocol
    - e.g. Perl script that “prints” HTML
- Languages such as PHP are based on the page with sections of code interleaved
- Includes specialty servers that are language specific: Apache Tomcat, WebLogic
- WSGI, ASGI, Rack, PSGI, JSGI specify how to map a language to a server, available on multiple servers

Client Side Scripting

- Browser downloads code to run on the requesting machine
- Can be any language that the browser, or its plugins, support
- Usually Javascript (now)
- Client-side logic can be in either a static or dynamic web site
- AJAX is a on-demand call from browser to get more content for the page, could be snippet of HTML or data

Single Page Applications

- Extreme version of a dynamic web site
- All UI and interface with the user is done on the client side
- After initial load, all communication with the server is via REST or similar and only contains data
- Popular frameworks: AngularJS, React, Ember.js, ExtJS, Vue.js

REST (and others)

- REST (REpresentational State Transfer)
  - Transfer of data over the HTTP protocol
  - Path portion of URL indicates the data in question
  - Typically uses JSON or XML for data
  - Uses GET, POST to retrieve/send
  - Also has PUT, PATCH, DELETE
- SOAP, others similar mechanisms

HTTP, HTTPS, Others

- HTTP is unencrypted, defaults to port 80
  - Multiple versions of protocol exist
  - HTTP/3 in draft now, supported by some browsers
- HTTPS is encrypted, defaults to port 443
  - HTTP with communication encrypted using TLS
- SPDY: deprecated protocol invented by Google, primary driver to get world to move to HTTP/2
- QUIC: another Google protocol, mostly used by Chrome to connect to Google servers

Stateless

- HTTP is a stateless protocol
- Every call is independent of every other call
- Series of hacks have been built on top of it to maintain state (e.g. user login status)
  - Cookies:
    - Header sent back and forth between browser and server
  - Session management:
    - Token created on server maps to server side state information, token sent back and forth either via HTTP parameters or as a Cookie
- "Hacks" are a security risk -- never trust what the user sends

Web Proxy

- Server that forwards HTTP requests
- Used to control traffic flow over networks
  - Limit destinations
  - Load balancing
  - Throttling

Web Sockets

- Bi-directional, full-duplex communication protocol
- Distinct from HTTP
- Allows for: simultaneous edit of documents from multiple users, head-to-head games, etc.

Mobile Technologies

Platforms

- Android:
  - Google
  - Modified version of Linux
  - Primarily Java based
  - Google Play is public store
- iOS:
  - Apple
  - Primarily Swift or Objective-C
  - iTunes is public store

Application Types

- Native: specific to platform
  - Multi-platform means developing twice
- Mixed:
  - Develop underlying libraries in common language
  - Use native language for UI
- Web Only:
  - User must use browser on device to go web-site, same tech as web
- Browser Application:
  - Web-application wrapped in a custom browser
  - Access to device features such as camera and location
  - Mostly cross platform
  - Never quite “feels” native

Application Types

- Except for “web only”:
  - Compiled languages
  - Signed content
  - Only available in a online store
    - Corner cases: “jail break”, F-Droid
- Far more restrictive than desktop development
  - Applications generally can’t see each other or effect each other

Server Side

- Most applications use web technologies on the server side
  - AJAX
  - REST, SOAP
- Pairs well with Single Page Applications