Web Programming

CSCI 311

Web-Aware Software

Any software that is capable of communication over a network (especially the internet).

For example:

- Skype
- Word (you can download resources – themes, clipart, etc.)
- Call of Duty
- World of Warcraft

For the purpose of this course, we label these “web-aware” – not “web applications.”

Web Applications – in this course

We label software a “web application” if it is designed to run in a web browser – specifically, without plug-ins.

A complete web app consists of:

- A front-end running in a browser (i.e. Mozilla, Chrome, Internet Explorer, Safari, Opera, etc.)
- A back-end running on a server-machine – usually remote
- The two components must interact

The Web

The Web – internet – is a global system of interconnected computer networks.

Internet traffic consists of client/server communication, via a set of standardized protocols.

History of the Web

(Web Technology Timeline)

Through the years, many web technologies were invented and implemented – some became standardized, then dropped.

Some technologies persist – though in a different form.

In five years – this course would look very different.

The Web Today

One major tendency in the web today is the standardization of client-side technologies.

These are, in turn, implemented in browsers, which may differ, but all adhere to the base-level standard.

The current client-side standard set is established and maintained by the W3C, and is referred to using the umbrella term – **HTML5**.

While not all browser fully support the standard, by 2014 – when HTML5 is finalized – they all should.

The HTML5 Standard

Includes:

- HTML 5 – for semantic page structuring
- CSS 3 – for page styling
- JavaScript – for programming on the web
- Web Workers – for program performance
- Local Storage – for caching data on client machines
- Canvas – for real-time graphics
- (Native Audio and Video) – partially implemented

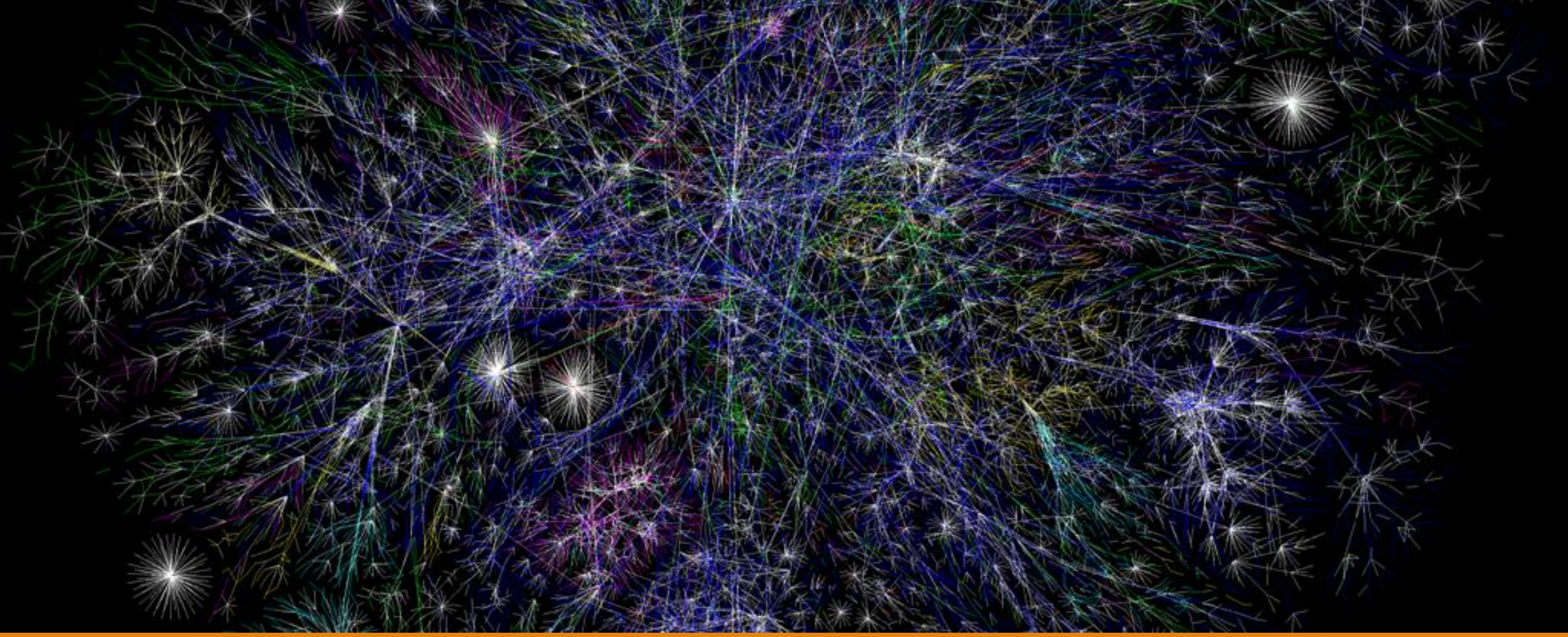
Browser Support for HTML5

Modern Web Applications

Using the robust infrastructure of the web, and combined with the newly emerging HTML5 client-side standard – we may develop software that:

- Has low latency/high enjoyment user experience which rivals that of traditional desktop software
- Is web-enabled
- Is device-portable/agnostic
- Has few version control issues
- Has a low barrier of entry for users – because there is no need to install software beyond the browser

Browser Quest – A showcase of Canvas, HTML5 Audio and WebSockets by Mozilla Foundation



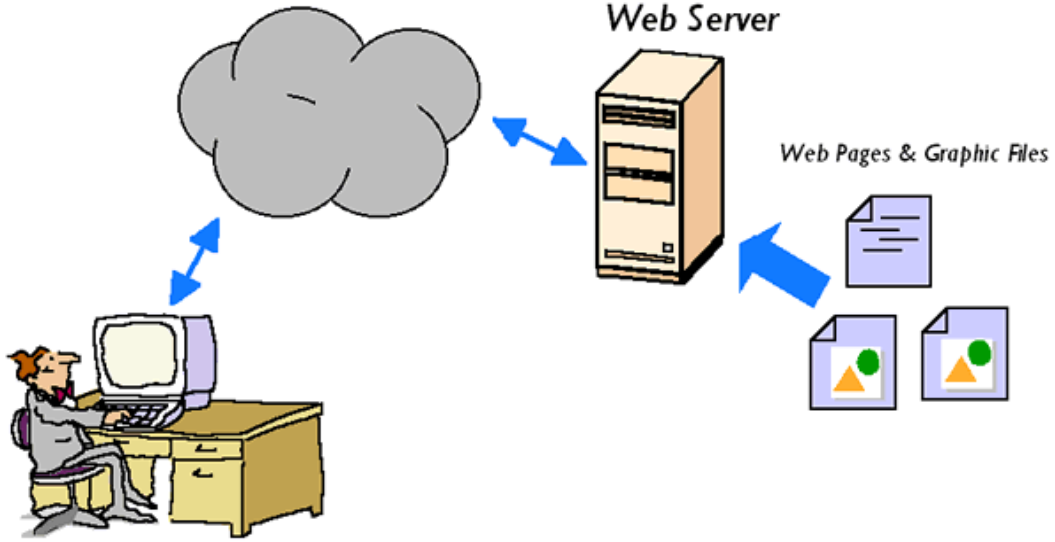
How the Web Works (in very simple terms)

or – Internet, How? How Internet!

The Client/Server Model

The “web” consists of billions of individual machines.

When one machine wants to communicate with another, it must send a communication request – the requesting machine is a client, the responding is a server.



TCP/IP

Individual machines are identified by their

.

The IP (protocol) is used to route a message from one machine to another.

TCP is a transfer protocol which ensures the message is delivered unharmed – this is done using a concept known as a checksum.

Many communication protocols may piggyback on TCP – we are interested in HTTP.

HTTP (hypertext transfer protocol)

HTTP is a standardized protocol for communication – as opposed to transfer – of clients and servers on the web.

HTTP – request

An HTTP client-sent message – a request – consists of:

- A method name
- A destination indicator – an address (IP or URL) and a resource name (a file)
- Optional parameters – collected from a user form

Using the DNS servers and IP address, the destination machine is identified.

The message is than delivered to the destination machine – which resolves the request accordingly.

HTTP – request: GET

- A method name
- A destination indicator – an address (IP or URL) and a resource name (a file)
- Optional parameters – collected from a user form

```
GET
/ServiceLogin?service=mail&passive=true&rm
=false

host:accounts.google.com
user-agent:Mozilla/5.0 (Windows NT 6.1)
AppleWebKit/537.11 (KHTML, like Gecko)
Chrome/23.0.1271.97 Safari/537.11
version:HTTP/1.1
accept:text/html,application/xhtml+xml,app
lication/xml;q=0.9,*/*;q=0.8
accept-charset:ISO-8859-1,utf-
8;q=0.7,*;q=0.3
accept-encoding:gzip,deflate,sdch
accept-language:en-US,en;q=0.8
referer:https://signup.netflix.com/
```

HTTP – request: GET

- A method name
- A destination indicator – an address (IP or URL) and a resource name (a file)
- Optional parameters – collected from a user form

```
GET
/ServiceLogin?service=mail&passive=true&rm
=false

host:accounts.google.com
user-agent:Mozilla/5.0 (Windows NT 6.1)
AppleWebKit/537.11 (KHTML, like Gecko)
Chrome/23.0.1271.97 Safari/537.11
version:HTTP/1.1
accept:text/html,application/xhtml+xml,app
lication/xml;q=0.9,*/*;q=0.8
accept-charset:ISO-8859-1,utf-
8;q=0.7,*;q=0.3
accept-encoding:gzip,deflate,sdch
accept-language:en-US,en;q=0.8
referer:https://signup.netflix.com/
```

HTTP – request: GET

- A method name
- A destination indicator – an address (IP or URL) and a resource name (a file)
- Optional parameters – collected from a user form

```
GET
/ServiceLogin?service=mail&passive=true&rm
=false

host:accounts.google.com
user-agent:Mozilla/5.0 (Windows NT 6.1)
AppleWebKit/537.11 (KHTML, like Gecko)
Chrome/23.0.1271.97 Safari/537.11
version:HTTP/1.1
accept:text/html,application/xhtml+xml,app
lication/xml;q=0.9,*/*;q=0.8
accept-charset:ISO-8859-1,utf-
8;q=0.7,*;q=0.3
accept-encoding:gzip,deflate,sdch
accept-language:en-US,en;q=0.8
referer:https://signup.netflix.com/
```

HTTP – request: POST

POST = GET ++

- A method name
- A destination indicator – an address (IP or URL) and a resource name (a file)
- Optional parameters – collected from a user form

```
POST
/mail/ca/u/0/?ui=2&ik=107df84a1e&rid=mail%3Ai.680f.0.
1&view=cv&th=13c04134dc1284ab&th=13c0255dc95e706
host:mail.google.com
scheme:https
version:HTTP/1.1
accept:*/*
…
ui:2
ik:107df84a1e
rid:mail:i.680f.0.1
view:cv
prf:1
nsc:1
mb:0
search:inbox
```

HTTP – request: POST

POST = GET ++

- A method name
- A destination indicator – an address (IP or URL) and a resource name (a file)
- Optional parameters – collected from a user form

```
POST
/mail/ca/u/0/?ui=2&ik=107df84a1e&rid=mail%3Ai.680f.0.
1&view=cv&th=13c04134dc1284ab&th=13c0255dc95e706
host:mail.google.com
scheme:https
version:HTTP/1.1
accept:*/*
…
ui:2
ik:107df84a1e
rid:mail:i.680f.0.1
view:cv
prf:1
nsc:1
mb:0
search:inbox
```

HTTP – request: POST

POST = GET ++

- A method name
- A destination indicator – an address (IP or URL) and a resource name (a file)
- Optional parameters – collected from a user form

```
POST
/mail/ca/u/0/?ui=2&ik=107df84a1e&rid=mail%3Ai.680f.0.
1&view=cv&th=13c04134dc1284ab&th=13c0255dc95e706
host:mail.google.com
scheme:https
version:HTTP/1.1
accept:*/*
…
ui:2
ik:107df84a1e
rid:mail:i.680f.0.1
view:cv
prf:1
nsc:1
mb:0
search:inbox …
```

HTTP – response

An HTTP server-sent response includes:

- A status code – indicating the success or failure of a request
- Content type
- The requested resource (if the request was successful)

The communication then terminates.

HTTP – response

An HTTP server sent response includes:

- A status code – indicating the success or failure of a request
- Content type
- The requested resource (if the request was successful)

Response Header

cache-control:no-cache, no-store, max-age=0, must-revalidate
content-encoding:gzip
content-length:10445
content-type:text/javascript; charset=UTF-8
date:Fri, 04 Jan 2013 13:20:42 GMT
expires:Fri, 01 Jan 1990 00:00:00 GMT
pragma:no-cache
server:GSE
status:200 OK
version:HTTP/1.1
x-content-type-options:nosniff
x-frame-options:SAMEORIGIN
x-xss-protection:1; mode=block

HTTP – response

An HTTP server sent response includes:

- A status code – indicating the success or failure of a request
- Content type
- The requested resource (if the request was successful)

Response Header

cache-control:no-cache, no-store, max-age=0, must-revalidate
content-encoding:gzip
content-length:10445
content-type:text/javascript; charset=UTF-8
date:Fri, 04 Jan 2013 13:20:42 GMT
expires:Fri, 01 Jan 1990 00:00:00 GMT
pragma:no-cache
server:GSE
status:200 OK
version:HTTP/1.1
x-content-type-options:nosniff
x-frame-options:SAMEORIGIN
x-xss-protection:1; mode=block

HTTP communication

All data in the request-response messages except for the content (the requested resource) is *metadata* – called the http header.

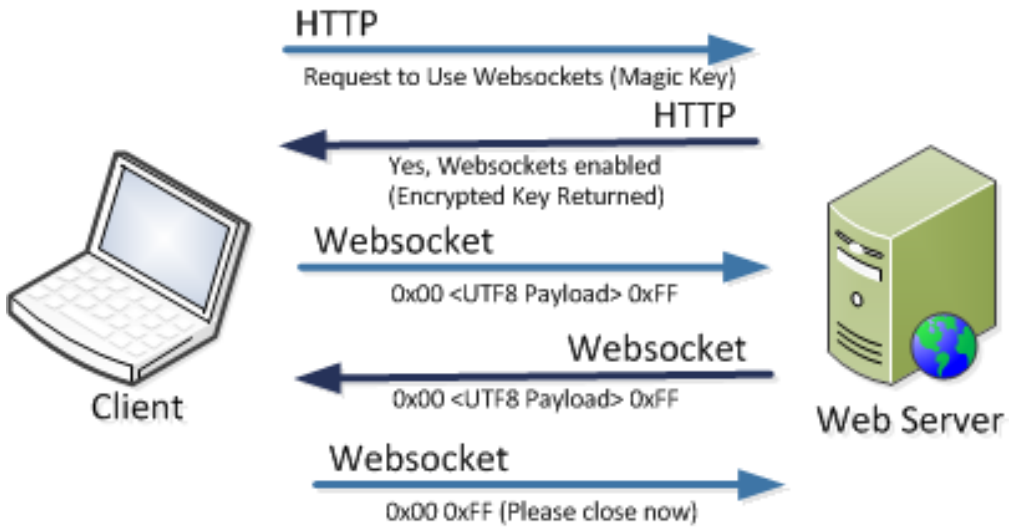
This helps the software on both sides to understand the message and act appropriately (but it also adds overhead).

WebSocket

WebSocket is a communication protocol that piggybacks on top of HTTP.

A WebSocket session begins as an HTTP request, which tells the server to establish and keep an open two-sided connection with the specific client.

Additionally, WebSocket communications lack most header information, such as resource type and so on. This means that the client and server must know what to anticipate in a message; but it also significantly reduces bandwidth overhead.



Course Mechanics

Prerequisites

Prior programming experience with any language

HTML and CSS basics

Topics covered

- JavaScript – the language of the web
- HTML5 and CSS 3 – GUI
- Drawing on Canvas
- HTTP servers:
  - PHP
  - Server pages
  - NodeJS
- WebSockets
- Storing data:
  - XML and JSON file format
  - SQLite relational database
  - MongoDB NoSQL database
- Security and scalability

(Web Technology Timeline)

Topics covered in-depth

- JavaScript – the language of the web
- HTML5 and CSS 3 – GUI
- Drawing on Canvas
- HTTP servers:
  - PHP
  - Server pages
  - NodeJS
- WebSockets
- Storing data:
  - XML and JSON file format
  - SQLite relational database
  - MongoDB NoSQL database
- Security and scalability

(Web Technology Timeline)

Deliverables

4 programming assignments 50%:

- Math games with JavaScript
- Board games with JavaScript, DOM, canvas
- Multi-player chat game
- Players database

Final project 30% (0 or 1 collaborator)

Lab assignments 20%

In-class quizzes 15%

Project

By the end you should be able to implement project that is of interest to **you**

Our goal is to build an application or a prototype that clients around the globe could access (and we become obscenely reach)

Submissions

All deliverables will be submitted electronically

Assignments are due at 11:59 p.m. on the due date - check website for final due dates

Late Work Policy: Accepted up to a week after the due date with a 3% penalty per day

Marking by Demonstrations (???)

Work ethics

"The work you submit must be your own, done without participation by others. It is an academic offense to hand in anything written by someone else without acknowledgement."

You are not helping your friend when you *give* him or her a copy of your work

You are hurting your friend when you *ask* him or her to give you a copy of their work

Lab Environment

Browser: Google Chromium

Text Editor: Emacs

Nodejs

SQLite database

MongoDB

Deliver and upload to Amazon EC-2 server

Course Resources

Appendix

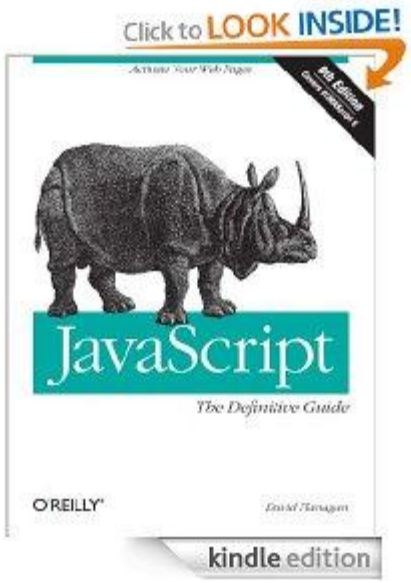
Links

- Mozilla Developer Network
- An Implementation of the HTML5 Outline Algorithm
- HTML Rocks – Google HTML5 Showcase
- Browser Support for HTML5
- Browser Quest – A showcase of Canvas , HTML5 Audio and WebSockets by Mozilla Foundation

Good JavaScript reference book

JavaScript: The Definitive guide [Kindle Edition], by David Flanagan

Also can read and download this book through free trial on Safari books online



Optional textbooks

HTML, XHTML, and CSS: Your visual blueprint for designing effective Web pages, by Rob Haddleston

PHP & MySQL: Your visual blueprint for creating dynamic, database-driven Web sites, by Janet Valade

JavaScript: Your Visual Blueprint for Building Dynamic Web Pages, 2nd Edition by Eric Pascarello