

Lecture

CS571 - Course Introduction

CS571: Web Technologies

- Instructor: Prof. Marco Papa
- Office hours: on **Zoom**:
 - **Mon day 5:00PM-6:00PM PDT**
 - **Meeting ID: 924 6746 7426**
 - **Passcode: 629843**
 - **iCalendar (.ics) available on D2L**
- E-mail: papa@usc.edu
- Zoom Lectures stored on DEN Desire2Learn (D2L)
- 24/7 Q&A access: Piazza, **3 min. average response time**
- Producer access: daily (6 days / week)
- Instructor access: weekly
- Quick way to ask a “personal” question: Private message to Instructors on Piazza

General Information

- Lectures:
 - **Session 30378D: Tue, Thu 5:30PM – 7:20PM (SGM 124)**
 - **Session 30016D: Tue, Thu 5:30PM – 7:20PM (DEN)**
 - **Session 30338D: Tue, Thu 5:30PM – 7:20PM (ONLINE)**
- Producers: Zoom office hours on course website and Piazza
- Course website: <https://www.csci571.com>
- DEN's D2L for Zoom Lectures: <https://courses.uscden.net>
- Assignments – yes, attendance – up to you
- Two **Exams**, one at half course and one towards the end
- Exams auto-graded using **D2L Quiz Tool**
- Mobile Application – final “comprehensive” project live demo
- Web storage: **GitHub Pages**
- Cloud Storage: **Google Cloud**

General Rules

NO FACULTY D-CLEARANCE

Unless you are a “superstar” undergrad ☺

NO COURSE OVERLAP

Unless you agree it is “your” problem

Fixed dates / times:

Exam #1: February 22, 6PM PDT

Exam #2: April 26, 6PM PDT

Final Mobile Project: May 5

Exams electronically graded using D2L Quiz Tool

Final Mobile project/presentation Zoom video uploaded to D2L

General In Person Class Rules

NO CONTACT

Do not approach Professor at lectern
BEFORE, DURING or AFTER lecture

Wear N95, K95 or surgical masks

Post questions on Zoom

Grader will answer during lecture or table for answering in Piazza

Benefits of Prof. Papa CSCI571

- Top 20 students in each section offered available Grader/Producer positions at end of semester. **BY INVITATION ONLY**
- Students that are invited to **MS Honors Program** and scored an A will be provided with **Recommendation Letter** (14 “recommended” students already awarded MS Honors)
- Students that score an “A”, can request “proof of skill” for H1B applications.
- Prof. Papa will act as “reference” in job applications

Course Objectives

- This course focuses on the phenomenon known as the World Wide Web
- Core technologies are:
 - HyperText Markup Language (HTML) and Cascading Style Sheets (CSS)
 - HyperText Transfer Protocol (HTTP)
 - Web servers, their configuration and performance properties
 - Server-Side programming using JavaScript and Python
 - Client-side programming using JavaScript and JS Frameworks
 - Ajax Development Style
- Newer Technologies of Interest
 - Responsive Website Design (Bootstrap, etc.)
 - JS Frameworks (Angular, React and Node.js)
 - Web Services (REST)
 - Web security, TOR, Dark web
 - Native Mobile frameworks (Java / Android and Swift / iOS)
 - React (native)
 - Cloud computing (AWS, GCP, Azure)
 - Serverless Applications, Containers, Docker
 - AWS Lambda, Google Cloud Functions, Azure Functions

Software and Storage

- **Student Disk space on GitHub Pages:**
 - <https://pages.github.com/>
 - **GitHub Student Developer Pack:**
<https://education.github.com/pack>
 - Allows GitHub Pages with “private” repository
 - 1GB of free web space
 - Used for homework 2 and 3 and Table of Exercises
- **Website / Web Services in the cloud**
 - Amazon’s Elastic Compute Cloud (AWS)
 - **Google Cloud Platform (GCP) Preferred!!!**
 - Microsoft Azure
 - AWS Lambda, Google Cloud Functions, Azure Functions
 - Serverless.com
 - Node.js
 - Docker

Other Issues

- Piazza class news group
 - Activate your membership by self-joining at:
piazza.com/usc/spring2022/csci571
 - Class Access Code: **lafc3252usc**
- Academic Integrity Policy
 - Do NOT submit the same program; you can discuss the project with fellow students, but do not develop code with other students; do not download code online; do not post code online; we use MOSS to check for plagiarism (similar code). We scan all the exams. See "Academic Integrity Policy".
- Downloading course slides and software
 - Class slides access. Username: **csci571**, password: **notes1**
 - All software and installation instructions can be downloaded from the class website.

Student Evaluations

- Comments:
 - "Amazing assignments. Learnt a lot on the course."
 - "Projects seemed similar to an actual client for web development would ask for."
 - "Even though the assignments were hard I learnt a lot from them."
 - "It is not a fair game for beginners. I've spent almost 3 weeks to do a homework, and I still can not finish it on time."
 - "Course projects are impressive!"
 - "This class has posted assignments easily x10 times larger than other classes."
 - "HW8 and HW9 take *forever*."
 - "The homework assignments are so difficult."
 - "I had to do so much googling on my own to learn about concepts used in the homework assignments."
 - "Tough class with a lot of valuable assignments."
 - "Massive assignments."
 - "The workload of this course is too much, especially the last two homework."

Academic Integrity Violations

- Spring 2018 violations (16):
 - Sanctioned: 16
 - Appeals to Engineering Panel Review: 8, no changes
 - Appeals to Engineering Dean: 2, no changes
 - **F in course:** 12
 - 0 + full letter grade reduction: 4 (C-, C-, C-, B-)
- Fall 2018 violations (0):
 - None!
- Spring 2019 violations (4):
 - **F in course:** 1
 - 0 + full letter grade reduction: 1 (B-)
 - C in the course: 2
- Fall 2019 violations (6):
 - **F in course:** 1
 - 0 + full letter grade reduction: 5 (B-, C)
- Spring 2020 violations (4):
 - 0 + full letter grade reduction: 1 (A-, C)
- Spring 2021 violations (2)
 - Full letter grade reduction

Piazza

general advice
retrieved from →
last semester

The screenshot shows a web browser window with the title bar "File Edit View History Bookmarks Tools Help" and the address bar "CSCI 571 (20 unread) https://piazza.com/class/hxnr3zbzv156m". The main content area displays the Piazza course interface for "CSCI 571".

Class at a Glance (Updated 1 minute ago)

- 20 unread posts**
- no unanswered questions**
- no unresolved followups**
- 21 total posts
- 172 total contributions
- 0 instructors' responses
- 0 students' responses
- n/a avg. response time

Student Enrollment ..out of 350 (estimated)
13 enrolled

Share Your Class
Professors appreciate Piazza best when they see how it is being used.
Allow colleagues to view your class through a demo link - a restricted, read only version of your class where all students' names are anonymized and all student information hidden.

https://piazza.com/demo_login?nid=hxnr3zbzv156m&auth=ff4abc4
Opening this link in the same browser will log you out as horowitz@usc.edu

Recent Posts

- Instr Tip 1: Add link to a question (2 Mon)
- Instr Homework #3: Practice Writing (41 Mon)
- Instr Homework #2: Creating Your... (44 Mon)
- Instr Homework #1: Join the Clas... (6 Mon)
- Instr Welcome to CSCI 571! (5 Mon)
- WEEK 7/13 - 7/19 (7/15/14)
 - Private Introduce Piazza to your stud... (1)
 - Private Get familiar with Piazza (1)
 - Private Tips & Tricks for a success... (1)
- Welcome to Piazza! (7/15/14)
 - Piazza is a Q&A platform designed to get you great answers from classmates and instructors fast. We've put together thi

Who am I?

- PhD in CS from USC, class of '88
- PhD Advisor: Prof. Ellis Horowitz
- Initial career: MS Windows, Commodore Amiga developer
- System Architect -> Team Lead -> IT Project Manager -> VP Engineering
- CTO (Chief Technology Officer) at Luckman Interactive and CareerBuilder
- Faculty at USC since 2003
- Chief Technologist at LASC (Los Angeles Superior Court) since 2002
- O365 SharePoint Project Manager since 2014
- Active member of LA CTO Forum (invitation only group of Silicon Beach CTOs)
- Season ticket holder of USC Football and  LAFC (Los Angeles Football Club) and  supporter

Characterizing Web Content

There are very few studies that examine the types of content on the web, however . . .

(From IEEE Spectrum, Jan. 2004, pp. 75) :

- Claim: 30% of the web is porn
- Claim: 30% of the web is duplicate information
- 50,000,000 pages are either new or changed each day
- 65% of the web pages are in English

(From Personal Computer World, Optenet, Sep. 2008) :

- Claim: 35% of the web is porn, 11% is e-commerce
- <http://www.optenet.com/en-us/new.asp?id=162>

(From Forbes, Sept. 2011) :

- Claim: 4% is porn, 13% are porn Web Searches
- <http://www.forbes.com/sites/julieruvolo/2011/09/07/how-much-of-the-internet-is-actually-for-porn/>

(From BBC, July 2013) :

- Claim: is porn 4% or 37%?
- <http://www.bbc.com/news/technology-23030090/>

Sample Web Sites (Modest Size)

- Running a web site can get complicated; here is one example.
- The facts:
 - **www.fogdog.com**, online sale of sporting goods
 - Revenues: \$5 million per year
 - 2.2 million-page views per month
 - average of 20,000 unique visitors per day
- The solution (in-house):
 - Commodity hardware
 - Linux server running Apache 2.0 web servers
 - Using MySQL data base
 - They **moved to Ebay!**
 - <https://www.ebay.com/str/fogdog>
 - F5 BIG-IP OS, Apache 2.0.64 web server

Sample Web Sites (Medium size)

- Here is a popular, alternate strategy for maintaining a web site
- The facts:
 - **www.autobytel.com**, new and used cars (now **AutoWeb**)
 - Market Cap: \$33.92M (Dec. 2019)
 - Quarterly Revenues: \$28.6M (3rd Quarter 2019)
 - Lead traffic: 31.7M visits (3rd Quarter 2019)
 - Mobile version launched in 2012
 - Stock symbol: **AUTO** (Nasdaq)
- Original Microsoft solution:
 - Microsoft Windows Server
 - Microsoft IIS 7.5 web server
 - Microsoft SQL server database
 - Akamai CDN
- Today:
 - Windows Server, Microsoft IIS/7.5 web server

Sample Web Sites (large size)

- The facts:
 - **www.etrade.com**, online investing services and resources
 - Market Cap: \$10.37B (Dec. 2019)
 - Yearly Revenues: \$2.9B (12/2018)
 - 60 million-page views per month
 - average of 53,000 unique visitors per day
 - 4.9 million accounts (Jan. 2015)
 - 25,000 new retail accounts opened (Oct 2015)
 - 1,952,000 customer transactions per month
 - Taken over by Morgan Stanley in 2020 for \$13 billion
- The solution:
 - IBM 90 xSeries running Linux/**Citrix Netscaler, Apache** and Tomcat web servers, AWS Route 53 (DNS)
 - Hardware facility for load balancing and redundancy
 - Oracle database system
 - Proprietary programming systems

Web Server Farms

- Until recently all serious web sites were maintained using web server farms;
 - A group of computers acting as servers and housed in a single location;
 - Internet Service Providers (ISP's) provide web hosting services using a web server farm
- Hardware and software is used to load balance requests across the machines
- Other issues addressed by web server farms include:
 - **Redundancy** eliminates single point of failure; backup and failover strategy is required
 - **Security**, secure areas are placed behind firewalls which monitor web traffic, network address translation, port translation, SSL

Popular Web Hosting Services

- ***For individuals and small business:***
 - **1&1**
<https://www.ionos.com>
 - **GoDaddy.com**
<http://www.godaddy.com/products/secure-hosting.aspx?ci=72738>
 - **Yahoo**
<http://www.iwebhostingplans.com/yahoo/yahoowebhosting.asp>
- ***For companies willing to pay MUCH higher costs:***
 - **Rackspace**
http://www.rackspace.com/index.php?CMP=Google_hosting
 - **Network Solutions**
<http://www.networksolutions.com/web-hosting/index.jsp>
- ***Reviews and price comparisons:***
 - <http://www.hosting-review.com>
 - See next slide “monthly” prices
 - <http://www.pcmag.com/category2/0,2806,2269,00.asp>

Web Hosting Services

TOP 10 WEB HOSTING PROVIDERS - Updated December 2021

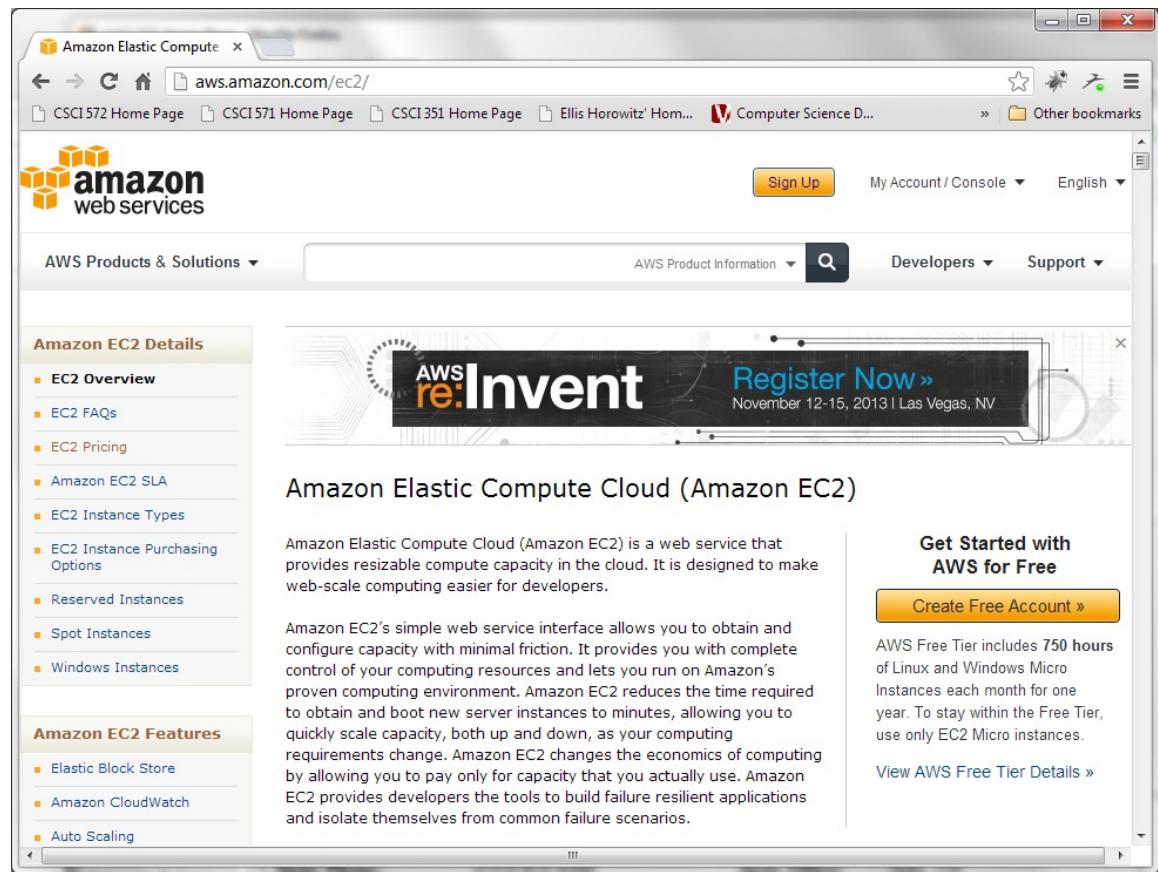
RANK	SHARED WEB HOST	PRICE	SALES INDEX	TREND	UPTIME SPEED	CUSTOMER REVIEWS	EDITOR'S REVIEW	SCORE INDEX	VISIT
1	 HostPapa MORE ▾	\$3.95					 Read	94%	Visit Site
2	 iPage MORE ▾	\$3.25					 Read	89%	Visit Site
3	 HostGator MORE ▾	\$3.95					 Read	88%	Visit Site
4	 TRUSTED BY OVER 10 MILLION CUSTOMERS MORE ▾	\$4.99					 Read	87%	Visit Site
5	 GoDaddy MORE ▾	\$6.29					 Read	86%	Visit Site
6	 TMD Hosting THE MOST DEDICATED MORE ▾	\$2.95					 Read	85%	Visit Site
7	 bluehost MORE ▾	\$3.95		-			 Read	85%	Visit Site
8	 YAHOO! Akco Small Business MORE ▾	\$3.99		-			 Read	84%	Visit Site
9	 SiteGround MORE ▾	\$6.99					 Read	83%	Visit Site
10	 A2 HOSTING MORE ▾	\$3.92					 Read	83%	Visit Site

Cloud Computing

- **Cloud computing** is Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices **on demand**, like the electricity grid.
- Users no longer have need for expertise in, or control over, the technology infrastructure "in the cloud" that supports them.
- It typically includes web-based tools or applications that users can access and use through a web browser as if it were a program installed locally on their own computer.¹
- Typical cloud computing providers deliver common business applications online that are accessed from another Web service or software like a Web browser, while the software and data are stored on servers.
- The major cloud service providers include Amazon, Google, Microsoft, Salesforce, Skytap, HP, IBM, Amazon, Google and Apple (iCloud).

An Example - Amazon's Elastic Compute Cloud

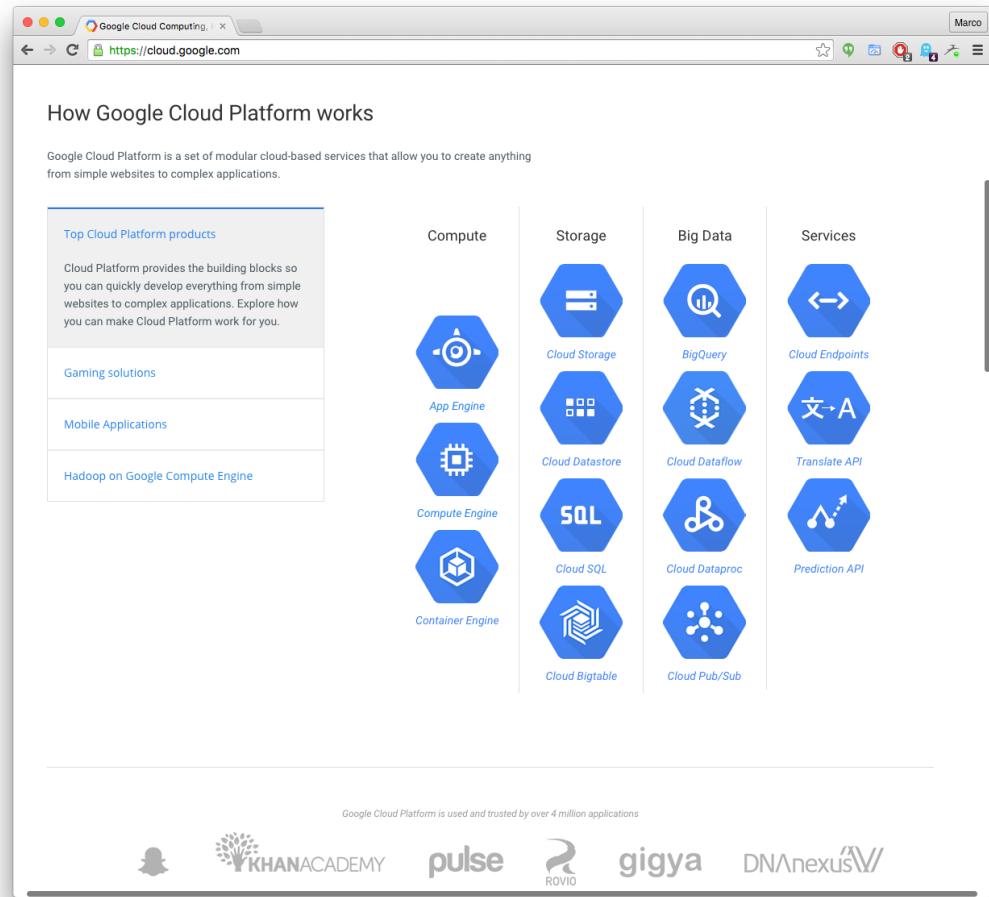
- A web service providing resizable compute capacity
- The “elastic” nature means the service instantly scales to meet demand with no up-front investment
- Users create an Amazon Machine Image (AMI), a virtual computer running your selected operating system (Linux, Windows, etc)
- Users use Amazon’s Simple Storage Service (S3) for large-scale, persistent storage
- You only pay for running AMI
- All accounts are limited to 5 Elastic IPv4 addresses per region
- See: aws.amazon.com/ec2



Amazon currently runs in 8 regions: US East, US West (Oregon), US West (Northern CA), Ireland, Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney), South America (Sao Paulo)

An Example - Google Cloud Platform

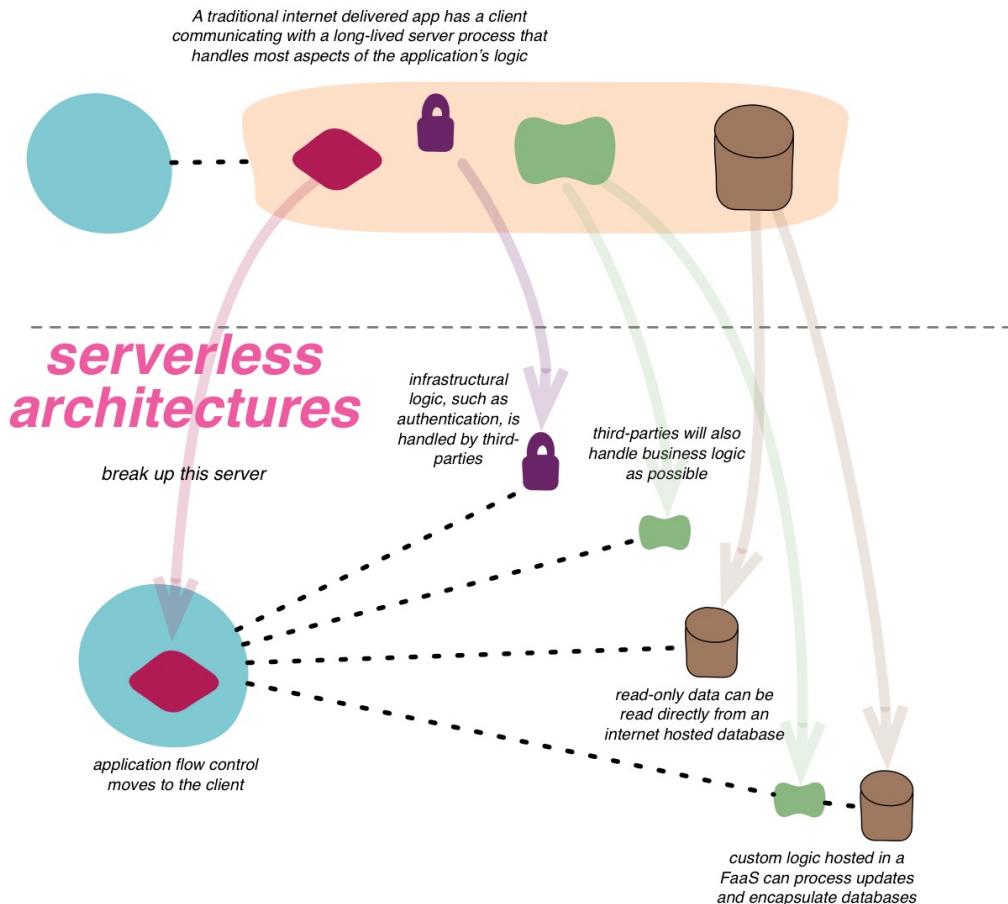
- A web service providing basic Compute, Storage, Big Data and Services.
- Additional services for massively scalable Gaming solutions, Mobile Applications backend, and Apache Hadoop.
- App Engine – A platform for building scalable web applications and mobile backends. App Engine scales applications automatically in response to the amount of traffic it receives.
- Compute Engine - Offers predefined virtual machine configurations: Debian, CentOS, CoreOS, SUSE, Ubuntu, Red Hat, FreeBSD, or Windows 2008/2012.



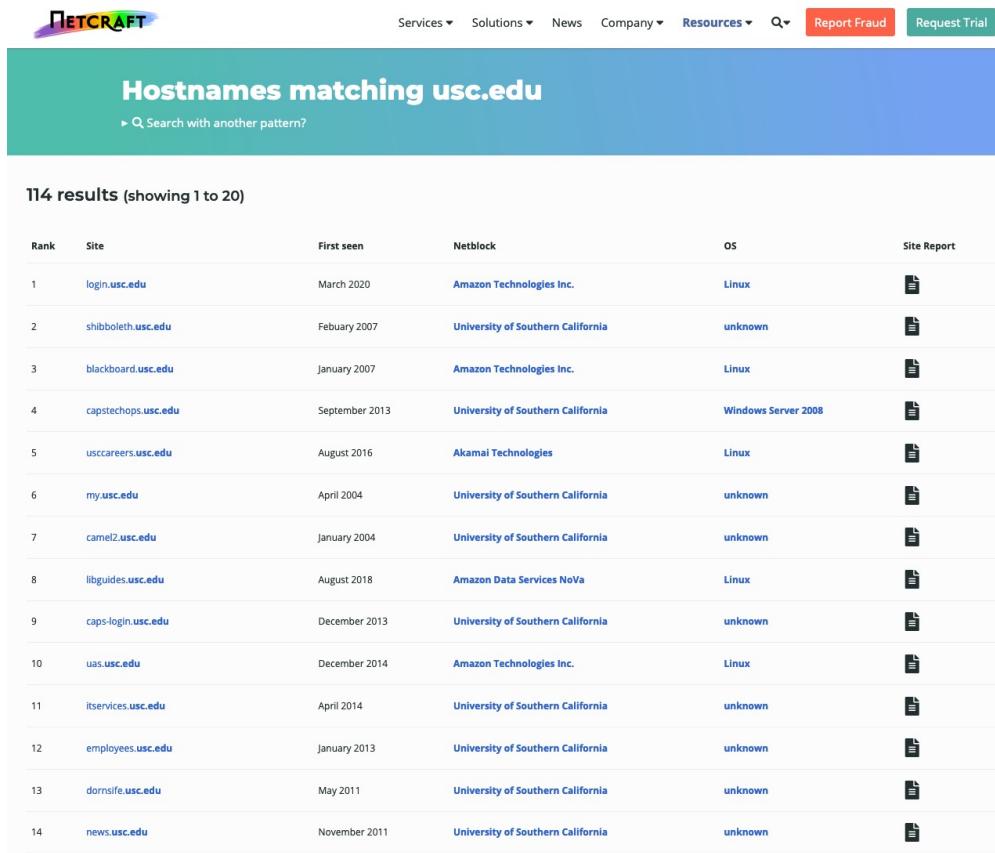
Google uses software-defined networking technology to route packets across the globe and enable fast edge-caching so that data is where it needs to be to serve users.

Serverless Architecture

- Internet based systems where the application development does not use the usual server process.
- They rely solely on a combination of:
 - third-party services, or Backend as a Service (BaaS)
 - client-side logic
 - service hosted remote procedure calls, or Function as a Service (FaaS).
- AWS Lambda is one of the most popular implementations of FaaS at present, but there are others. See:
<https://aws.amazon.com/lambda/>
- Serverless and contains will be covered later in the course



USC Has Many Web Servers Running



The screenshot shows a search results page from Netcraft. The header includes the Netcraft logo, navigation links for Services, Solutions, News, Company, Resources, a search bar, a Report Fraud button, and a Request Trial button. The main title is "Hostnames matching usc.edu". Below it is a search bar with placeholder text "Search with another pattern?". A subtitle indicates "114 results (showing 1 to 20)". The results table has columns: Rank, Site, First seen, Netblock, OS, and Site Report. The data shows 14 entries, all from the University of Southern California, with various first seen dates and netblocks like Amazon Technologies Inc., University of Southern California, and Akamai Technologies.

Rank	Site	First seen	Netblock	OS	Site Report
1	login.usc.edu	March 2020	Amazon Technologies Inc.	Linux	Report
2	shibboleth.usc.edu	February 2007	University of Southern California	unknown	Report
3	blackboard.usc.edu	January 2007	Amazon Technologies Inc.	Linux	Report
4	capstechops.usc.edu	September 2013	University of Southern California	Windows Server 2008	Report
5	usccareers.usc.edu	August 2016	Akamai Technologies	Linux	Report
6	my.usc.edu	April 2004	University of Southern California	unknown	Report
7	camel2.usc.edu	January 2004	University of Southern California	unknown	Report
8	libguides.usc.edu	August 2018	Amazon Data Services NoVa	Linux	Report
9	caps-login.usc.edu	December 2013	University of Southern California	unknown	Report
10	uas.usc.edu	December 2014	Amazon Technologies Inc.	Linux	Report
11	itservices.usc.edu	April 2014	University of Southern California	unknown	Report
12	employees.usc.edu	January 2013	University of Southern California	unknown	Report
13	dornsife.usc.edu	May 2011	University of Southern California	unknown	Report
14	news.usc.edu	November 2011	University of Southern California	unknown	Report

- Netcraft lists **106** separate sites / web servers with usc.edu in their name, e.g.
- www.usc.edu
- mat.usc.edu
- www.cs.usc.edu
- dornsife.usc.edu
- web-applusc.edu
- www-scf.usc.edu
- However, some may not be connected to USC, e.g.
- www.usc.edu.au
- Check at:
<https://searchdns.netcraft.com>

Web Browsers Use Standard Layout Engines

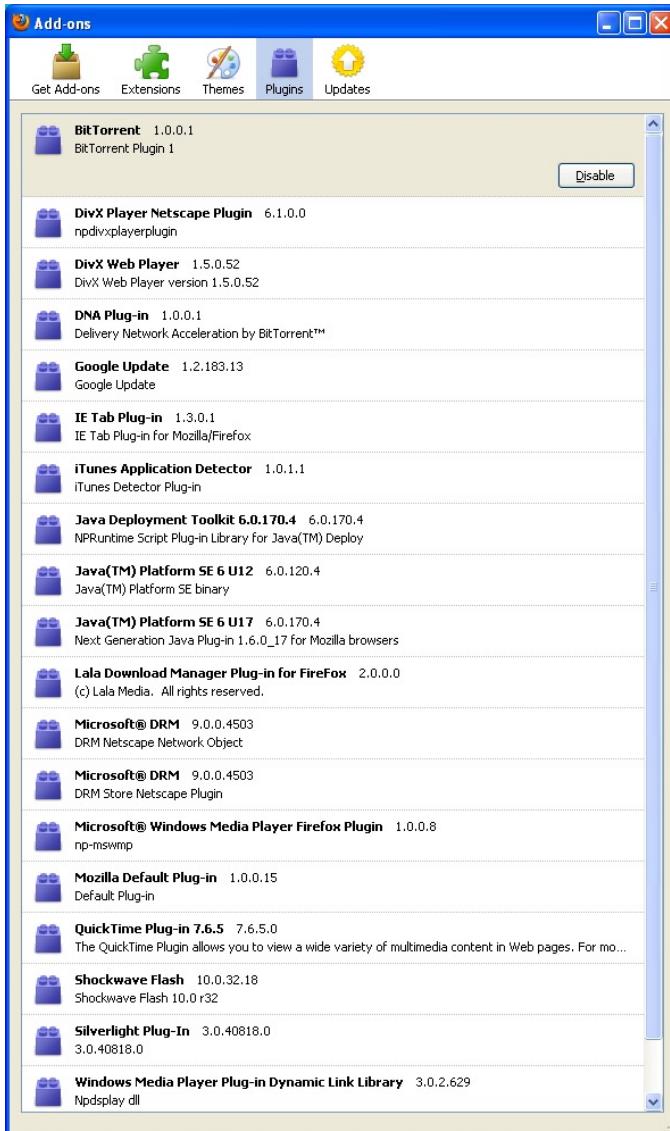
- **WebKit** is a software component used to render web pages; it is open source.
 - It is used by Google's Chrome and Apple's Safari web browsers
 - WebKit is also the name of the Mac OS X system framework version of the engine that's used by Safari, Dashboard, Mail, and many other OS X applications;
- **Gecko** is a layout engine developed by Mozilla Corporation, known as the layout engine of the Firefox web browser.
 - It is used to display web pages and, in some cases, an application's user interface.
 - It offers a rich programming API that makes it suitable for a wide variety of roles in Internet-enabled applications, such as web browsers
 - Its development originated with Netscape Communications Corporation
- Some web kits and the browsers that use them
 - **Gecko-based:** FireFox (Mozilla), Flock, Netscape
 - **Trident-shells:** Internet Explorer (Microsoft, retired)
 - **EdgeHTML:** Edge Legacy (Microsoft), fork of Trident 7, retired in Aug. 2021
 - **WebKit-based:** Chrome and Android (Google), Midori, Safari and Mobile Safari (Apple), Symbian^3 (Nokia) and many others
 - **Chromium-based:** Chrome, Microsoft Edge (2021)
 - **Presto-based:** Opera, Nintendo DS, Opera Mini, Opera Mobile
 - **Java-based:** HotJava, Lobo

Capabilities of a Browser

- Web browsers fetch and display documents from other WWW sites; their capabilities include:
 - A mouse-driven graphical user interface
 - Display of
 - Hypertext documents conforming to latest HTML standard
 - Text with fonts, styles, and varying point sizes
 - Foreign-language character sets conforming to ISO-8859
 - Forms composed of edit boxes, check boxes, radio boxes, lists, text areas, etc.
 - Graphics in different formats (GIF, JPEG, MPEG, PNG, XBM) including monochrome, color

GIF = graphic interchange format, MPEG = Motion Picture Experts Group, JPEG = Joint Photographic Experts Group, PNG = Portable Network Graphics, XBM = x bitmap

Capabilities of a Browser



- Ability to invoke helper applications and plug-ins, (**Obsolete in HTML5**) e.g.
 - *Adobe Acrobat* - used to view pdf files
 - *Windows Media Player* to play digital sound files
 - *Adobe Flash Player*, used to display video. **Retired in 2020.**)
- Ability to communicate over a secure channel, using SSL
- Ability to maintain and exchange digital certificates
- Ability to run scripts in JavaScript
- Ability to run *Java applets* and *Active X components* (**also obsolete in HTML5**)

The Browser Wars - Desktop Statistics

2021	Chrome	Edge	Firefox	Safari	Opera
November	80.0 %	6.8 %	5.8 %	3.9 %	2.4 %
October	80.3 %	6.7 %	5.7 %	3.9 %	2.3 %
September	80.9 %	6.5 %	5.6 %	3.6 %	2.2 %
August	81.4 %	6.1 %	5.6 %	3.3 %	2.1 %
July	81.6 %	6.0 %	5.6 %	3.3 %	2.2 %
June	81.7 %	5.9 %	5.6 %	3.4 %	2.2 %
May	81.2 %	5.8 %	5.8 %	3.5 %	2.4 %
April	80.7 %	5.6 %	6.1 %	3.7 %	2.4 %
March	80.8 %	5.5 %	6.3 %	3.7 %	2.3 %
February	80.6 %	5.4 %	6.6 %	3.9 %	2.3 %
January	80.3 %	5.3 %	6.7 %	3.8 %	2.3 %
2020	Chrome	Edge/IE	Firefox	Safari	Opera
December	80.5 %	5.2 %	6.7 %	3.7 %	2.3 %
November	80.0 %	5.3 %	7.1 %	3.9 %	2.3 %
October	80.4 %	5.2 %	7.1 %	3.7 %	2.1 %
September	81.0 %	4.9 %	7.2 %	3.6 %	2.0 %
August	81.2 %	4.6 %	7.3 %	3.4 %	2.0 %
July	81.3 %	4.3 %	7.6 %	3.4 %	2.0 %
June	80.7 %	3.9 %	8.1 %	3.7 %	2.1 %
May	80.7 %	3.5 %	8.5 %	4.1 %	1.6 %
April	80.7 %	3.4 %	8.6 %	4.2 %	1.5 %
March	81.4 %	3.5 %	8.7 %	3.7 %	1.3 %
February	82.0 %	3.4 %	8.7 %	3.4 %	1.2 %
January	81.9 %	3.0 %	9.1 %	3.3 %	1.3 %

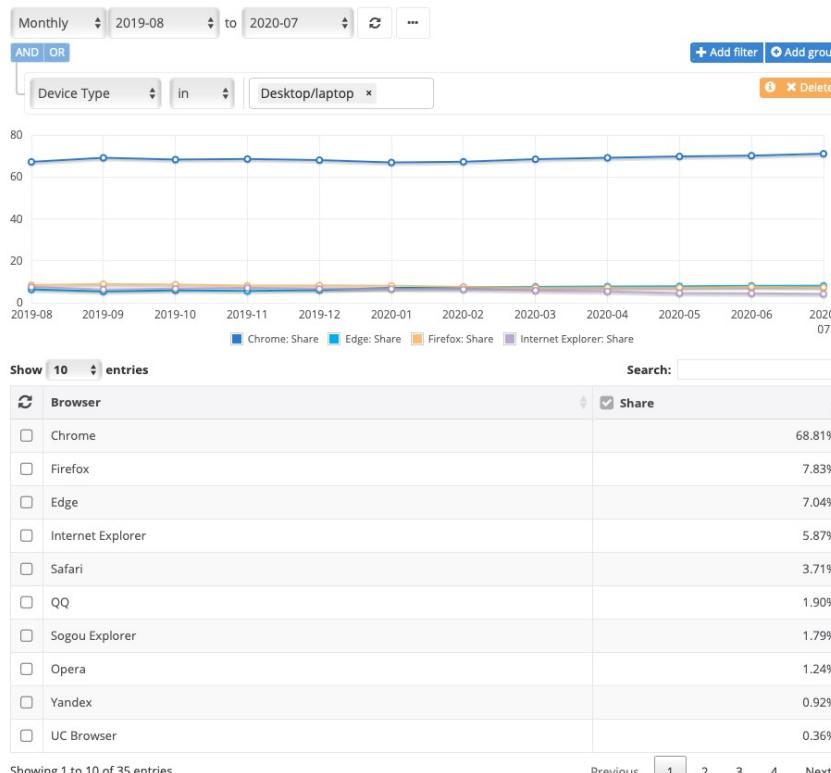
Conclusion of the above study:

- Chrome is the clear winner
- Firefox comes second, but losing ground
- Edge next
- Safari and Opera having small percentages
- WebKit total over 84%

- See http://www.w3schools.com/browsers/browsers_stats.asp
- See also <http://www.upsdell.com/BrowserNews/stat.htm>

Desktop/laptop Browser Market Share Statistics

Browser Market Share

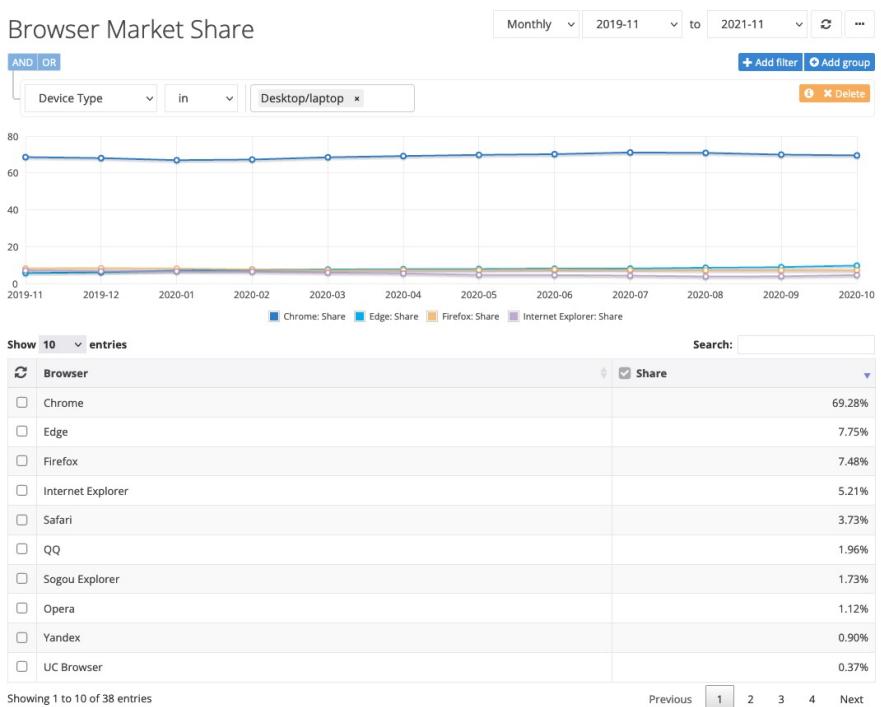


August 2020

<http://www.netmarketshare.com/>

Chrome leads with 69% market share.

Browser Market Share



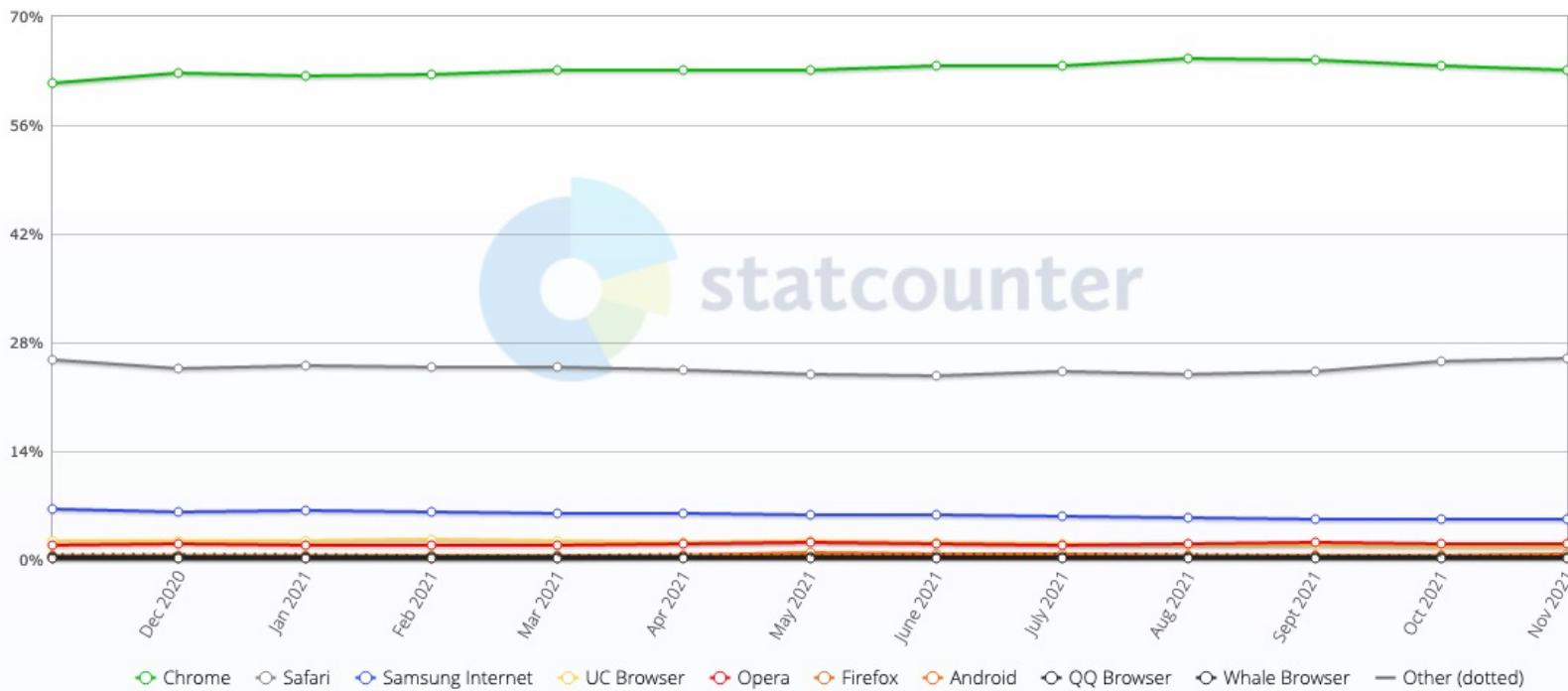
December 2021

The Browser Wars Comparison (cont'd)

Mobile Browser Market Share Worldwide

Nov 2020 - Nov 2021

Edit Chart Data



Save Chart Image (.png)

Download Data (.csv)

Embed HTML

<div id="mobile_browser-ww-monthly-202011-202111" width="600" he

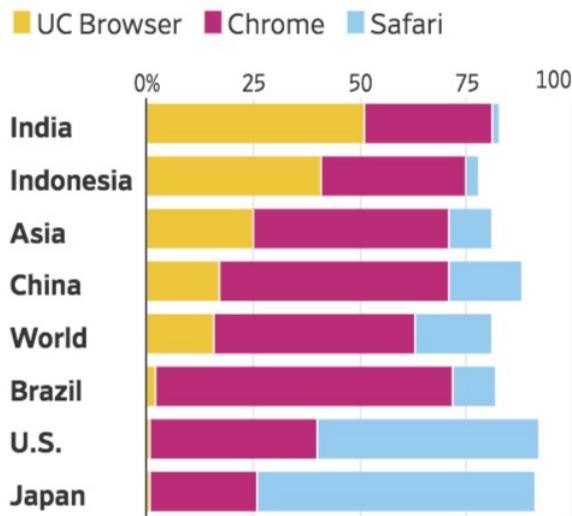
StatCounter Global Stats, May 2020 – April 2021, See <http://gs.statcounter.com>
Chrome has the lead with about 60%, followed by Safari at 24%, Samsung at 6%
and UC Browser from UCWeb of Alibaba Group of China at 3%.

The Browser Wars Comparison (cont'd)

Browser for the Next Billion

Alibaba's mobile browser, UC Browser, has a larger market share than Google's Chrome in India and Indonesia, where many of the world's 'next billion users' are getting online for the first time.

Mobile web browser market share



Note: Data Oct.-Dec. 2016 through Oct.-Dec. 2017.

Source: StatCounter

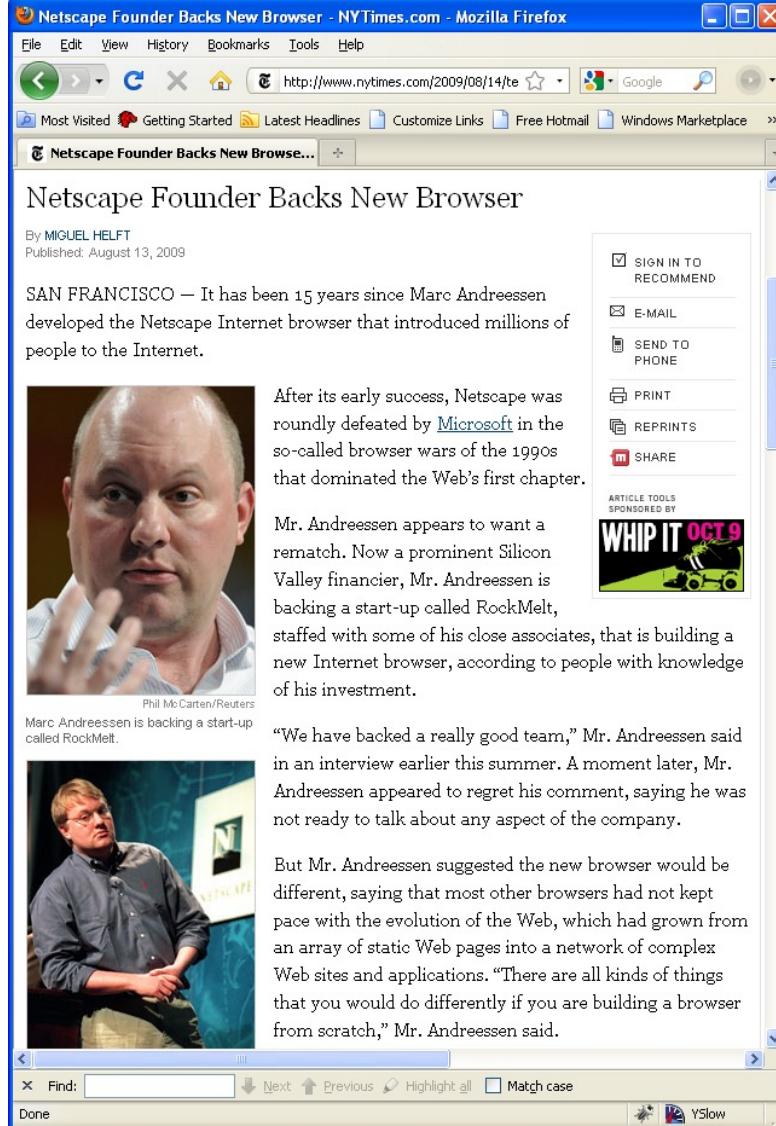
StatCounter Global Stats, Oct.-Dec. 2016 through Oct.-Dec. 2017, See
<https://www.wsj.com/articles/a-browser-youve-never-heard-of-is-dethroning-google-in-asia-1514808002>

Browsers are the Gateway to the Web/Internet

Despite Netscape's failure, there is now a new business model for browsers; Google will pay Mozilla \$300 million/year for 3 years to keep Google its default search engine. (Dec. 25, 2011)

Similarly, Google is rumored to have been paying Apple \$1 billion in 2011 to keep Google the default search engine for Safari on iOS devices and OS X.
http://articles.businessinsider.com/2012-03-09/tech/31138467_1_google-maps-ben-schachter-google-searches

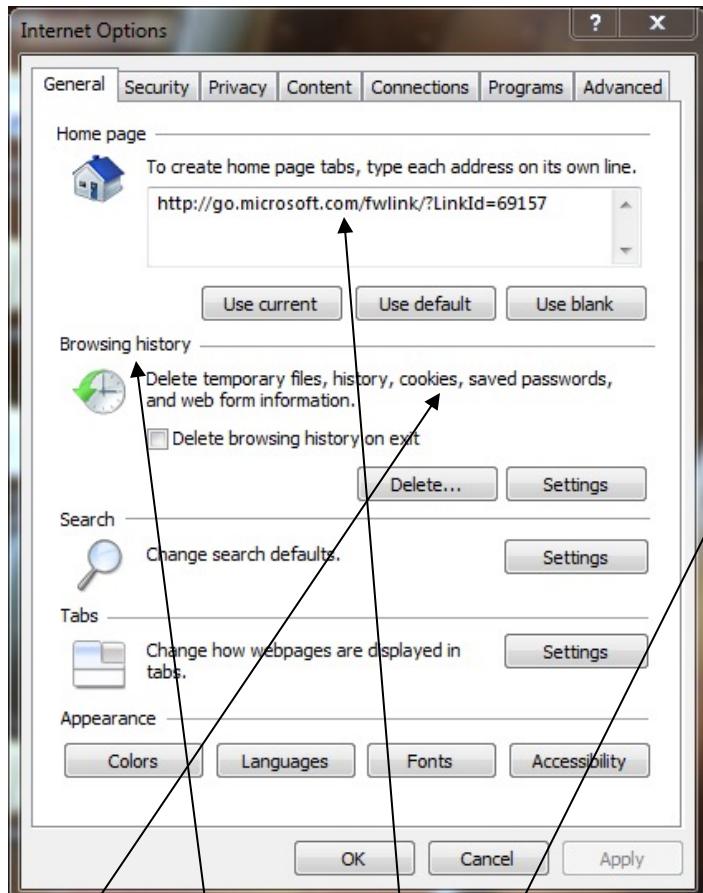
June, 2014: Apple announces DuckDuckGo will be another built-in search engine on Safari (no user tracking)



The screenshot shows a Mozilla Firefox browser window with the title bar "Netscape Founder Backs New Browser - NYTimes.com - Mozilla Firefox". The address bar shows the URL "http://www.nytimes.com/2009/08/14/technology/internet/14browser.html?_r=1&scp=1&sq=browser%20wars&st=cse". The main content area displays an article titled "Netscape Founder Backs New Browser" by MIGUEL HELFT, published on August 13, 2009. The article discusses how SAN FRANCISCO — It has been 15 years since Marc Andreessen developed the Netscape Internet browser that introduced millions of people to the Internet. It mentions that after early success, Netscape was defeated by Microsoft in the browser wars of the 1990s. The article notes that Mr. Andreessen is backing a start-up called RockMelt, staffed with some of his close associates, that is building a new Internet browser. A sidebar on the right provides sharing options like "SIGN IN TO RECOMMEND", "E-MAIL", "SEND TO PHONE", "PRINT", "REPRINTS", and "SHARE". Below the article, there is a small image of Marc Andreessen and a quote from him. At the bottom of the browser window, there is a toolbar with "Find:", "Next", "Previous", "Highlight all", "Match case", and a "Done" button.

Browser Options Menus for IE and Firefox

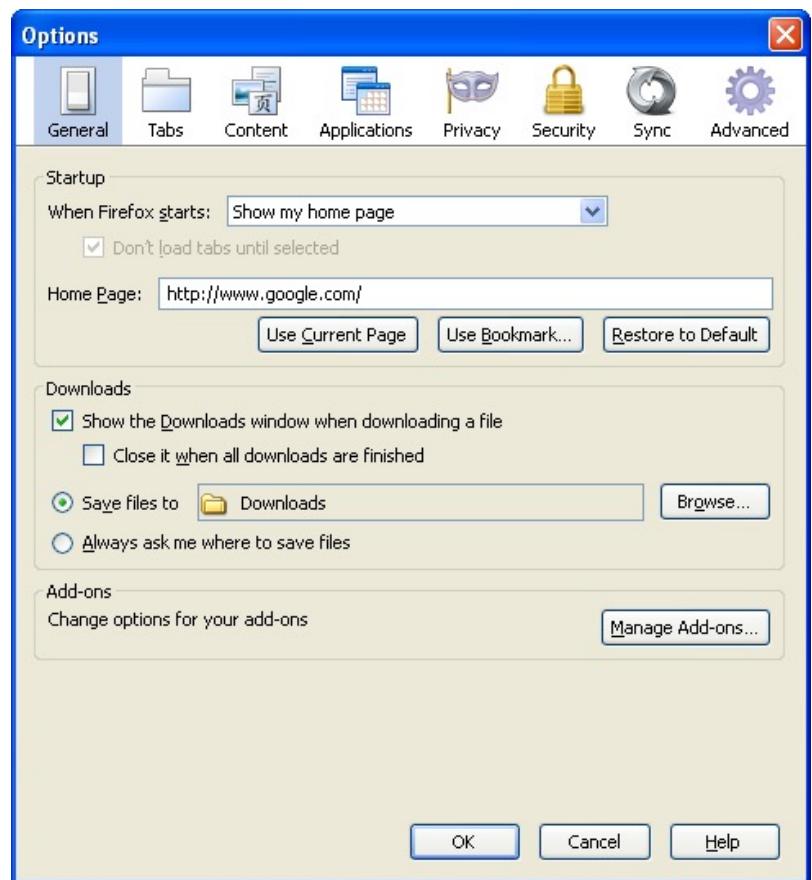
Click on Tools -> Internet Options



Internet Explorer

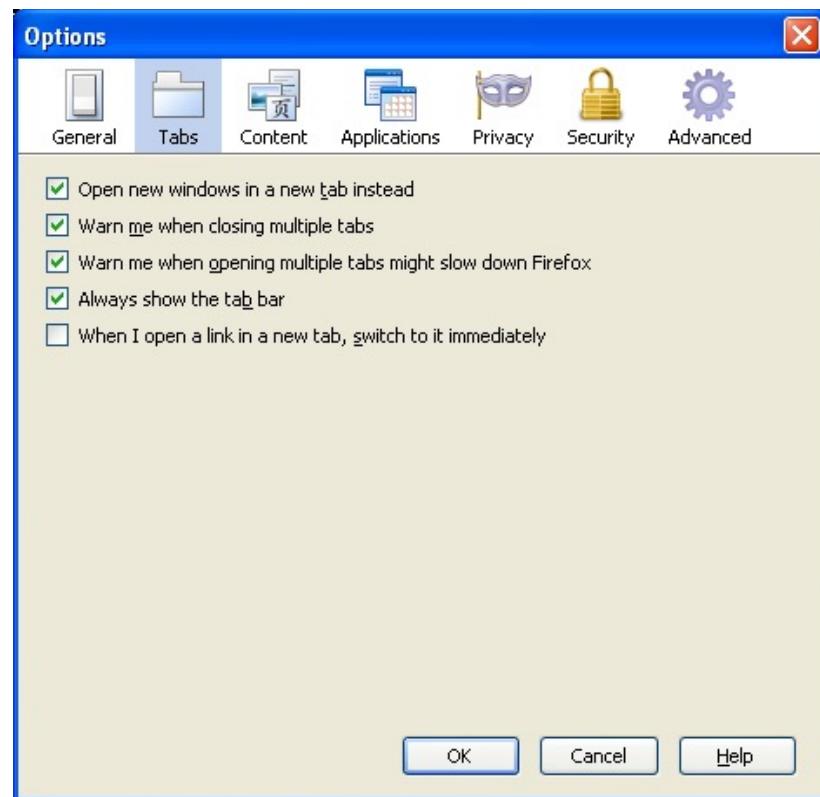
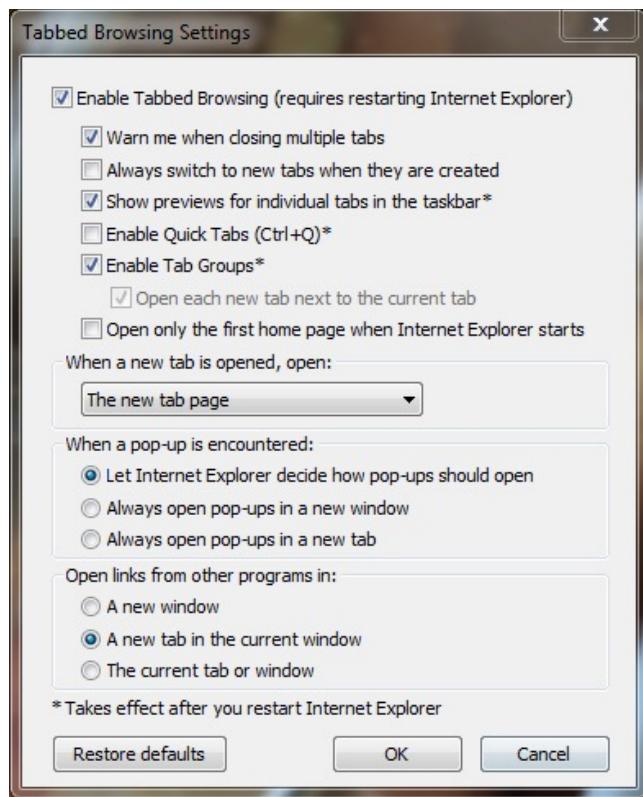
Cookies, History, default opening page

Click on Tools -> Options



Firefox – Tools | Options

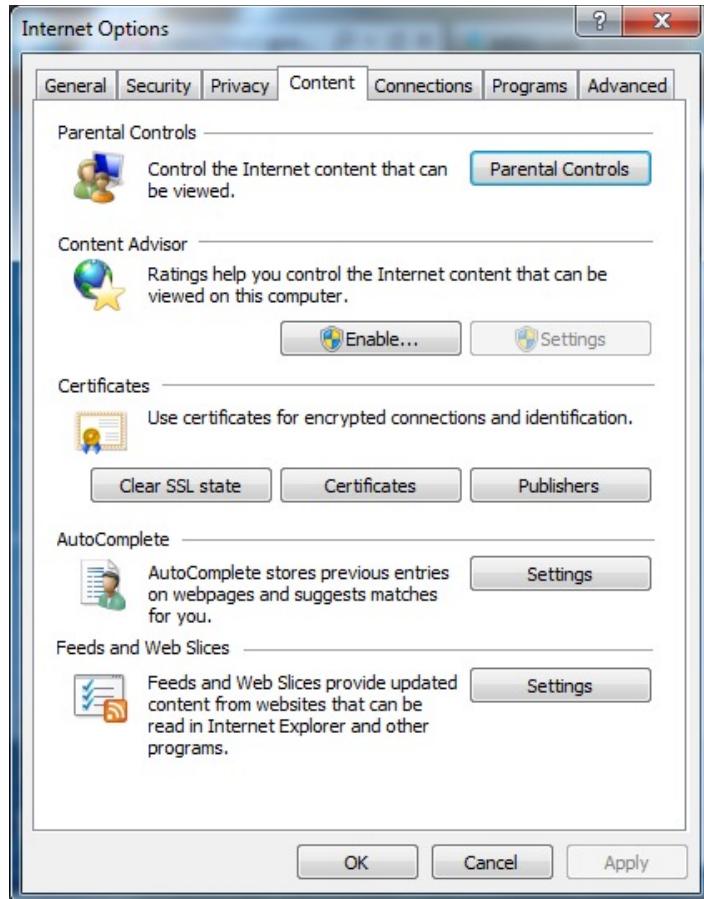
IE and Firefox Tab Control Options



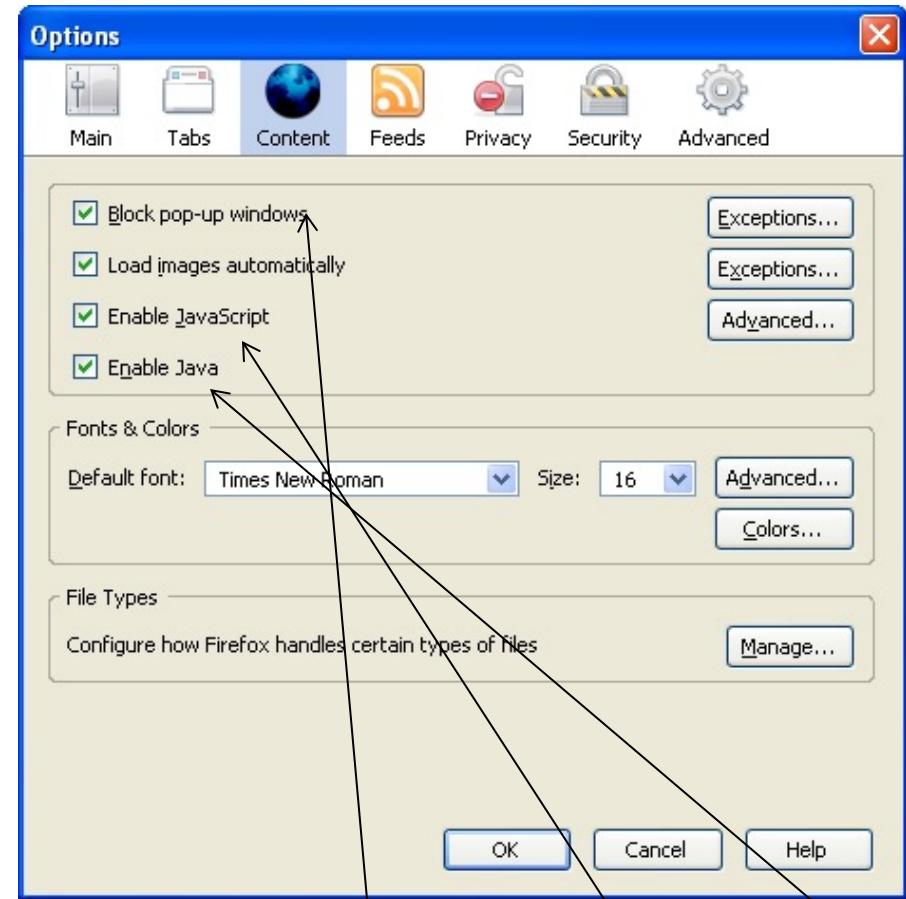
Internet Explorer

Firefox

IE and Firefox - Content Options

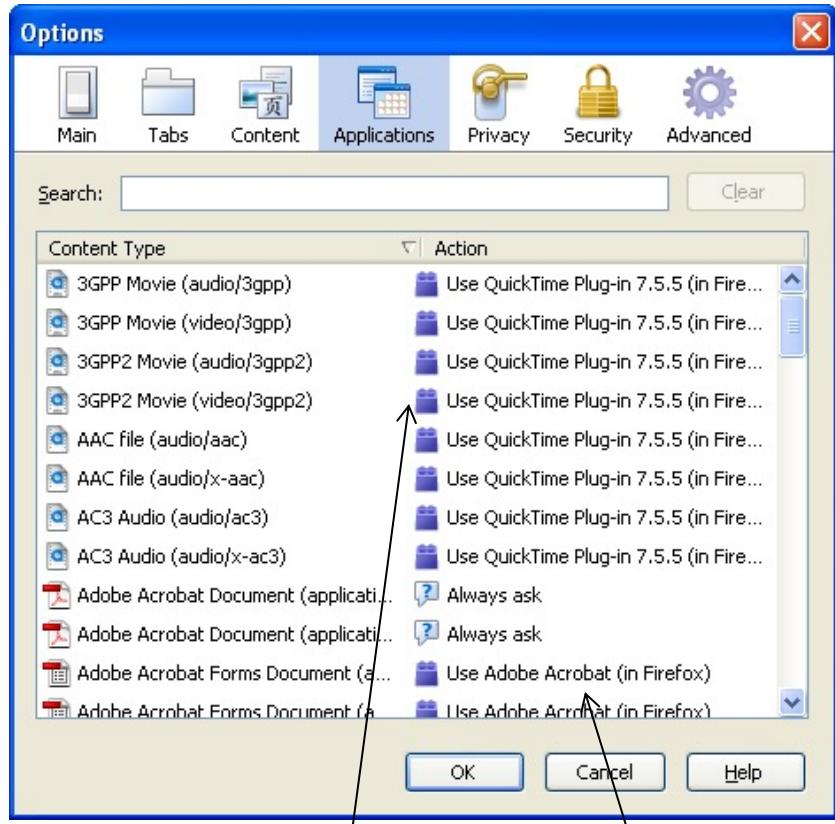
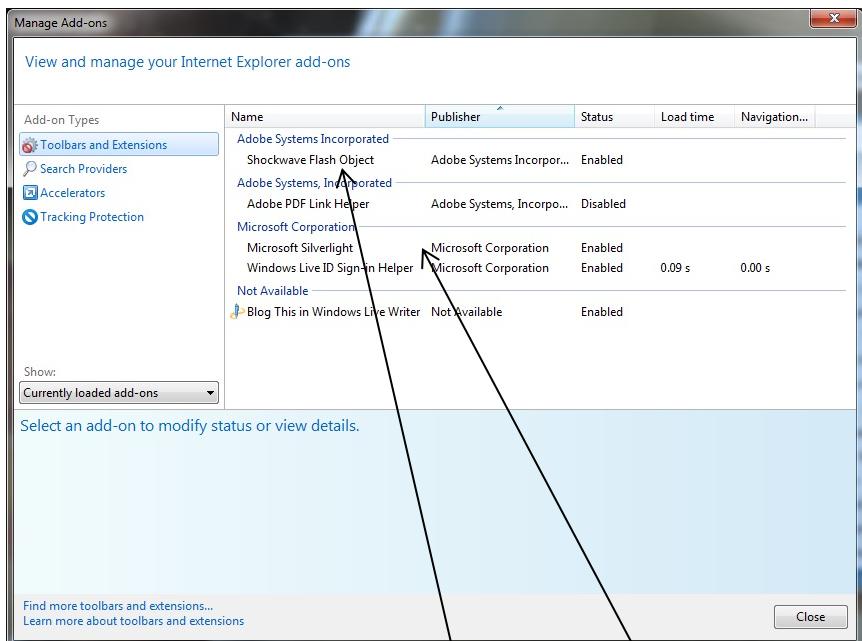


Internet Explorer



Firefox (PopUps, JavaScript, Java)

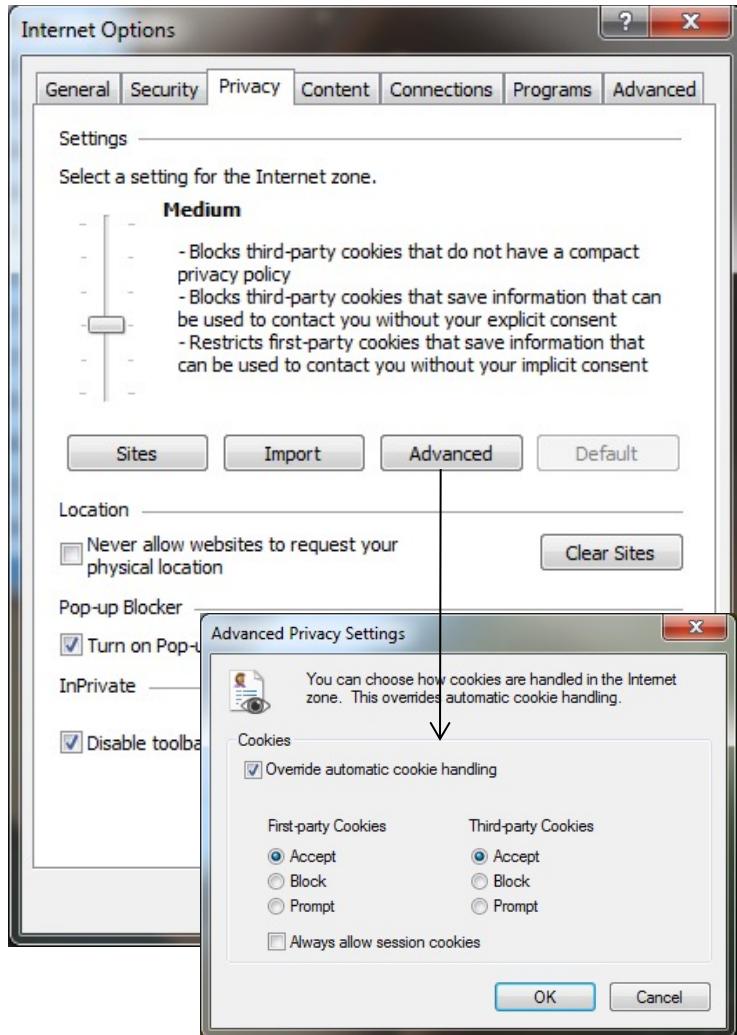
IE and Firefox Applications Options



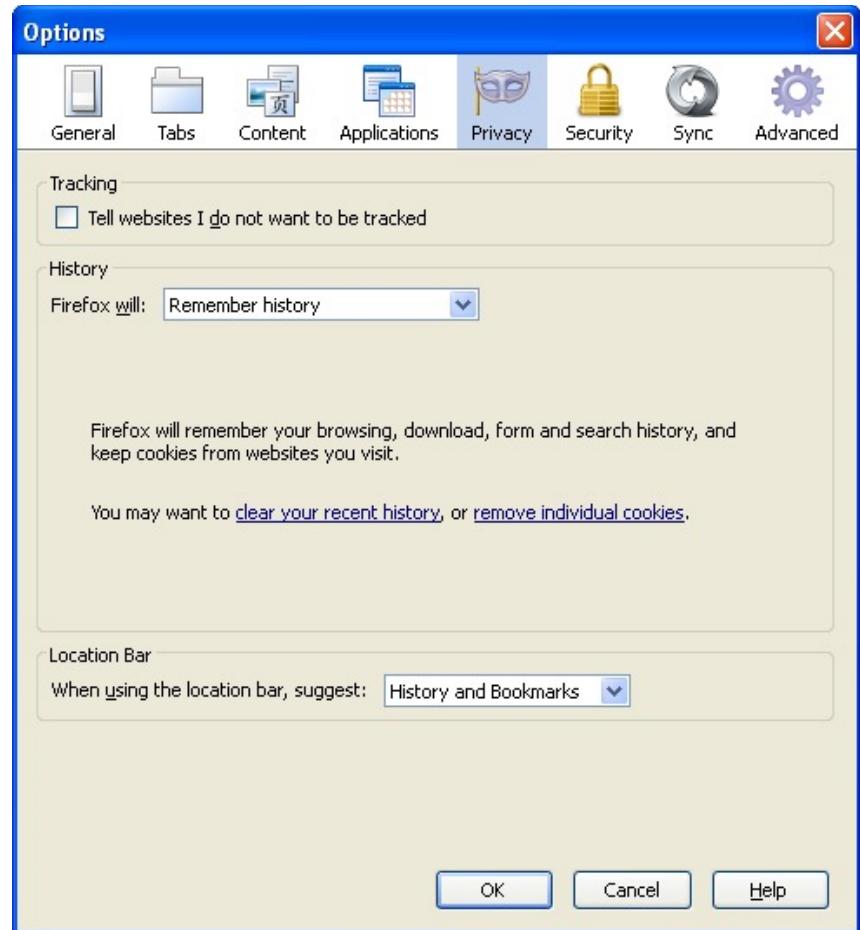
Internet Explorer (Adobe, Silverlight)
(Microsoft has now dropped Silverlight)

Firefox (Quicktime, Acrobat)

IE and Firefox Privacy Options

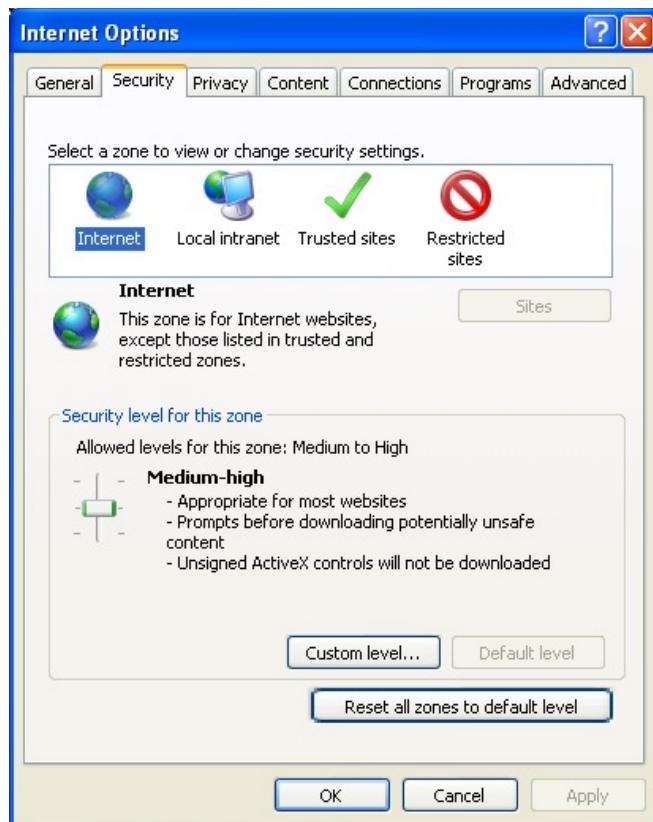


Internet Explorer (Cookies)

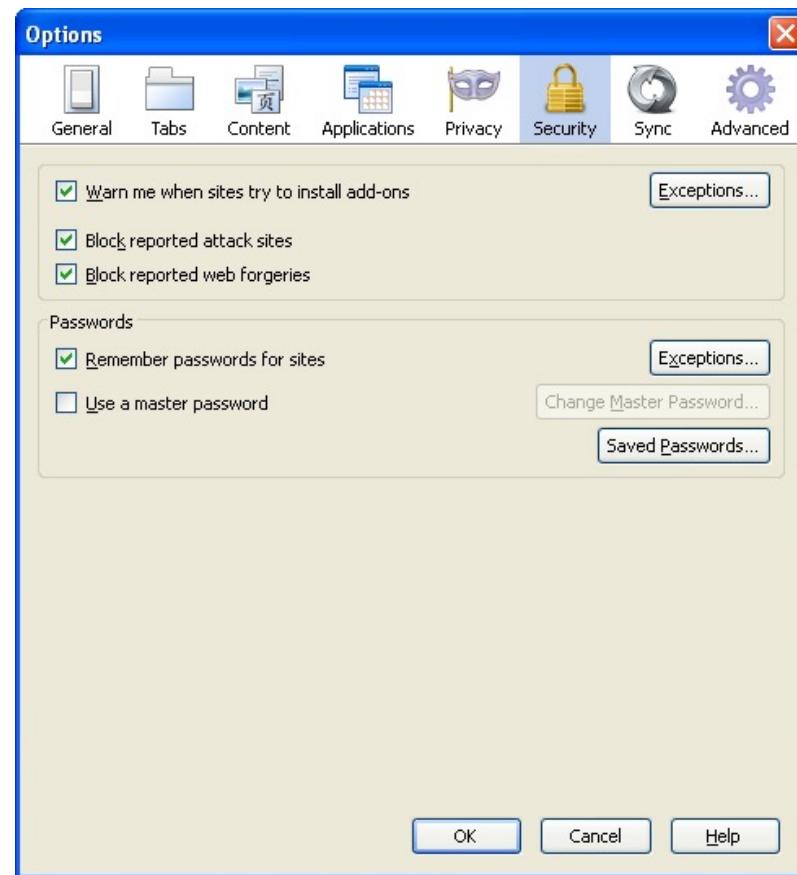


Firefox (History, Cookies)

IE and Firefox Security Options

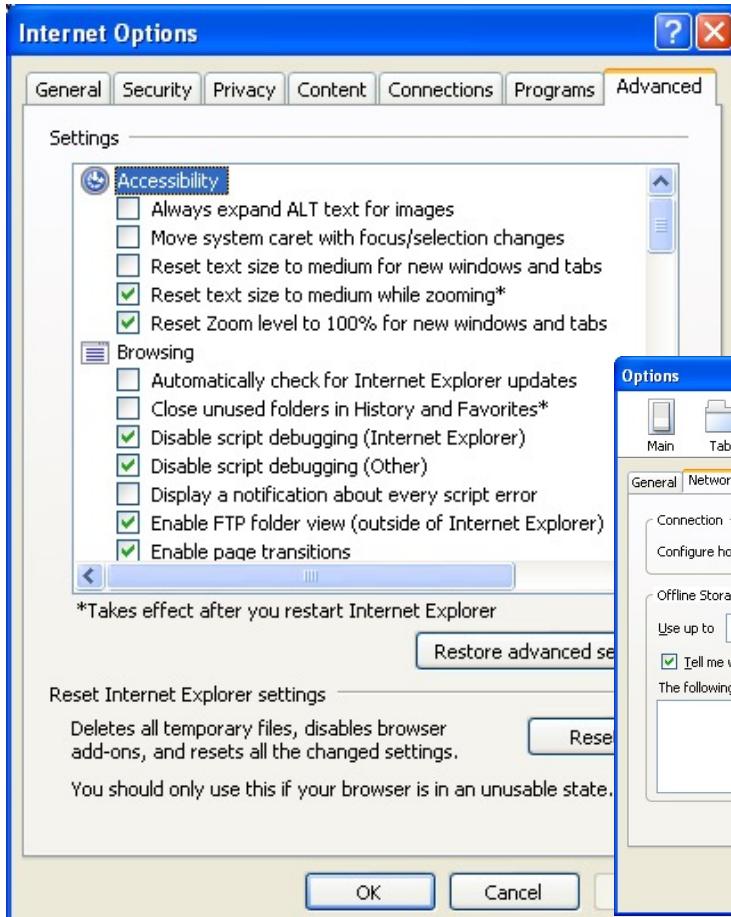


Internet Explorer

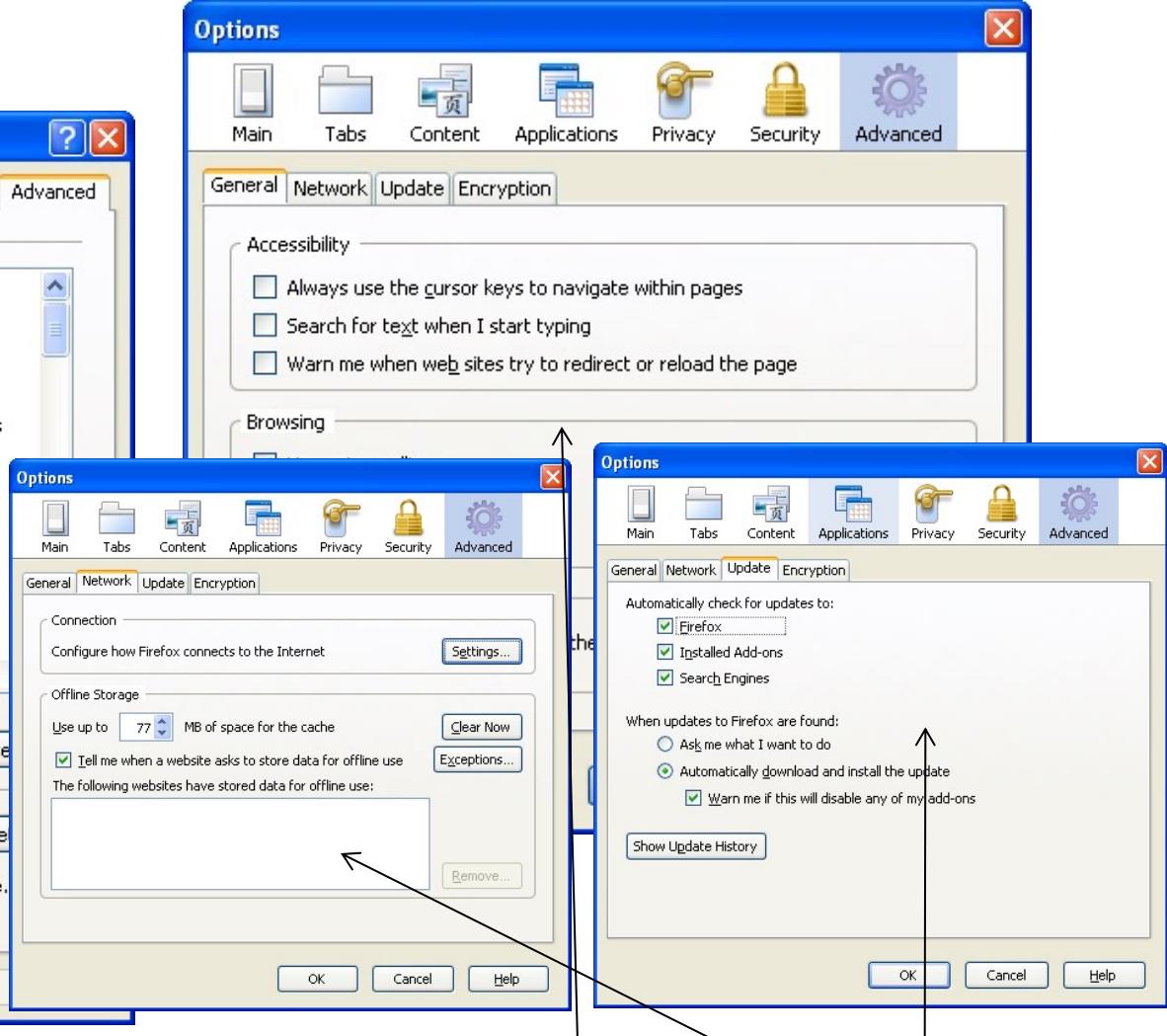


Firefox

IE and Firefox Advanced Options

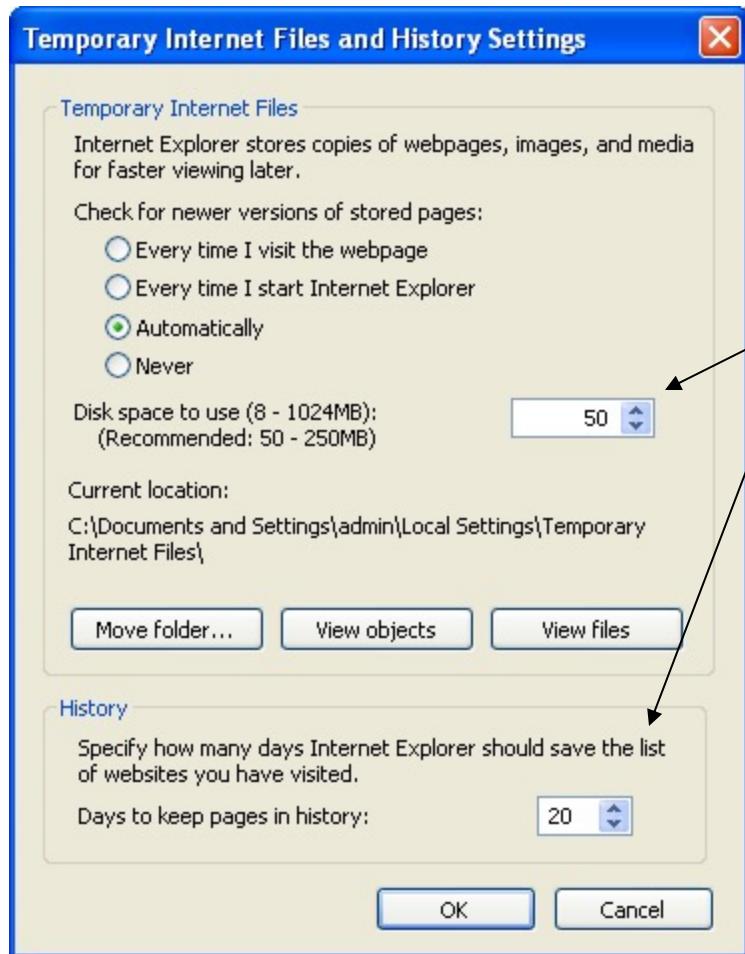


Internet Explorer



Firefox (General, Network, Update)

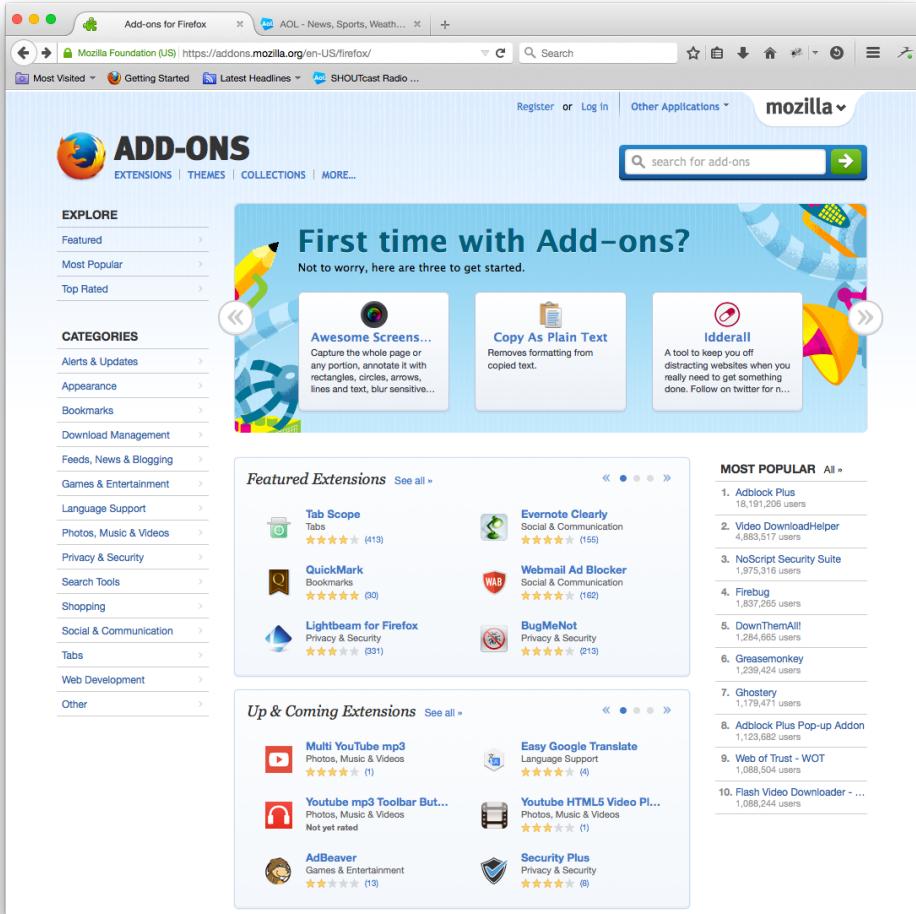
Internet Explorer Browser Caching



- History
 - Links and URLs that have been accessed by the browser over a period of time
- Disk cache
 - Temporary internet files, a folder on the disk that contains cached copies of files
- Memory cache
 - Session-based information that is cached during the session
- Offline content
 - Web content is downloaded when online and viewed offline

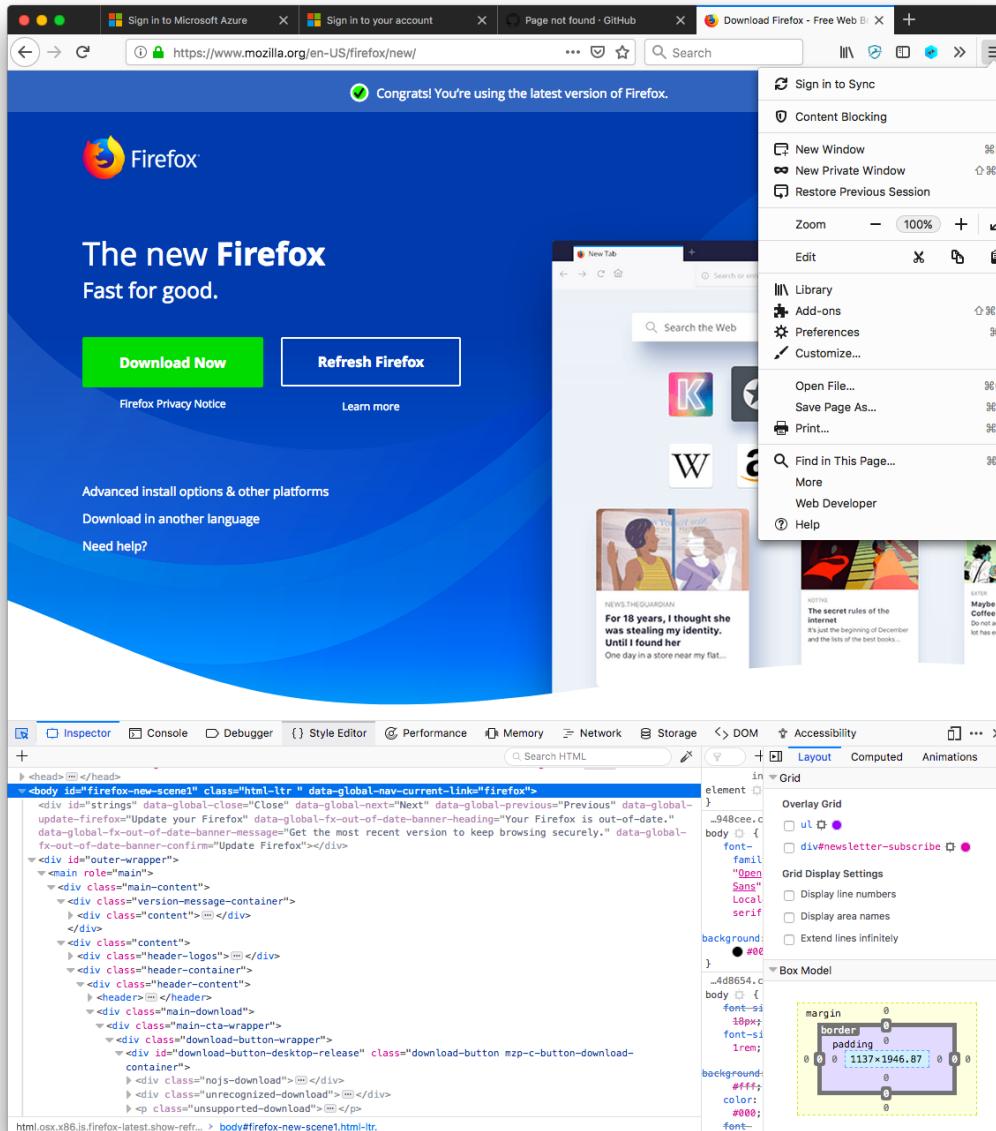
IE caching options screen (Tools | Options | General | Browsing History)

Browsers Have Many Plugins Available

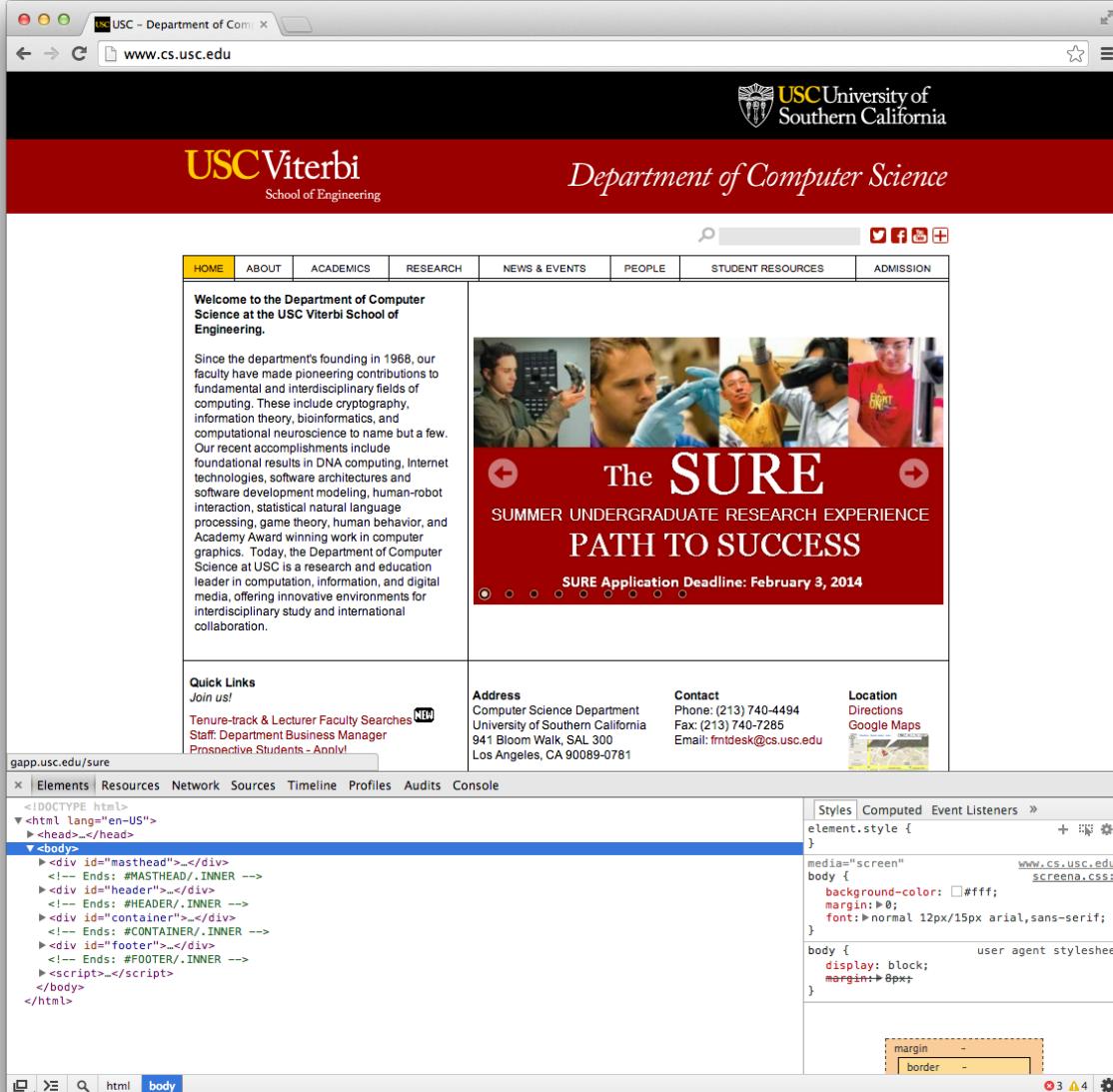


- Plug-ins are also listed as “Extensions”
- Firefox plug-ins that will be especially useful in this course are:
 - HTTP Header Live
- HTML5 does away with most video / audio plug-ins

Firefox: Tools | Web Developer



Chrome: Menu | More Tools | Developer Tools



Evolution of Web Sites

Client-centric Static	Server Applications	Web services	Service Oriented Arch.	Multi-platform (desktop, tablet, phone)	IoT, Wearables, Cloud computing, Serverless Arch. (BaaS, FaaS) Docker
	Databases	Multiple layers	(SOA)	Client-centric	
	Dynamic web pages	Business and service Integration		Client-centric	
HTML Scripts CGI	ODBC, JDBC ASP Applets, ActiveX	XML, WML, SQL, .NET COM+, Beans	Ajax, Web 2.0, JSON	HTML5, CSS3, JS gestures navigation	JS Frameworks AWS, GCP, Azure Microservices containers
1 st gen	2 nd gen	3 rd gen	4 th gen	5 th gen	6 th gen
1991	1997	2000	2005	2008	2014

Lecture

CS571 - Course Introduction

CS571: Web Technologies

- Instructor: Prof. Marco Papa
- Office hours: on **Zoom**:
 - **Mon day 5:00PM-6:00PM PDT**
 - **Meeting ID: 924 6746 7426**
 - **Passcode: 629843**
 - **iCalendar (.ics) available on D2L**
- E-mail: papa@usc.edu
- Zoom Lectures stored on DEN Desire2Learn (D2L)
- 24/7 Q&A access: Piazza, **3 min. average response time**
- Producer access: daily (6 days / week)
- Instructor access: weekly
- Quick way to ask a “personal” question: Private message to Instructors on Piazza

General Information

- Lectures:
 - **Session 30378D: Tue, Thu 5:30PM – 7:20PM (SGM 124)**
 - **Session 30016D: Tue, Thu 5:30PM – 7:20PM (DEN)**
 - **Session 30338D: Tue, Thu 5:30PM – 7:20PM (ONLINE)**
- Producers: Zoom office hours on course website and Piazza
- Course website: <https://www.csci571.com>
- DEN's D2L for Zoom Lectures: <https://courses.uscden.net>
- Assignments – yes, attendance – up to you
- Two **Exams**, one at half course and one towards the end
- Exams auto-graded using **D2L Quiz Tool**
- Mobile Application – final “comprehensive” project live demo
- Web storage: **GitHub Pages**
- Cloud Storage: **Google Cloud**

General Rules

NO FACULTY D-CLEARANCE

Unless you are a “superstar” undergrad ☺

NO COURSE OVERLAP

Unless you agree it is “your” problem

Fixed dates / times:

Exam #1: February 22, 6PM PDT

Exam #2: April 26, 6PM PDT

Final Mobile Project: May 5

Exams electronically graded using D2L Quiz Tool

Final Mobile project/presentation Zoom video uploaded to D2L

General In Person Class Rules

NO CONTACT

Do not approach Professor at lectern
BEFORE, DURING or AFTER lecture

Wear N95, K95 or surgical masks

Post questions on Zoom

Grader will answer during lecture or table for answering in Piazza

Benefits of Prof. Papa CSCI571

- Top 20 students in each section offered available Grader/Producer positions at end of semester. **BY INVITATION ONLY**
- Students that are invited to **MS Honors Program** and scored an A will be provided with **Recommendation Letter** (14 “recommended” students already awarded MS Honors)
- Students that score an “A”, can request “proof of skill” for H1B applications.
- Prof. Papa will act as “reference” in job applications

Course Objectives

- This course focuses on the phenomenon known as the World Wide Web
- Core technologies are:
 - HyperText Markup Language (HTML) and Cascading Style Sheets (CSS)
 - HyperText Transfer Protocol (HTTP)
 - Web servers, their configuration and performance properties
 - Server-Side programming using JavaScript and Python
 - Client-side programming using JavaScript and JS Frameworks
 - Ajax Development Style
- Newer Technologies of Interest
 - Responsive Website Design (Bootstrap, etc.)
 - JS Frameworks (Angular, React and Node.js)
 - Web Services (REST)
 - Web security, TOR, Dark web
 - Native Mobile frameworks (Java / Android and Swift / iOS)
 - React (native)
 - Cloud computing (AWS, GCP, Azure)
 - Serverless Applications, Containers, Docker
 - AWS Lambda, Google Cloud Functions, Azure Functions

Software and Storage

- **Student Disk space on GitHub Pages:**
 - <https://pages.github.com/>
 - **GitHub Student Developer Pack:**
<https://education.github.com/pack>
 - Allows GitHub Pages with “private” repository
 - 1GB of free web space
 - Used for homework 2 and 3 and Table of Exercises
- **Website / Web Services in the cloud**
 - Amazon’s Elastic Compute Cloud (AWS)
 - **Google Cloud Platform (GCP) Preferred!!!**
 - Microsoft Azure
 - AWS Lambda, Google Cloud Functions, Azure Functions
 - Serverless.com
 - Node.js
 - Docker

Other Issues

- Piazza class news group
 - Activate your membership by self-joining at:
piazza.com/usc/spring2022/csci571
 - Class Access Code: **lafc3252usc**
- Academic Integrity Policy
 - Do NOT submit the same program; you can discuss the project with fellow students, but do not develop code with other students; do not download code online; do not post code online; we use MOSS to check for plagiarism (similar code). We scan all the exams. See "Academic Integrity Policy".
- Downloading course slides and software
 - Class slides access. Username: **csci571**, password: **notes1**
 - All software and installation instructions can be downloaded from the class website.

Student Evaluations

- Comments:
 - "Amazing assignments. Learnt a lot on the course."
 - "Projects seemed similar to an actual client for web development would ask for."
 - "Even though the assignments were hard I learnt a lot from them."
 - "It is not a fair game for beginners. I've spent almost 3 weeks to do a homework, and I still can not finish it on time."
 - "Course projects are impressive!"
 - "This class has posted assignments easily x10 times larger than other classes."
 - "HW8 and HW9 take *forever*."
 - "The homework assignments are so difficult."
 - "I had to do so much googling on my own to learn about concepts used in the homework assignments."
 - "Tough class with a lot of valuable assignments."
 - "Massive assignments."
 - "The workload of this course is too much, especially the last two homework."

Academic Integrity Violations

- Spring 2018 violations (16):
 - Sanctioned: 16
 - Appeals to Engineering Panel Review: 8, no changes
 - Appeals to Engineering Dean: 2, no changes
 - **F in course:** 12
 - 0 + full letter grade reduction: 4 (C-, C-, C-, B-)
- Fall 2018 violations (0):
 - None!
- Spring 2019 violations (4):
 - **F in course:** 1
 - 0 + full letter grade reduction: 1 (B-)
 - C in the course: 2
- Fall 2019 violations (6):
 - **F in course:** 1
 - 0 + full letter grade reduction: 5 (B-, C)
- Spring 2020 violations (4):
 - 0 + full letter grade reduction: 1 (A-, C)
- Spring 2021 violations (2)
 - Full letter grade reduction

Piazza

general advice
retrieved from →
last semester

The screenshot shows a web browser window with the title bar "File Edit View History Bookmarks Tools Help" and the address bar "CSCI 571 (20 unread) https://piazza.com/class/hxnr3zbzv156m". The main content area displays the Piazza course interface for "CSCI 571".

Class at a Glance (Updated 1 minute ago)

- 20 unread posts**
- no unanswered questions**
- no unresolved followups**
- 21 total posts
- 172 total contributions
- 0 instructors' responses
- 0 students' responses
- n/a avg. response time

Student Enrollment ..out of 350 (estimated)
13 enrolled

Share Your Class
Professors appreciate Piazza best when they see how it is being used.
Allow colleagues to view your class through a demo link - a restricted, read only version of your class where all students' names are anonymized and all student information hidden.

https://piazza.com/demo_login?nid=hxnr3zbzv156m&auth=ff4abc4
Opening this link in the same browser will log you out as horowitz@usc.edu

Recent Posts

- Instr Tip 1: Add link to a question (2 Mon)
- Instr Homework #3: Practice Writing (41 Mon)
- Instr Homework #2: Creating Your... (44 Mon)
- Instr Homework #1: Join the Clas... (6 Mon)
- Instr Welcome to CSCI 571! (5 Mon)
- WEEK 7/13 - 7/19
- Private Introduce Piazza to your stud... (7/15/14)
- Private Get familiar with Piazza (7/15/14)
- Private Tips & Tricks for a success... (7/15/14)
- Welcome to Piazza! (7/15/14)

Who am I?

- PhD in CS from USC, class of '88
- PhD Advisor: Prof. Ellis Horowitz
- Initial career: MS Windows, Commodore Amiga developer
- System Architect -> Team Lead -> IT Project Manager -> VP Engineering
- CTO (Chief Technology Officer) at Luckman Interactive and CareerBuilder
- Faculty at USC since 2003
- Chief Technologist at LASC (Los Angeles Superior Court) since 2002
- O365 SharePoint Project Manager since 2014
- Active member of LA CTO Forum (invitation only group of Silicon Beach CTOs)
- Season ticket holder of USC Football and  LAFC (Los Angeles Football Club) and  supporter

Characterizing Web Content

There are very few studies that examine the types of content on the web, however . . .

(From IEEE Spectrum, Jan. 2004, pp. 75) :

- Claim: 30% of the web is porn
- Claim: 30% of the web is duplicate information
- 50,000,000 pages are either new or changed each day
- 65% of the web pages are in English

(From Personal Computer World, Optenet, Sep. 2008) :

- Claim: 35% of the web is porn, 11% is e-commerce
- <http://www.optenet.com/en-us/new.asp?id=162>

(From Forbes, Sept. 2011) :

- Claim: 4% is porn, 13% are porn Web Searches
- <http://www.forbes.com/sites/julieruvolo/2011/09/07/how-much-of-the-internet-is-actually-for-porn/>

(From BBC, July 2013) :

- Claim: is porn 4% or 37%?
- <http://www.bbc.com/news/technology-23030090/>

Sample Web Sites (Modest Size)

- Running a web site can get complicated; here is one example.
- The facts:
 - **www.fogdog.com**, online sale of sporting goods
 - Revenues: \$5 million per year
 - 2.2 million-page views per month
 - average of 20,000 unique visitors per day
- The solution (in-house):
 - Commodity hardware
 - Linux server running Apache 2.0 web servers
 - Using MySQL data base
 - They **moved to Ebay!**
 - <https://www.ebay.com/str/fogdog>
 - F5 BIG-IP OS, Apache 2.0.64 web server

Sample Web Sites (Medium size)

- Here is a popular, alternate strategy for maintaining a web site
- The facts:
 - **www.autobytel.com**, new and used cars (now **AutoWeb**)
 - Market Cap: \$33.92M (Dec. 2019)
 - Quarterly Revenues: \$28.6M (3rd Quarter 2019)
 - Lead traffic: 31.7M visits (3rd Quarter 2019)
 - Mobile version launched in 2012
 - Stock symbol: **AUTO** (Nasdaq)
- Original Microsoft solution:
 - Microsoft Windows Server
 - Microsoft IIS 7.5 web server
 - Microsoft SQL server database
 - Akamai CDN
- Today:
 - Windows Server, Microsoft IIS/7.5 web server

Sample Web Sites (large size)

- The facts:
 - **www.etrade.com**, online investing services and resources
 - Market Cap: \$10.37B (Dec. 2019)
 - Yearly Revenues: \$2.9B (12/2018)
 - 60 million-page views per month
 - average of 53,000 unique visitors per day
 - 4.9 million accounts (Jan. 2015)
 - 25,000 new retail accounts opened (Oct 2015)
 - 1,952,000 customer transactions per month
 - Taken over by Morgan Stanley in 2020 for \$13 billion
- The solution:
 - IBM 90 xSeries running Linux/**Citrix Netscaler, Apache** and Tomcat web servers, AWS Route 53 (DNS)
 - Hardware facility for load balancing and redundancy
 - Oracle database system
 - Proprietary programming systems

Web Server Farms

- Until recently all serious web sites were maintained using web server farms;
 - A group of computers acting as servers and housed in a single location;
 - Internet Service Providers (ISP's) provide web hosting services using a web server farm
- Hardware and software is used to load balance requests across the machines
- Other issues addressed by web server farms include:
 - **Redundancy** eliminates single point of failure; backup and failover strategy is required
 - **Security**, secure areas are placed behind firewalls which monitor web traffic, network address translation, port translation, SSL

Popular Web Hosting Services

- ***For individuals and small business:***
 - **1&1**
<https://www.ionos.com>
 - **GoDaddy.com**
<http://www.godaddy.com/products/secure-hosting.aspx?ci=72738>
 - **Yahoo**
<http://www.iwebhostingplans.com/yahoo/yahoowebhosting.asp>
- ***For companies willing to pay MUCH higher costs:***
 - **Rackspace**
http://www.rackspace.com/index.php?CMP=Google_hosting
 - **Network Solutions**
<http://www.networksolutions.com/web-hosting/index.jsp>
- ***Reviews and price comparisons:***
 - <http://www.hosting-review.com>
 - See next slide “monthly” prices
 - <http://www.pcmag.com/category2/0,2806,2269,00.asp>

Web Hosting Services

TOP 10 WEB HOSTING PROVIDERS - Updated December 2021

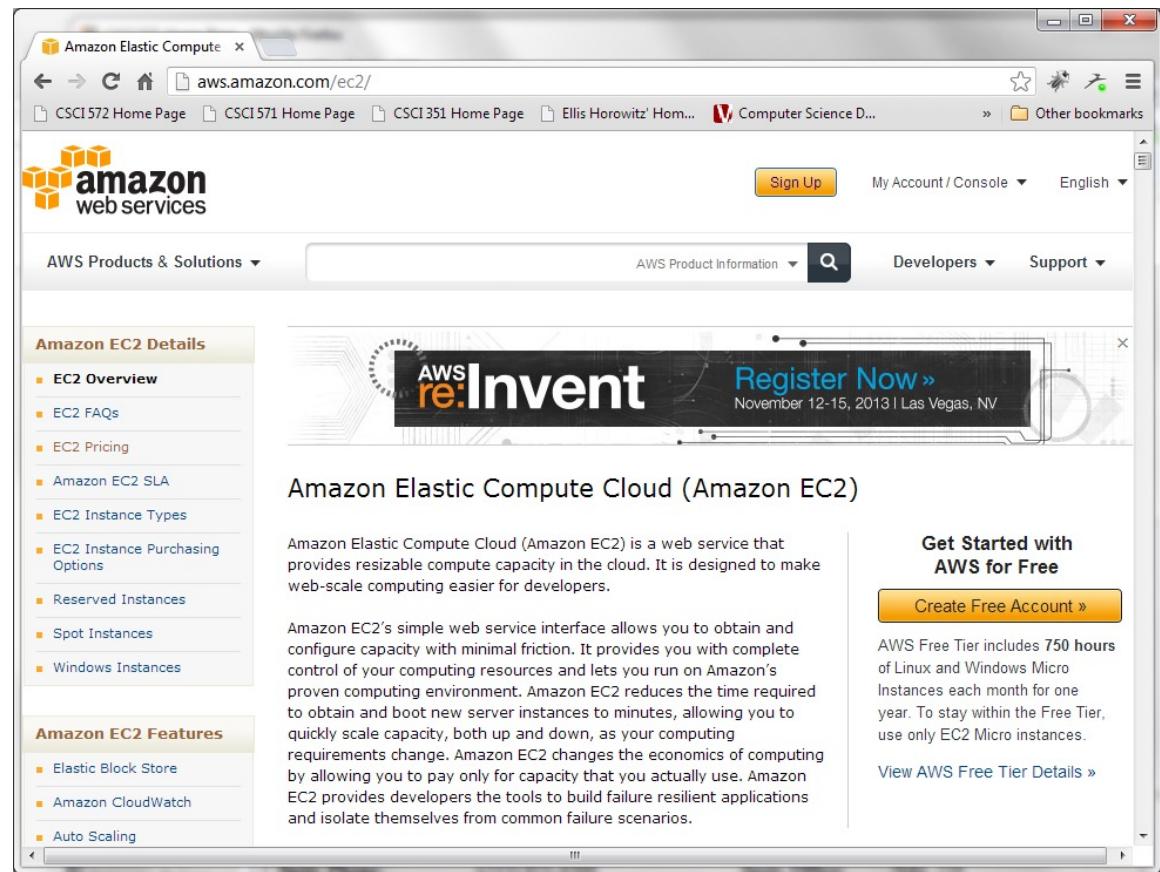
RANK	SHARED WEB HOST	PRICE	SALES INDEX	TREND	UPTIME SPEED	CUSTOMER REVIEWS	EDITOR'S REVIEW	SCORE INDEX	VISIT
1	 HostPapa MORE ▾	\$3.95					 Read	94%	Visit Site
2	 iPage MORE ▾	\$3.25					 Read	89%	Visit Site
3	 HostGator MORE ▾	\$3.95					 Read	88%	Visit Site
4	 TRUSTED BY OVER 10 MILLION CUSTOMERS MORE ▾	\$4.99					 Read	87%	Visit Site
5	 GoDaddy MORE ▾	\$6.29					 Read	86%	Visit Site
6	 TMD Hosting THE MOST DEDICATED MORE ▾	\$2.95					 Read	85%	Visit Site
7	 bluehost MORE ▾	\$3.95		-			 Read	85%	Visit Site
8	 YAHOO! Akco Small Business MORE ▾	\$3.99		-			 Read	84%	Visit Site
9	 SiteGround MORE ▾	\$6.99					 Read	83%	Visit Site
10	 A2 HOSTING MORE ▾	\$3.92					 Read	83%	Visit Site

Cloud Computing

- **Cloud computing** is Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices **on demand**, like the electricity grid.
- Users no longer have need for expertise in, or control over, the technology infrastructure "in the cloud" that supports them.
- It typically includes web-based tools or applications that users can access and use through a web browser as if it were a program installed locally on their own computer.¹
- Typical cloud computing providers deliver common business applications online that are accessed from another Web service or software like a Web browser, while the software and data are stored on servers.
- The major cloud service providers include Amazon, Google, Microsoft, Salesforce, Skytap, HP, IBM, Amazon, Google and Apple (iCloud).

An Example - Amazon's Elastic Compute Cloud

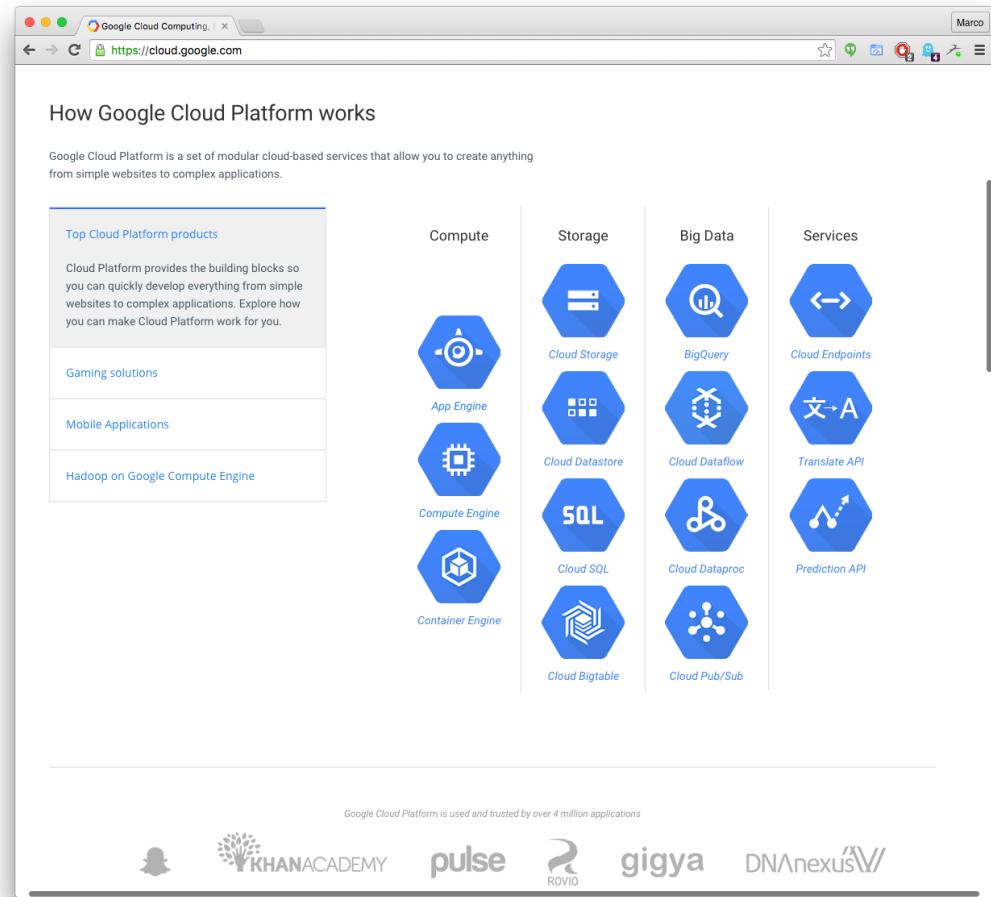
- A web service providing resizable compute capacity
- The “elastic” nature means the service instantly scales to meet demand with no up-front investment
- Users create an Amazon Machine Image (AMI), a virtual computer running your selected operating system (Linux, Windows, etc)
- Users use Amazon’s Simple Storage Service (S3) for large-scale, persistent storage
- You only pay for running AMI
- All accounts are limited to 5 Elastic IPv4 addresses per region
- See: aws.amazon.com/ec2



Amazon currently runs in 8 regions: US East, US West (Oregon), US West (Northern CA), Ireland, Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney), South America (Sao Paulo)

An Example - Google Cloud Platform

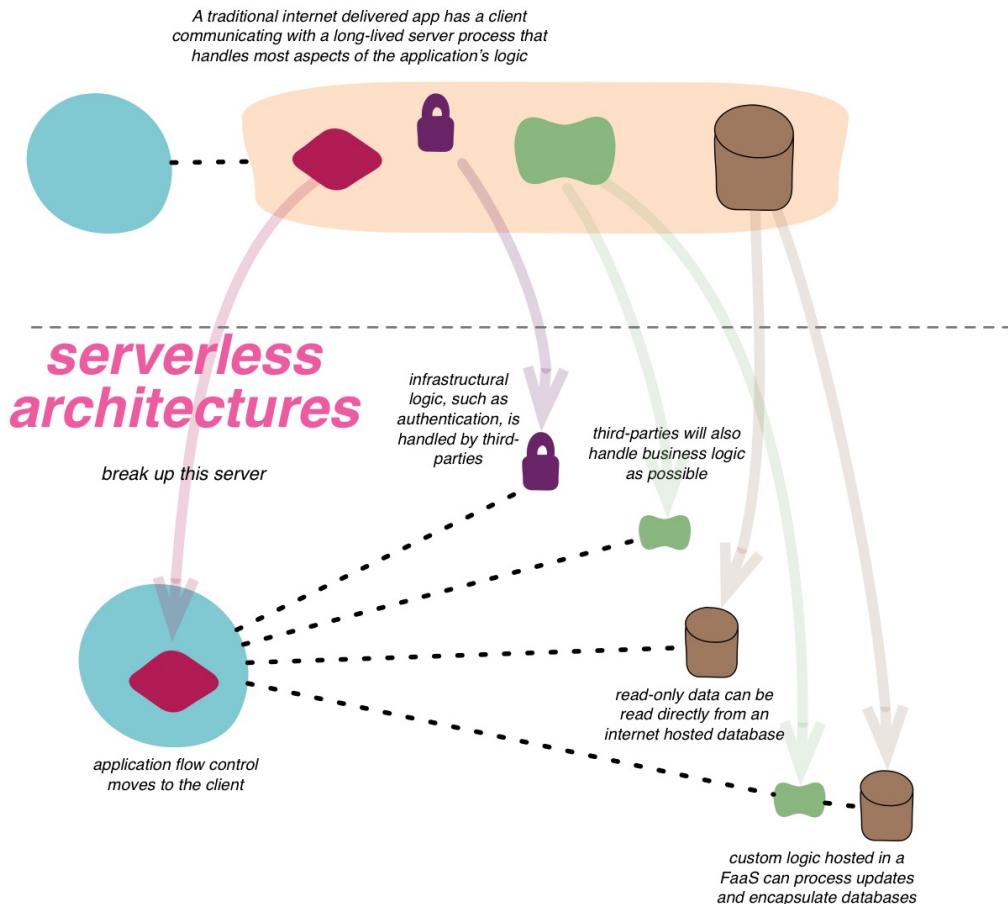
- A web service providing basic Compute, Storage, Big Data and Services.
- Additional services for massively scalable Gaming solutions, Mobile Applications backend, and Apache Hadoop.
- App Engine – A platform for building scalable web applications and mobile backends. App Engine scales applications automatically in response to the amount of traffic it receives.
- Compute Engine - Offers predefined virtual machine configurations: Debian, CentOS, CoreOS, SUSE, Ubuntu, Red Hat, FreeBSD, or Windows 2008/2012.



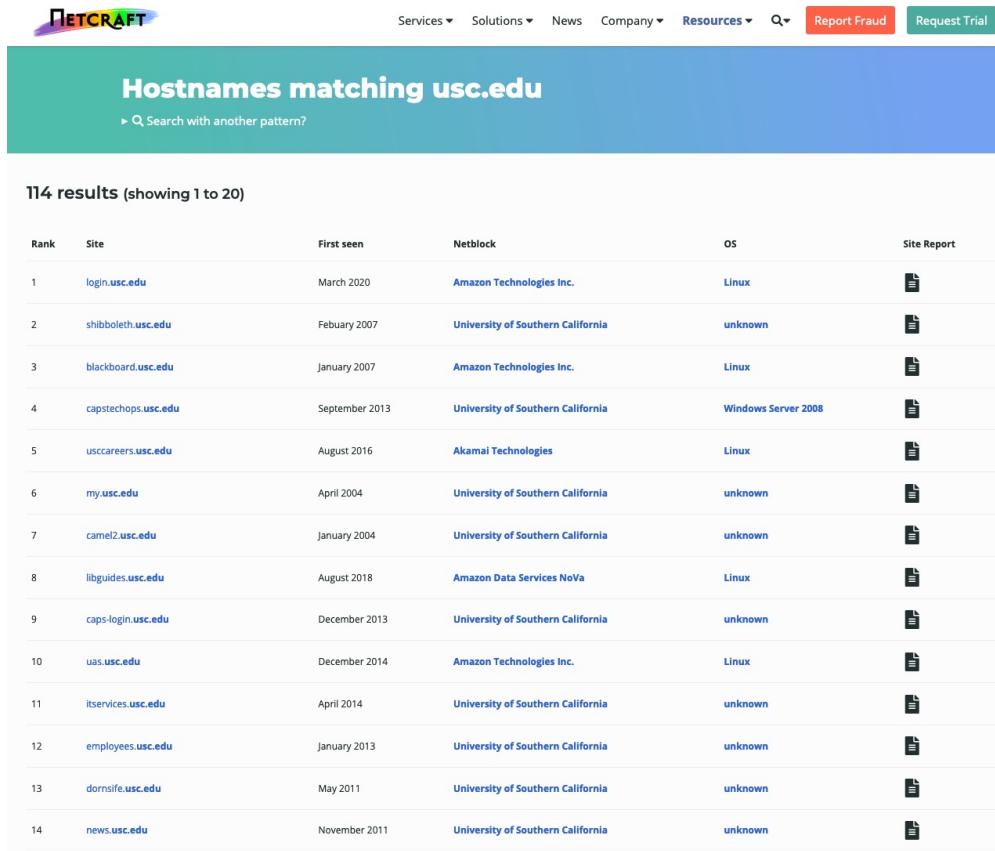
Google uses software-defined networking technology to route packets across the globe and enable fast edge-caching so that data is where it needs to be to serve users.

Serverless Architecture

- Internet based systems where the application development does not use the usual server process.
- They rely solely on a combination of:
 - third-party services, or Backend as a Service (BaaS)
 - client-side logic
 - service hosted remote procedure calls, or Function as a Service (FaaS).
- AWS Lambda is one of the most popular implementations of FaaS at present, but there are others. See:
<https://aws.amazon.com/lambda/>
- Serverless and contains will be covered later in the course



USC Has Many Web Servers Running



The screenshot shows a search results page from Netcraft. The header includes the Netcraft logo, navigation links for Services, Solutions, News, Company, Resources, a search bar, a Report Fraud button, and a Request Trial button. The main title is "Hostnames matching usc.edu". Below it is a search bar with placeholder text "Search with another pattern?". A subtitle indicates "114 results (showing 1 to 20)". The results table has columns: Rank, Site, First seen, Netblock, OS, and Site Report. The data shows 14 entries, all from the University of Southern California, with various first seen dates and netblocks like Amazon Technologies Inc., University of Southern California, and Akamai Technologies.

Rank	Site	First seen	Netblock	OS	Site Report
1	login.usc.edu	March 2020	Amazon Technologies Inc.	Linux	Report
2	shibboleth.usc.edu	February 2007	University of Southern California	unknown	Report
3	blackboard.usc.edu	January 2007	Amazon Technologies Inc.	Linux	Report
4	capstechops.usc.edu	September 2013	University of Southern California	Windows Server 2008	Report
5	usccareers.usc.edu	August 2016	Akamai Technologies	Linux	Report
6	my.usc.edu	April 2004	University of Southern California	unknown	Report
7	camel2.usc.edu	January 2004	University of Southern California	unknown	Report
8	libguides.usc.edu	August 2018	Amazon Data Services NoVa	Linux	Report
9	caps-login.usc.edu	December 2013	University of Southern California	unknown	Report
10	uas.usc.edu	December 2014	Amazon Technologies Inc.	Linux	Report
11	itservices.usc.edu	April 2014	University of Southern California	unknown	Report
12	employees.usc.edu	January 2013	University of Southern California	unknown	Report
13	dornsife.usc.edu	May 2011	University of Southern California	unknown	Report
14	news.usc.edu	November 2011	University of Southern California	unknown	Report

- Netcraft lists **106** separate sites / web servers with usc.edu in their name, e.g.
- www.usc.edu
- mat.usc.edu
- www.cs.usc.edu
- dornsife.usc.edu
- web-applusc.edu
- www-scf.usc.edu
- However, some may not be connected to USC, e.g.
- www.usc.edu.au
- Check at:
<https://searchdns.netcraft.com>

Web Browsers Use Standard Layout Engines

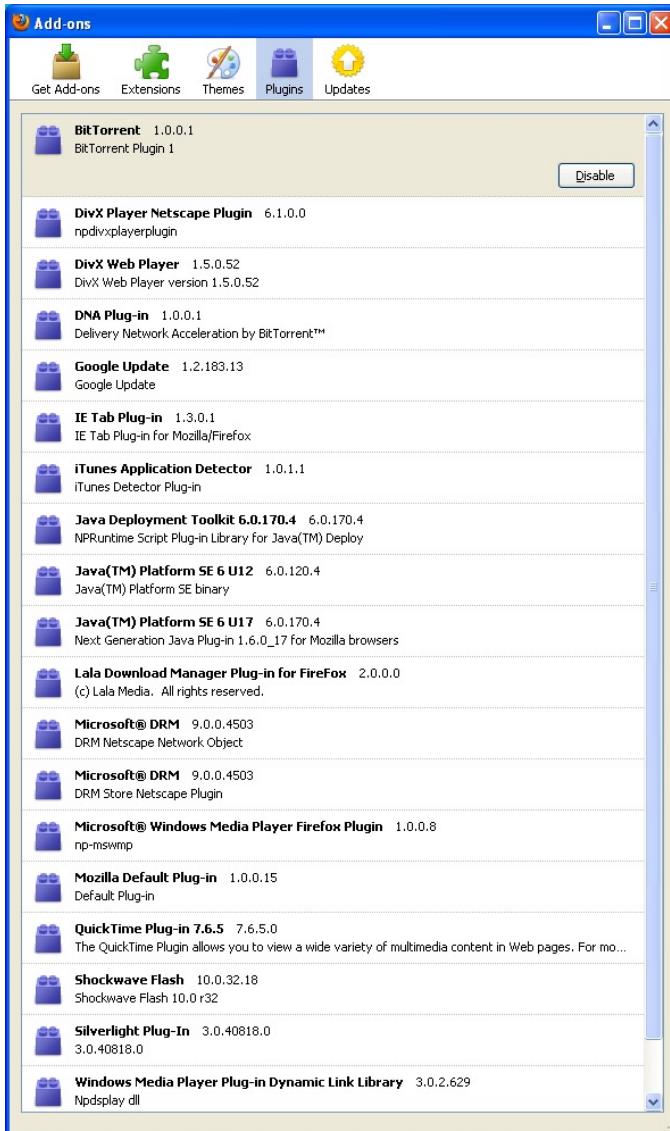
- **WebKit** is a software component used to render web pages; it is open source.
 - It is used by Google's Chrome and Apple's Safari web browsers
 - WebKit is also the name of the Mac OS X system framework version of the engine that's used by Safari, Dashboard, Mail, and many other OS X applications;
- **Gecko** is a layout engine developed by Mozilla Corporation, known as the layout engine of the Firefox web browser.
 - It is used to display web pages and, in some cases, an application's user interface.
 - It offers a rich programming API that makes it suitable for a wide variety of roles in Internet-enabled applications, such as web browsers
 - Its development originated with Netscape Communications Corporation
- Some web kits and the browsers that use them
 - **Gecko-based:** FireFox (Mozilla), Flock, Netscape
 - **Trident-shells:** Internet Explorer (Microsoft, retired)
 - **EdgeHTML:** Edge Legacy (Microsoft), fork of Trident 7, retired in Aug. 2021
 - **WebKit-based:** Chrome and Android (Google), Midori, Safari and Mobile Safari (Apple), Symbian^3 (Nokia) and many others
 - **Chromium-based:** Chrome, Microsoft Edge (2021)
 - **Presto-based:** Opera, Nintendo DS, Opera Mini, Opera Mobile
 - **Java-based:** HotJava, Lobo

Capabilities of a Browser

- Web browsers fetch and display documents from other WWW sites; their capabilities include:
 - A mouse-driven graphical user interface
 - Display of
 - Hypertext documents conforming to latest HTML standard
 - Text with fonts, styles, and varying point sizes
 - Foreign-language character sets conforming to ISO-8859
 - Forms composed of edit boxes, check boxes, radio boxes, lists, text areas, etc.
 - Graphics in different formats (GIF, JPEG, MPEG, PNG, XBM) including monochrome, color

GIF = graphic interchange format, MPEG = Motion Picture Experts Group, JPEG = Joint Photographic Experts Group, PNG = Portable Network Graphics, XBM = x bitmap

Capabilities of a Browser



- Ability to invoke helper applications and plug-ins, (**Obsoleted in HTML5**) e.g.
 - *Adobe Acrobat* - used to view pdf files
 - *Windows Media Player* to play digital sound files
 - *Adobe Flash Player*, used to display video. **Retired in 2020.**)
- Ability to communicate over a secure channel, using SSL
- Ability to maintain and exchange digital certificates
- Ability to run scripts in JavaScript
- Ability to run *Java applets* and *Active X components* (**also obsoleted in HTML5**)

The Browser Wars - Desktop Statistics

2021	Chrome	Edge	Firefox	Safari	Opera
November	80.0 %	6.8 %	5.8 %	3.9 %	2.4 %
October	80.3 %	6.7 %	5.7 %	3.9 %	2.3 %
September	80.9 %	6.5 %	5.6 %	3.6 %	2.2 %
August	81.4 %	6.1 %	5.6 %	3.3 %	2.1 %
July	81.6 %	6.0 %	5.6 %	3.3 %	2.2 %
June	81.7 %	5.9 %	5.6 %	3.4 %	2.2 %
May	81.2 %	5.8 %	5.8 %	3.5 %	2.4 %
April	80.7 %	5.6 %	6.1 %	3.7 %	2.4 %
March	80.8 %	5.5 %	6.3 %	3.7 %	2.3 %
February	80.6 %	5.4 %	6.6 %	3.9 %	2.3 %
January	80.3 %	5.3 %	6.7 %	3.8 %	2.3 %
2020	Chrome	Edge/IE	Firefox	Safari	Opera
December	80.5 %	5.2 %	6.7 %	3.7 %	2.3 %
November	80.0 %	5.3 %	7.1 %	3.9 %	2.3 %
October	80.4 %	5.2 %	7.1 %	3.7 %	2.1 %
September	81.0 %	4.9 %	7.2 %	3.6 %	2.0 %
August	81.2 %	4.6 %	7.3 %	3.4 %	2.0 %
July	81.3 %	4.3 %	7.6 %	3.4 %	2.0 %
June	80.7 %	3.9 %	8.1 %	3.7 %	2.1 %
May	80.7 %	3.5 %	8.5 %	4.1 %	1.6 %
April	80.7 %	3.4 %	8.6 %	4.2 %	1.5 %
March	81.4 %	3.5 %	8.7 %	3.7 %	1.3 %
February	82.0 %	3.4 %	8.7 %	3.4 %	1.2 %
January	81.9 %	3.0 %	9.1 %	3.3 %	1.3 %

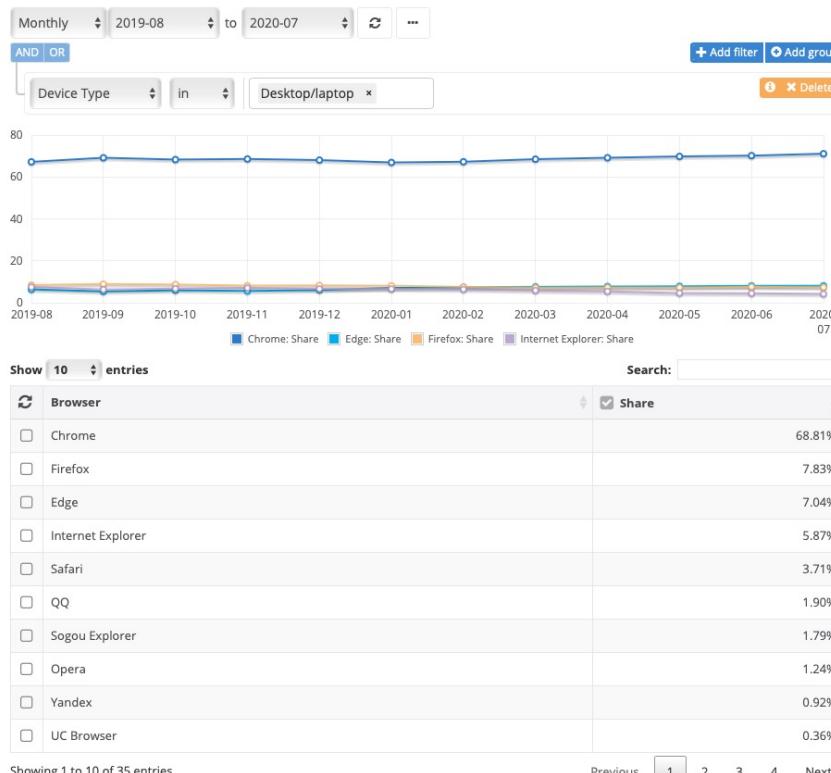
- See http://www.w3schools.com/browsers/browsers_stats.asp
- See also <http://www.upsdell.com/BrowserNews/stat.htm>

Conclusion of the above study:

- Chrome is the clear winner
- Firefox comes second, but losing ground
- Edge next
- Safari and Opera having small percentages
- WebKit total over 84%

Desktop/laptop Browser Market Share Statistics

Browser Market Share

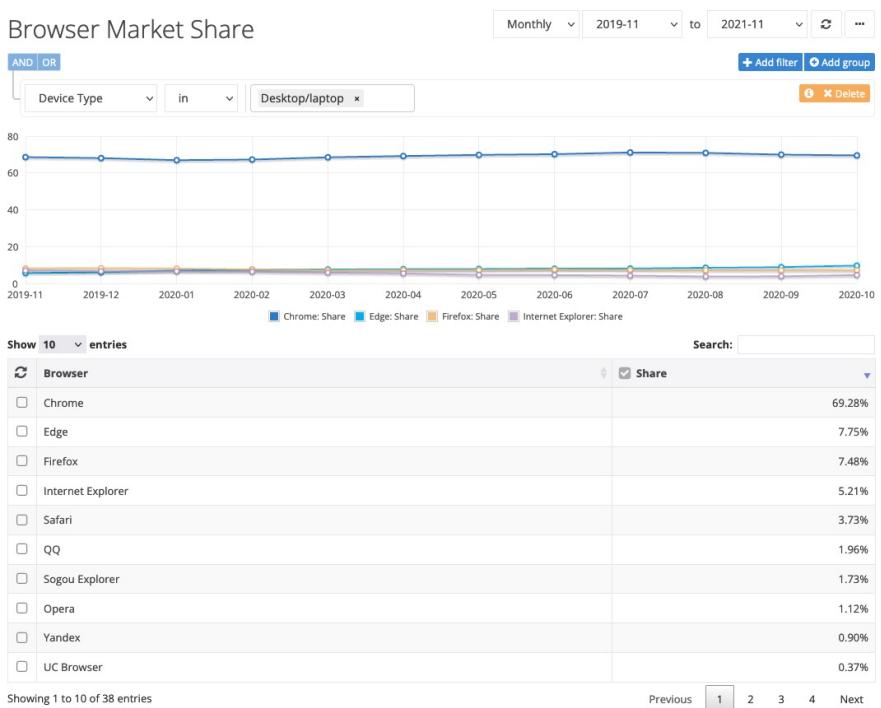


August 2020

<http://www.netmarketshare.com/>

Chrome leads with 69% market share.

Browser Market Share



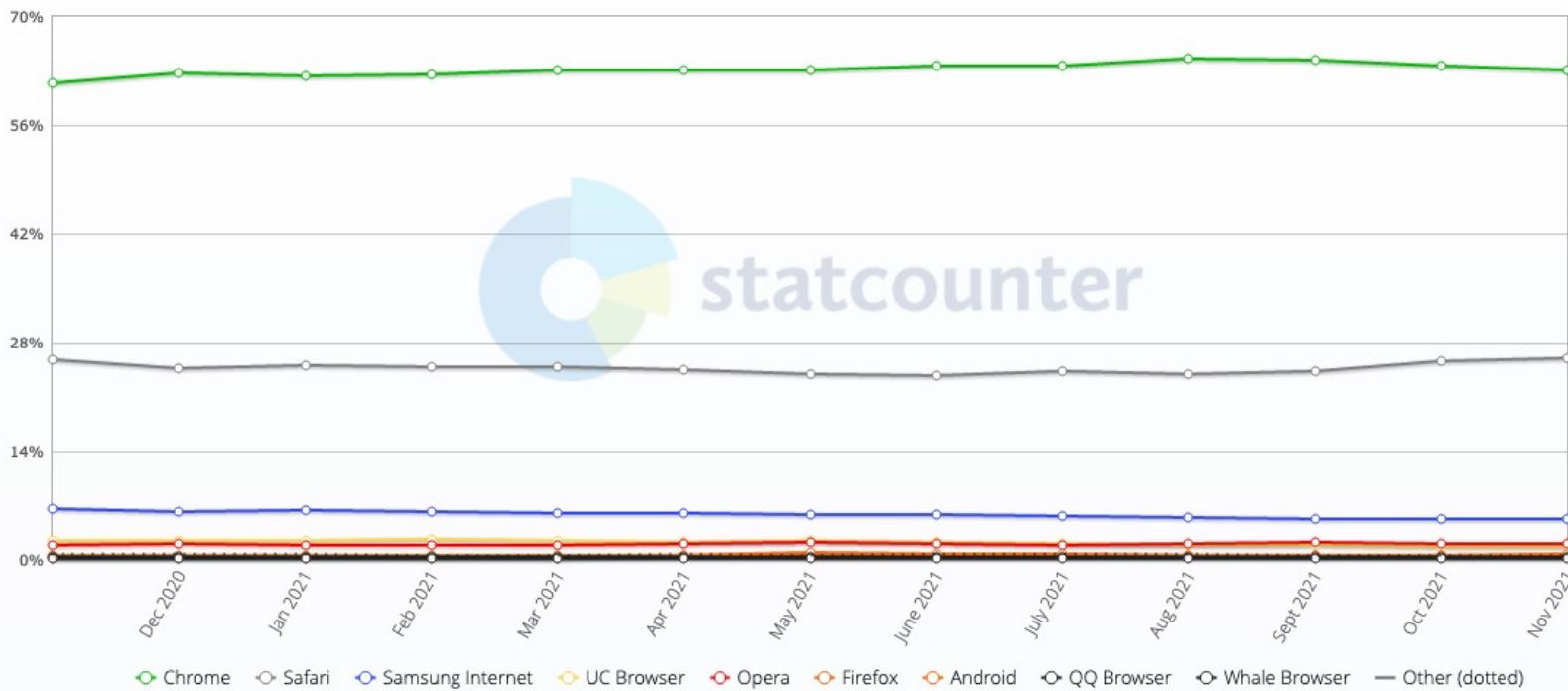
December 2021

The Browser Wars Comparison (cont'd)

Mobile Browser Market Share Worldwide

Nov 2020 - Nov 2021

Edit Chart Data



Save Chart Image (.png)

Download Data (.csv)

Embed HTML

```
<div id="mobile_browser-ww-monthly-202011-202111" width="600" he
```

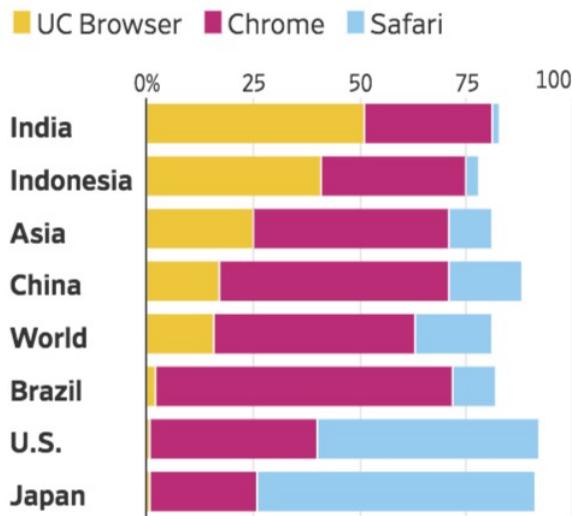
StatCounter Global Stats, May 2020 – April 2021, See <http://gs.statcounter.com>
Chrome has the lead with about 60%, followed by Safari at 24%, Samsung at 6%
and UC Browser from UCWeb of Alibaba Group of China at 3%.

The Browser Wars Comparison (cont'd)

Browser for the Next Billion

Alibaba's mobile browser, UC Browser, has a larger market share than Google's Chrome in India and Indonesia, where many of the world's 'next billion users' are getting online for the first time.

Mobile web browser market share



Note: Data Oct.-Dec. 2016 through Oct.-Dec. 2017.

Source: StatCounter

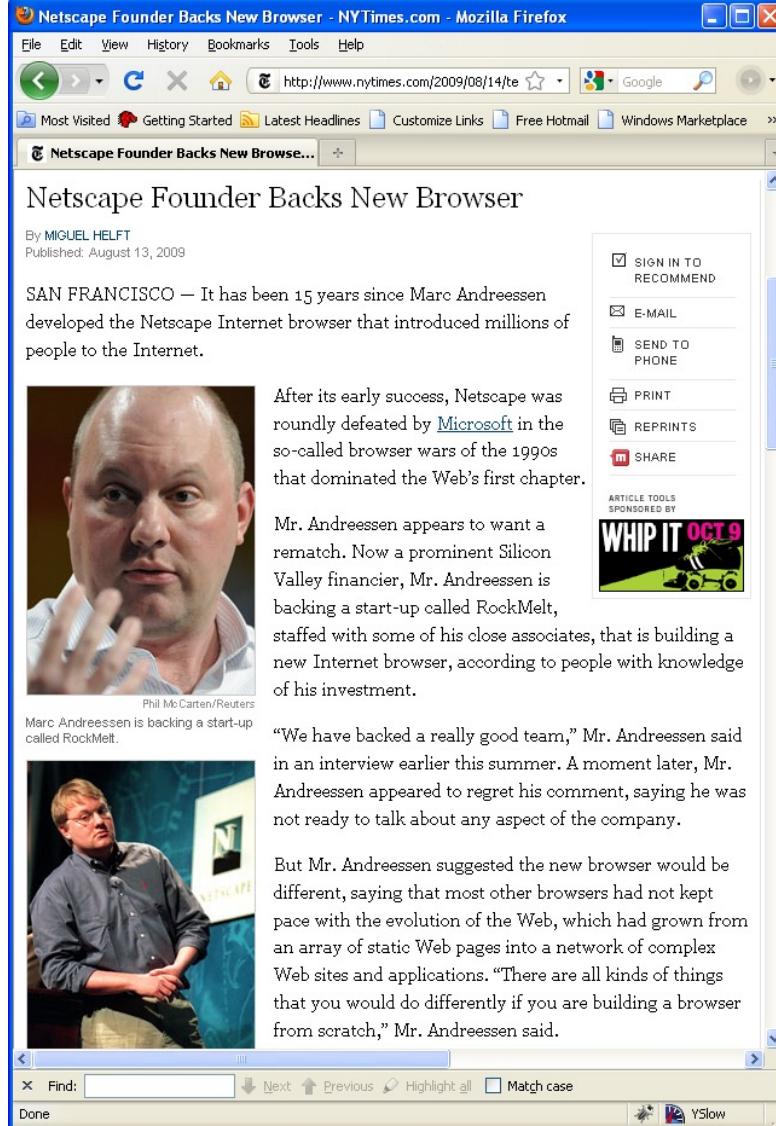
StatCounter Global Stats, Oct.-Dec. 2016 through Oct.-Dec. 2017, See
<https://www.wsj.com/articles/a-browser-youve-never-heard-of-is-dethroning-google-in-asia-1514808002>

Browsers are the Gateway to the Web/Internet

Despite Netscape's failure, there is now a new business model for browsers; Google will pay Mozilla \$300 million/year for 3 years to keep Google its default search engine. (Dec. 25, 2011)

Similarly, Google is rumored to have been paying Apple \$1 billion in 2011 to keep Google the default search engine for Safari on iOS devices and OS X.
http://articles.businessinsider.com/2012-03-09/tech/31138467_1_google-maps-ben-schachter-google-searches

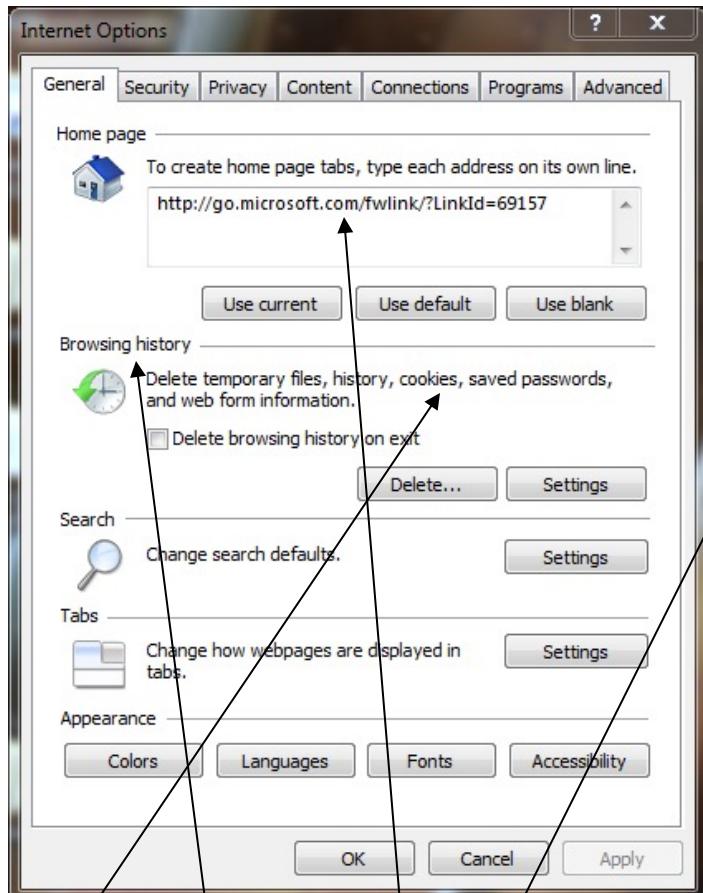
June, 2014: Apple announces DuckDuckGo will be another built-in search engine on Safari (no user tracking)



The screenshot shows a Mozilla Firefox window with the title bar "Netscape Founder Backs New Browser - NYTimes.com - Mozilla Firefox". The address bar shows the URL "http://www.nytimes.com/2009/08/14/technology/internet/14browser.html?_r=1&scp=1&sq=browser%20wars&st=cse". The main content area displays an article titled "Netscape Founder Backs New Browser" by MIGUEL HELFT, published on August 13, 2009. The article discusses Marc Andreessen's support for RockMelt, a new browser he is backing. It includes a photo of Andreessen and a quote from him. On the right side of the browser window, there is a sidebar with social sharing options like "SIGN IN TO RECOMMEND", "E-MAIL", "SEND TO PHONE", "PRINT", "REPRINTS", and "SHARE". Below the sidebar, there is an advertisement for "WHIP IT OCT 9". The bottom of the browser window shows standard navigation and search controls.

Browser Options Menus for IE and Firefox

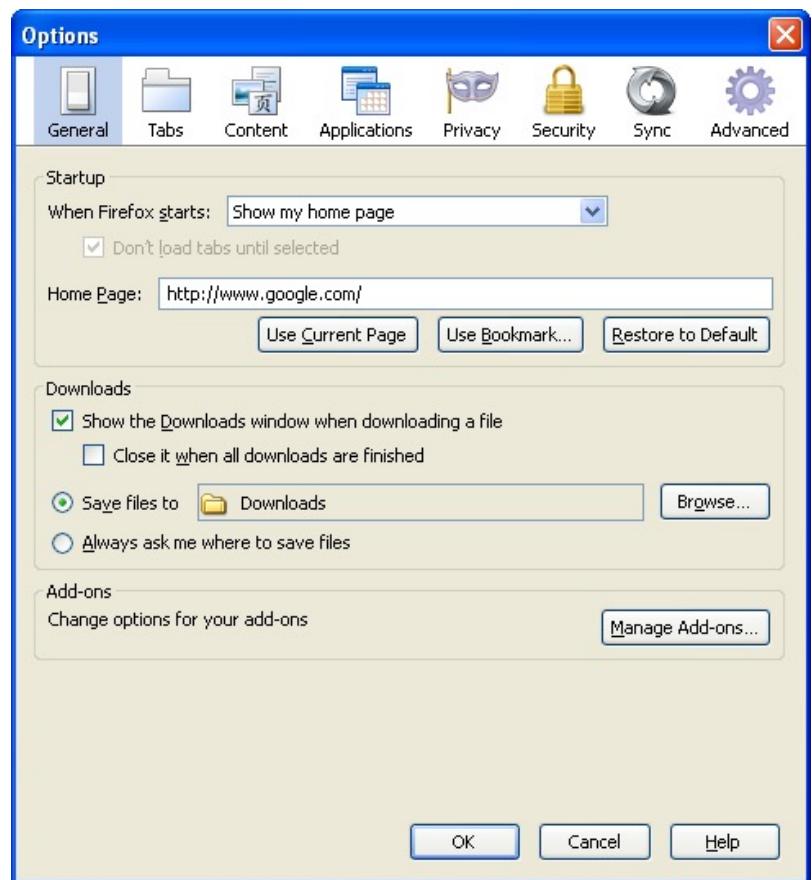
Click on Tools -> Internet Options



Internet Explorer

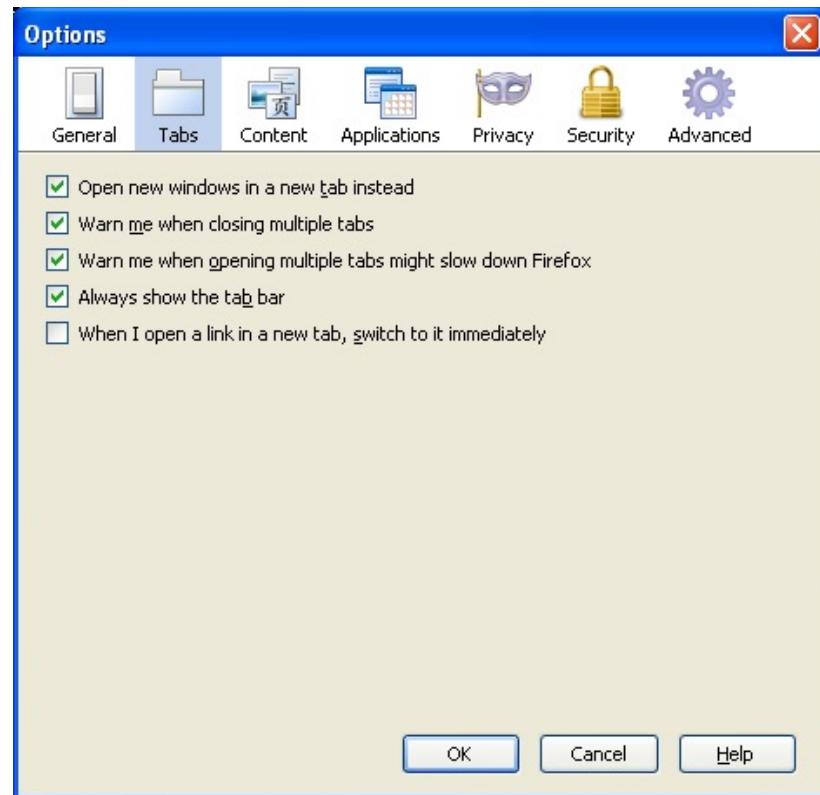
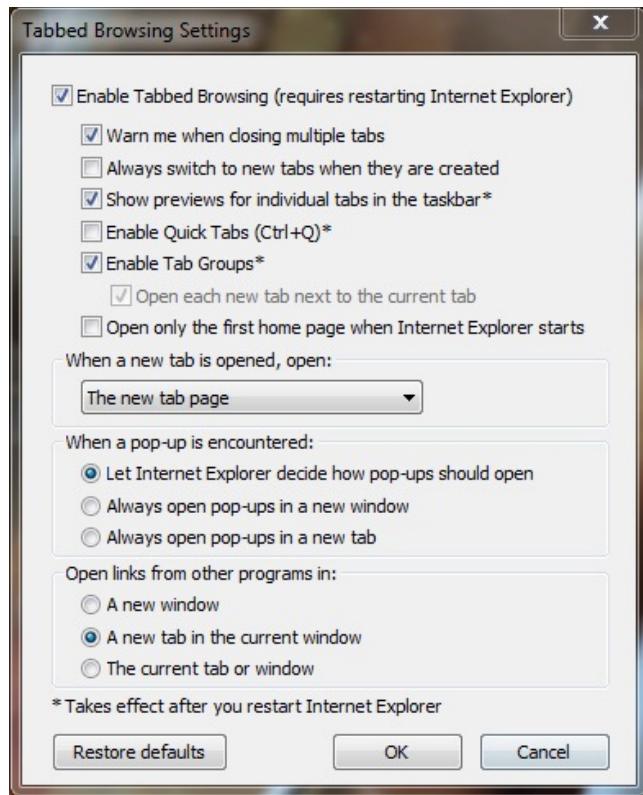
Cookies, History, default opening page

Click on Tools -> Options



Firefox – Tools | Options

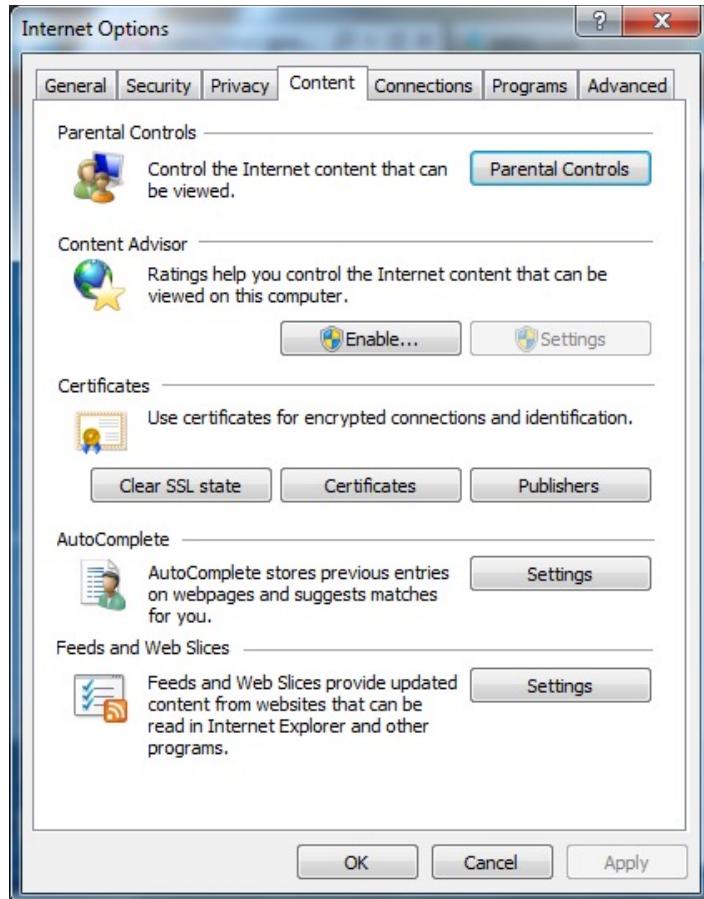
IE and Firefox Tab Control Options



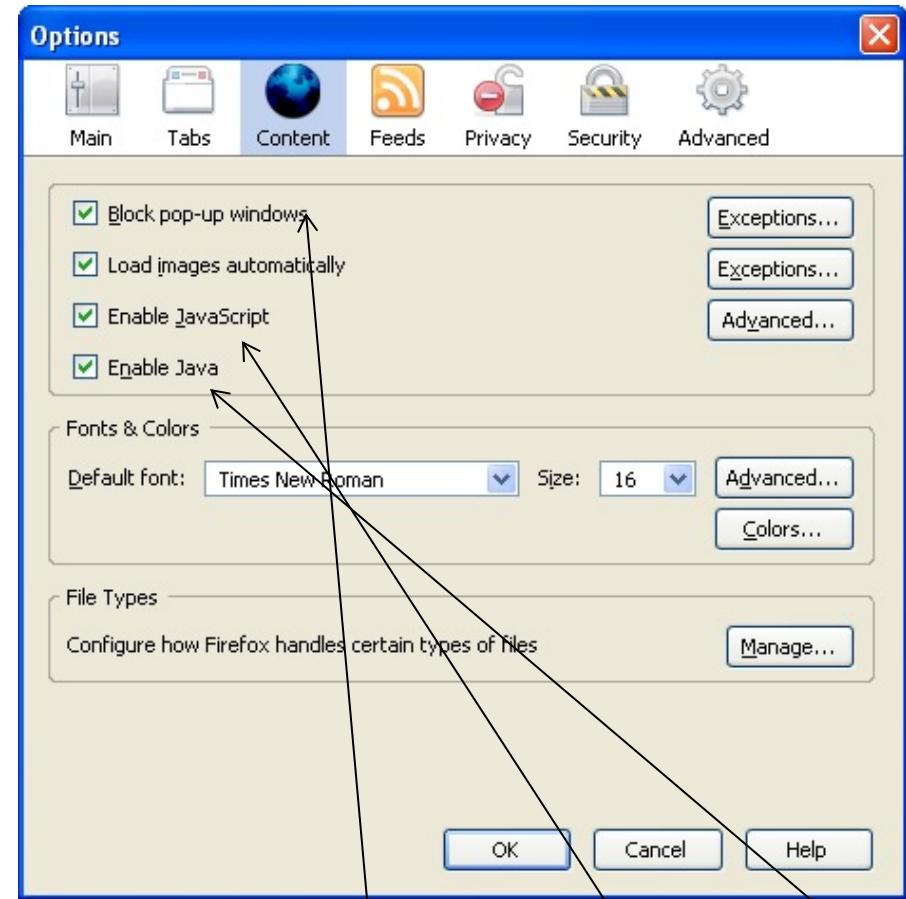
Internet Explorer

Firefox

IE and Firefox – Content Options

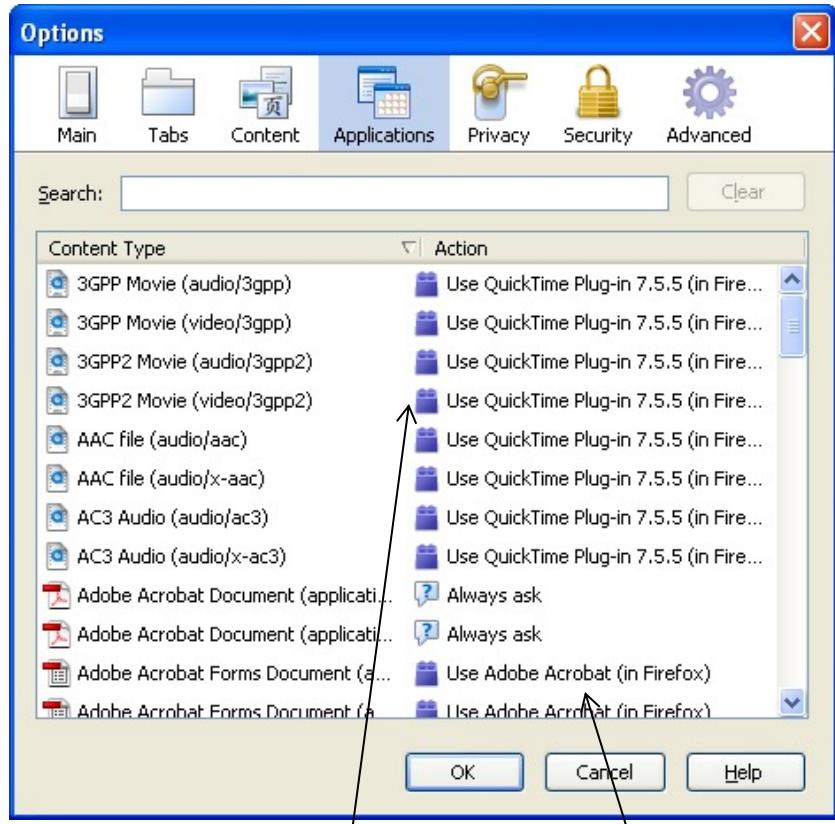
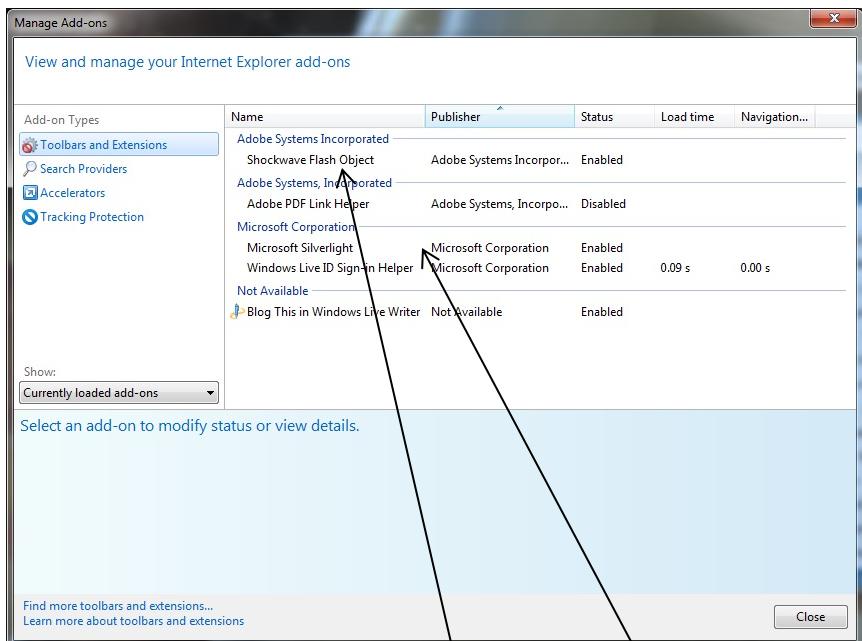


Internet Explorer



Firefox (PopUps, JavaScript, Java)

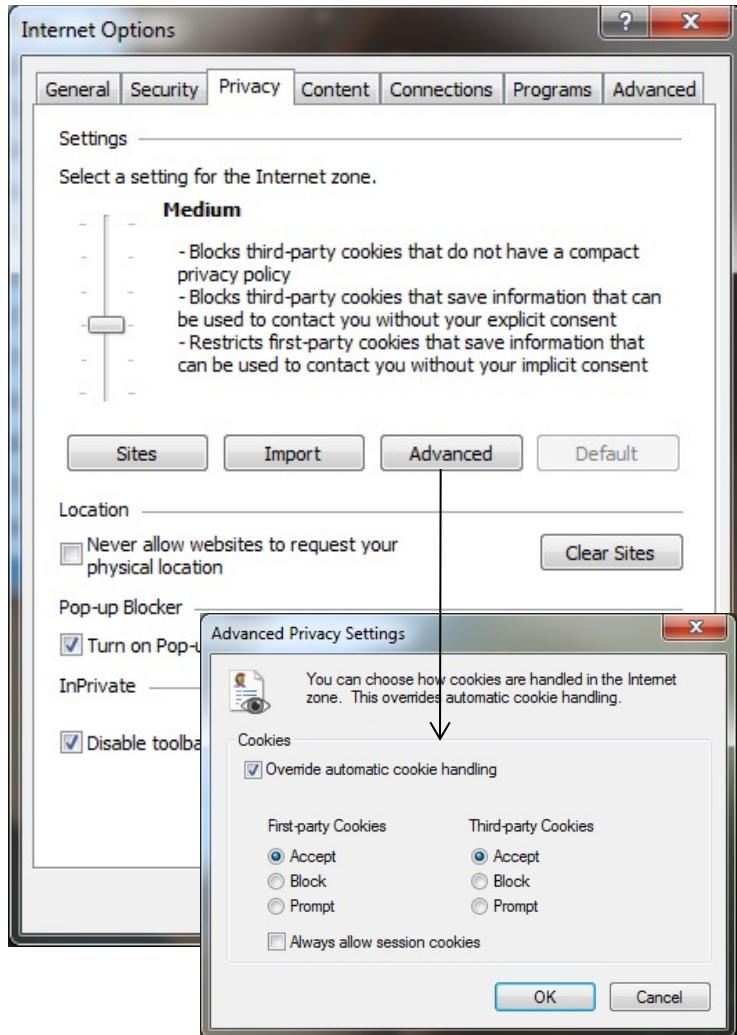
IE and Firefox Applications Options



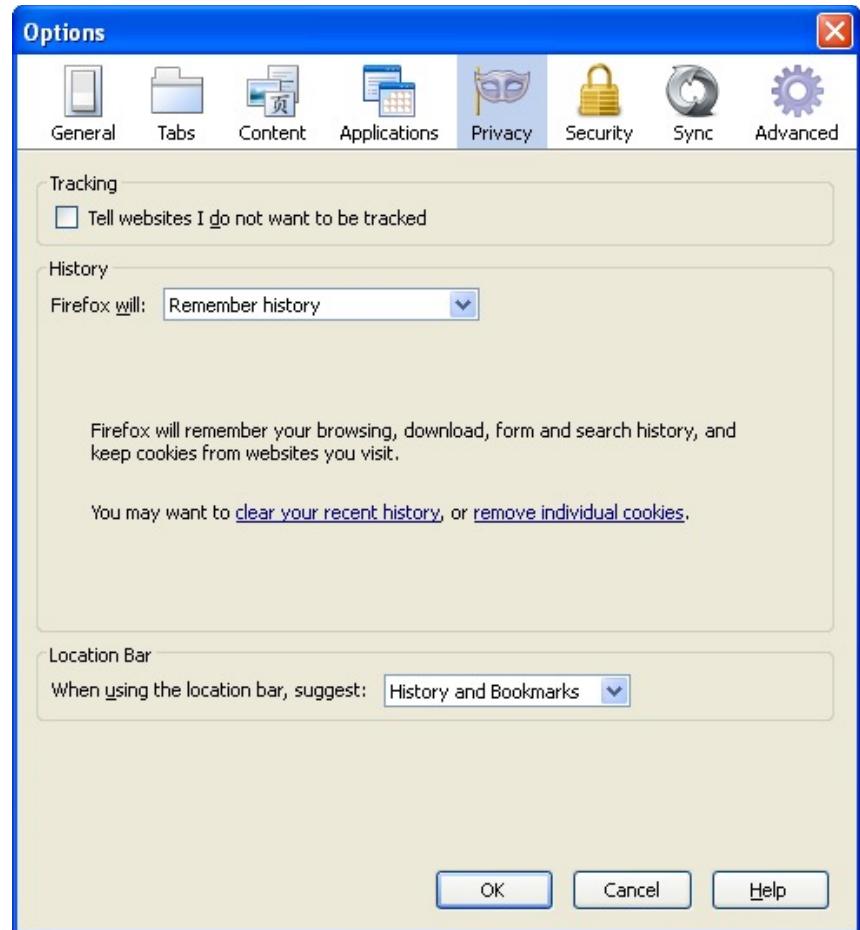
Internet Explorer (Adobe, Silverlight)
(Microsoft has now dropped Silverlight)

Firefox (Quicktime, Acrobat)

IE and Firefox Privacy Options

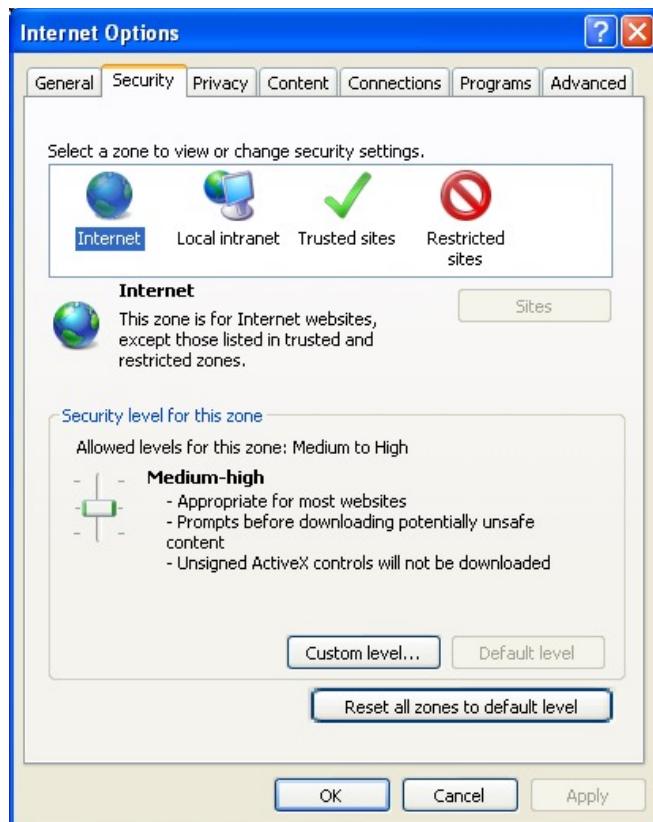


Internet Explorer (Cookies)

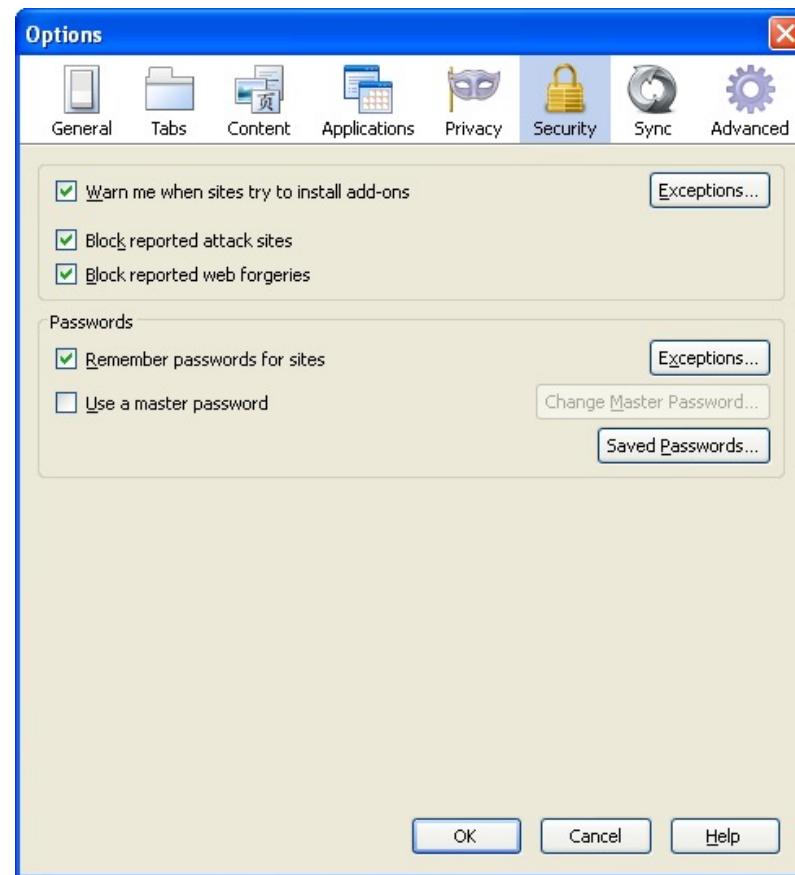


Firefox (History, Cookies)

IE and Firefox Security Options

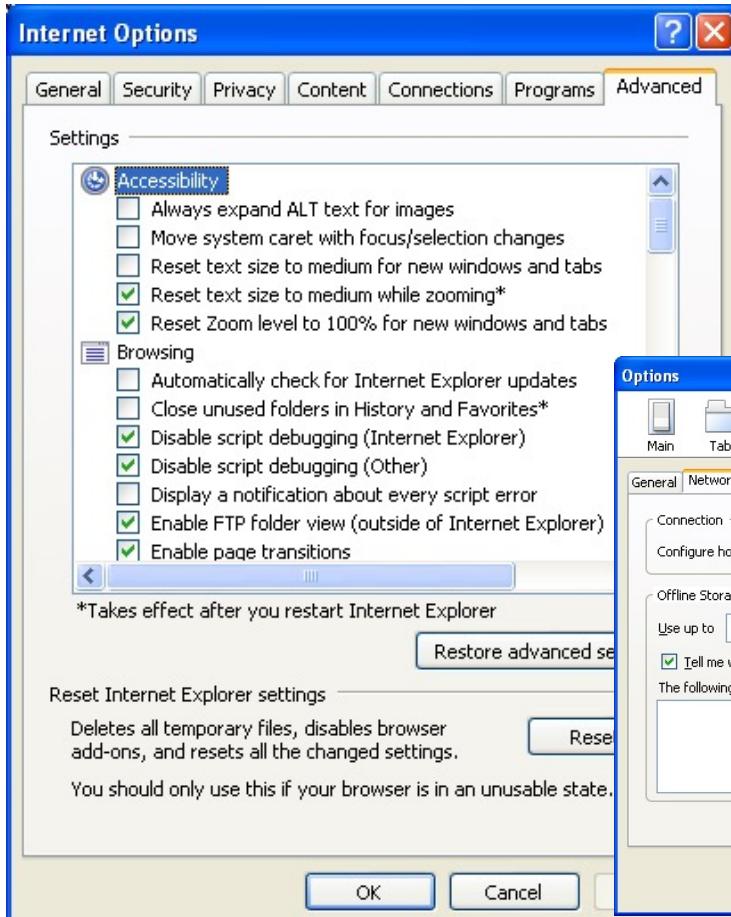


Internet Explorer



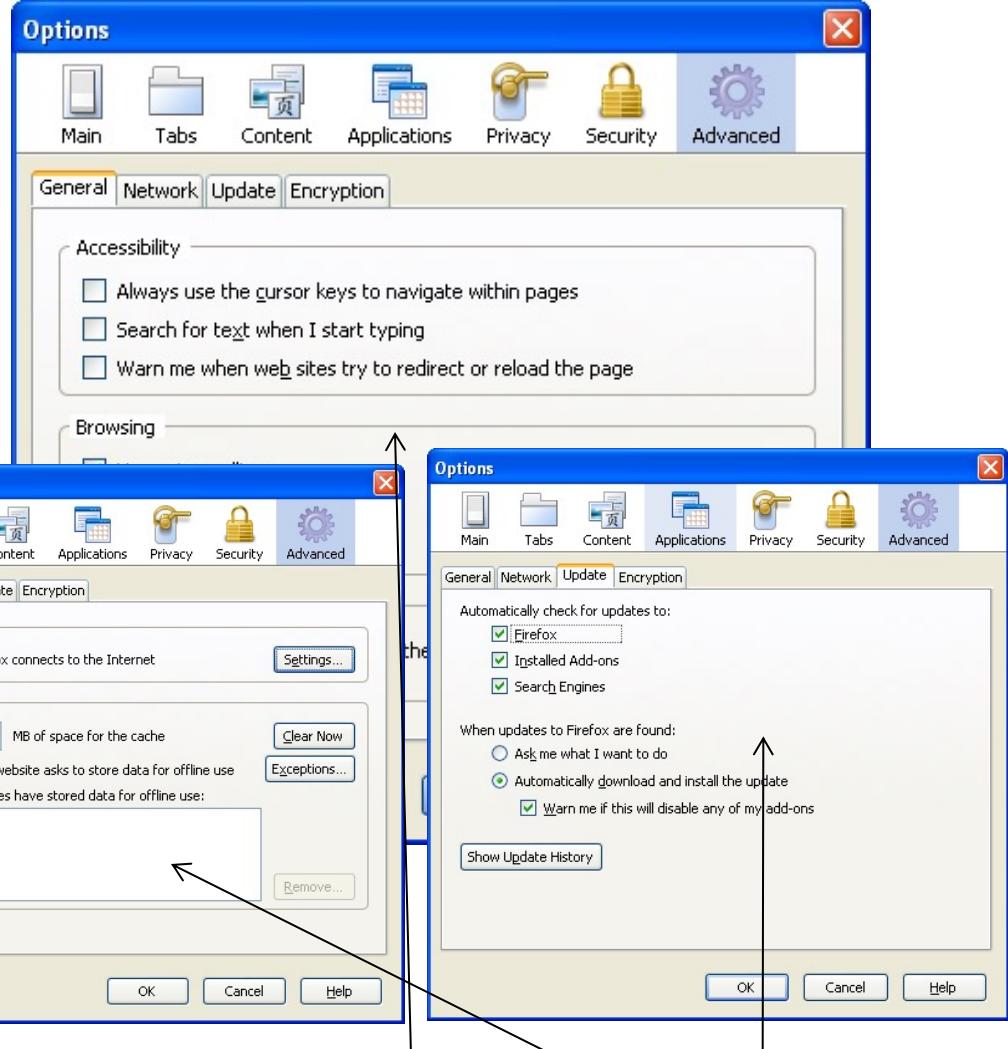
Firefox

IE and Firefox Advanced Options



Internet Explorer

Copyright © Ellis Horowitz & Marco Papa
1999-2022

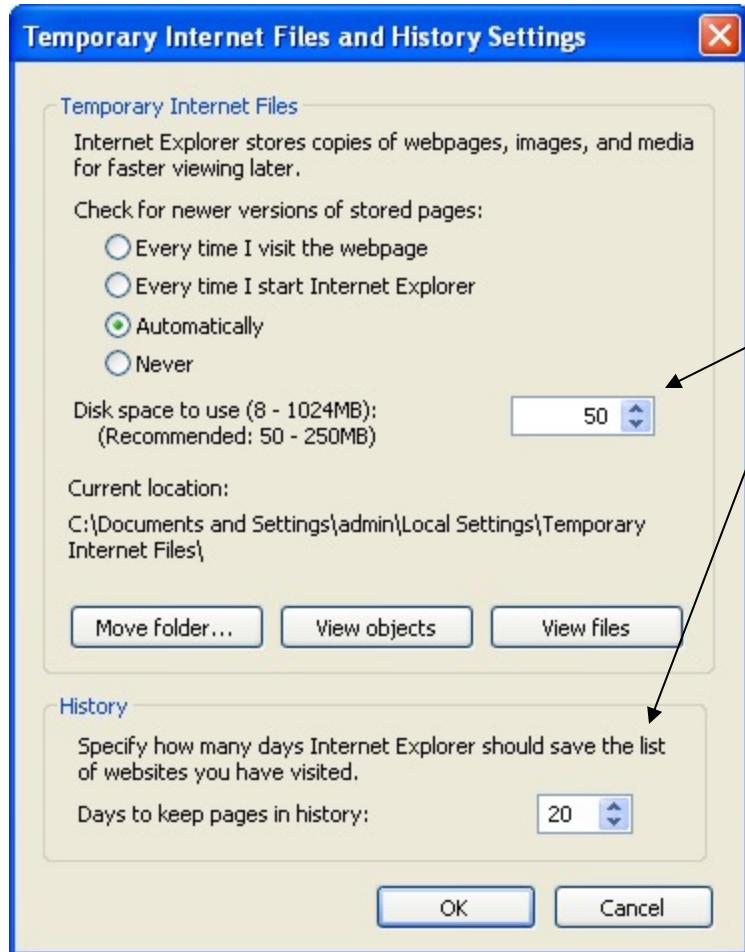


Firefox (General, Network, Update)

Course Intro

41

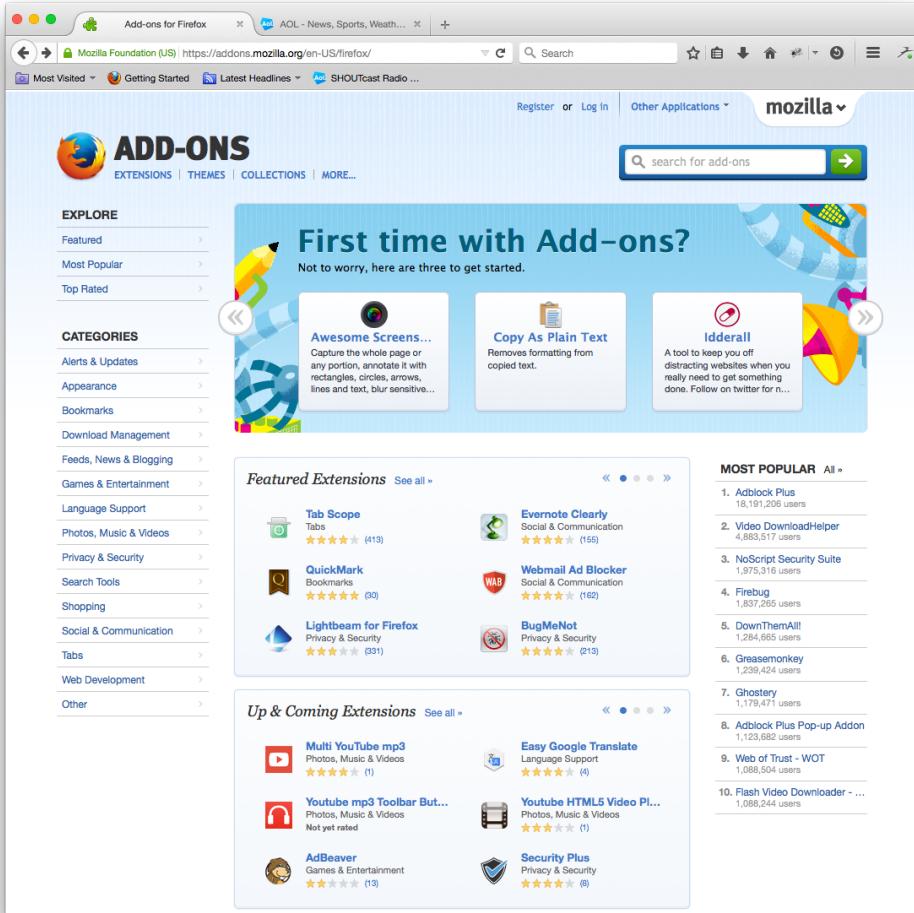
Internet Explorer Browser Caching



- History
 - Links and URLs that have been accessed by the browser over a period of time
- Disk cache
 - Temporary internet files, a folder on the disk that contains cached copies of files
- Memory cache
 - Session-based information that is cached during the session
- Offline content
 - Web content is downloaded when online and viewed offline

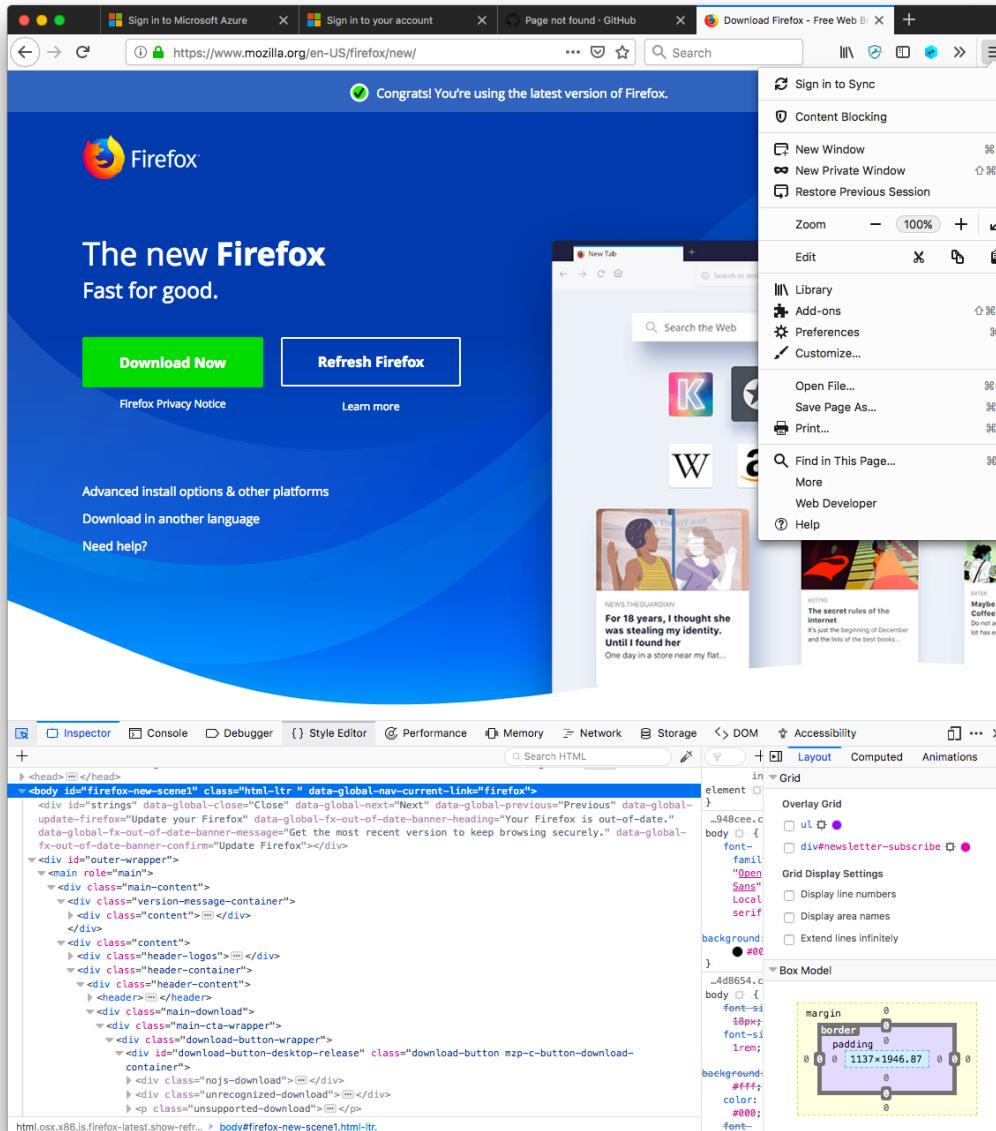
IE caching options screen (Tools | Options | General | Browsing History)

Browsers Have Many Plugins Available

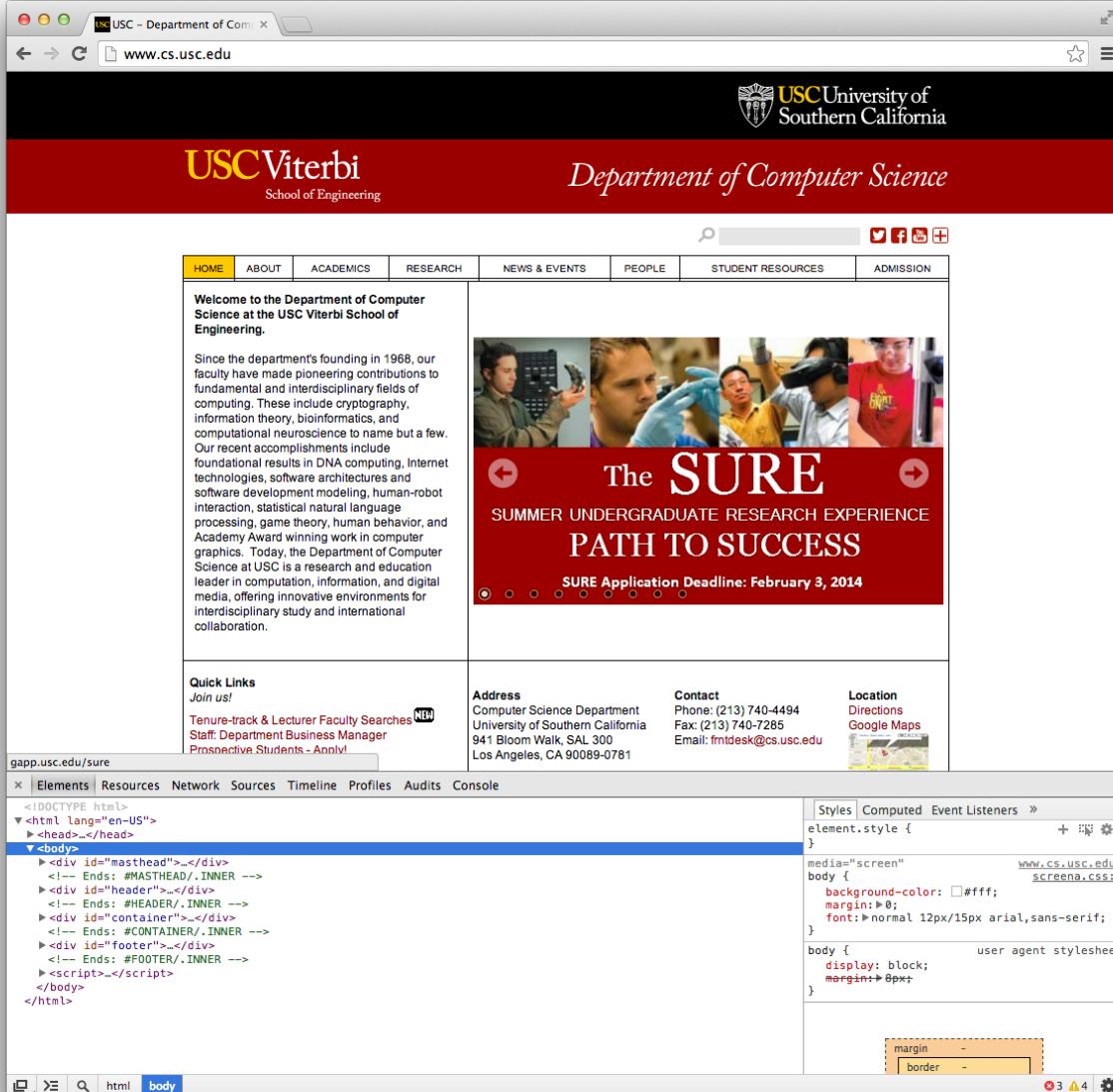


- Plug-ins are also listed as “Extensions”
- Firefox plug-ins that will be especially useful in this course are:
 - HTTP Header Live
- HTML5 does away with most video / audio plug-ins

Firefox: Tools | Web Developer



Chrome: Menu | More Tools | Developer Tools



Evolution of Web Sites

Client-centric Static	Server Applications	Web services	Service Oriented Arch.	Multi-platform (desktop, tablet, phone)	IoT, Wearables, Cloud computing, Serverless Arch. (BaaS, FaaS) Docker
	Databases	Multiple layers	(SOA)	Client-centric	
	Dynamic web pages	Business and service Integration		Client-centric	
HTML Scripts CGI	ODBC, JDBC ASP Applets, ActiveX	XML, WML, SQL, .NET COM+, Beans	Ajax, Web 2.0, JSON	HTML5, CSS3, JS gestures navigation	JS Frameworks AWS, GCP, Azure Microservices containers
1 st gen	2 nd gen	3 rd gen	4 th gen	5 th gen	6 th gen
1991	1997	2000	2005	2008	2014

HTML

What is HTML?

- HTML stands for **Hypertext Markup Language**
 - It is a **markup language**, which means it is used for processing, definition, and display of data
- HTML is a tag-based language, similar to XML
 - HTML tags are already defined though

Version	Year
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
HTML5	2014

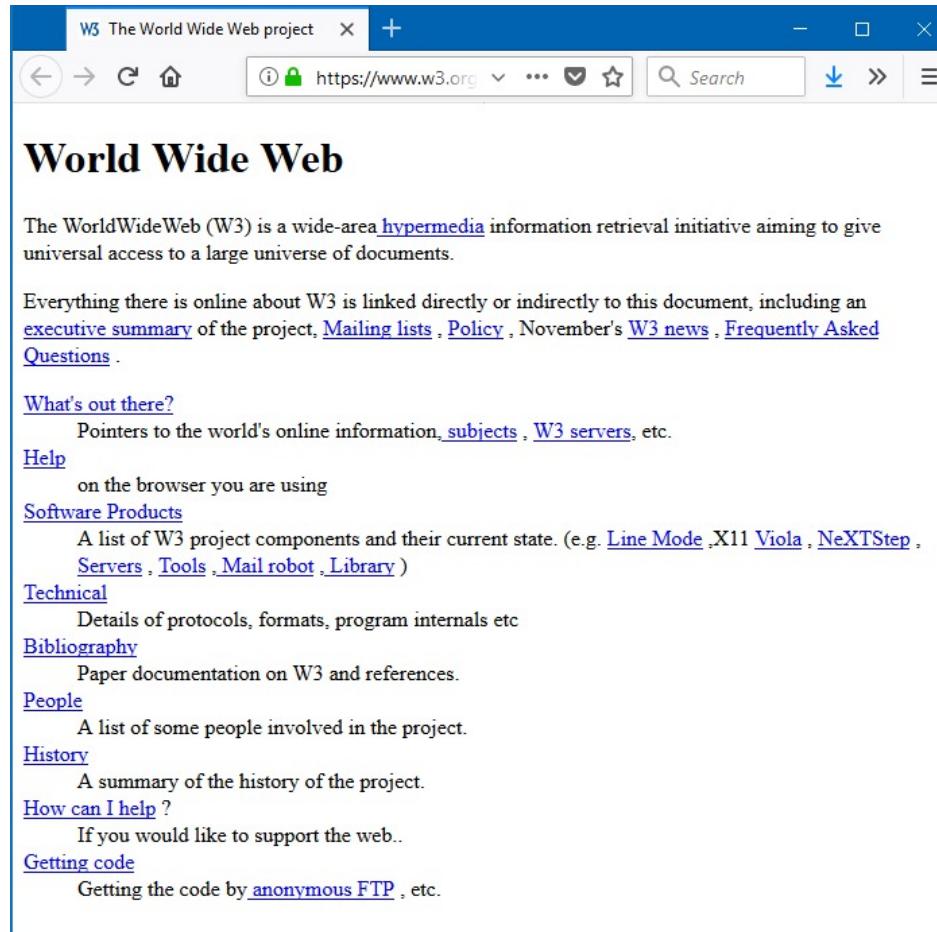


What is HTML? (cont'd)

- Hypertext Markup Language (HTML) is a language in which one can describe:
 - The display and format of text
 - The display of graphics
 - Pointers to other html files
 - Pointers to files containing graphics, digitized video and sound
 - Forms that capture information from the viewer
- HTML was developed by **Tim Berners-Lee** of CERN around 1990
- HTML is understood by WWW browsers—e.g., Internet Explorer, Firefox, Chrome, Safari, many others — which interpret and display the output to the viewer

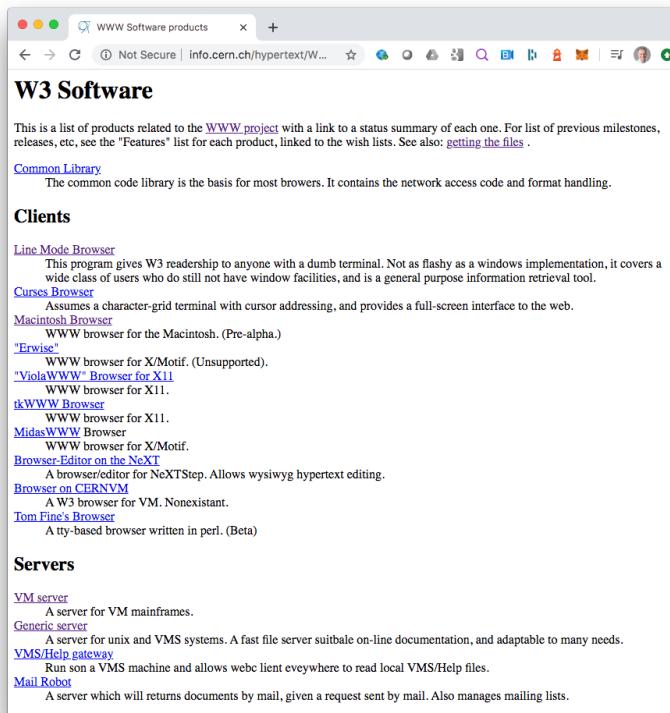
First Web Page

- The first web page ever displayed by Tim Berners-Lee at CERN



First Web Site

- The original web site of the “WWW Project” is still available at CERN
- <http://info.cern.ch/hypertext/www/TheProject.html>
- Includes documentation on the original browsers and servers



World Wide Web Motivation

- Originally created in 1989 at **CERN** (the European Organization for Nuclear Research) for sharing research in a faster manner than journal publications allowed

930430

ORGANISATION EUROPEENNE POUR LA RECHERCHE NUCLEAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

STATEMENT CONCERNING CERN W3 SOFTWARE RELEASE INTO PUBLIC
DOMAIN

TO WHOM IT MAY CONCERN

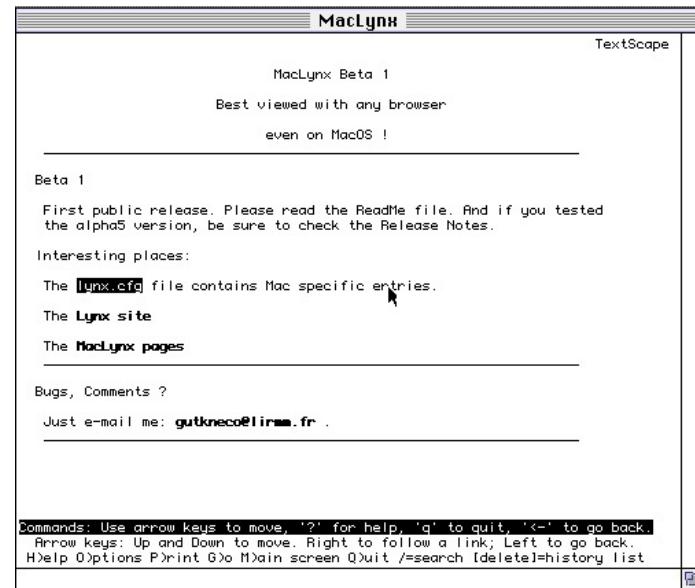
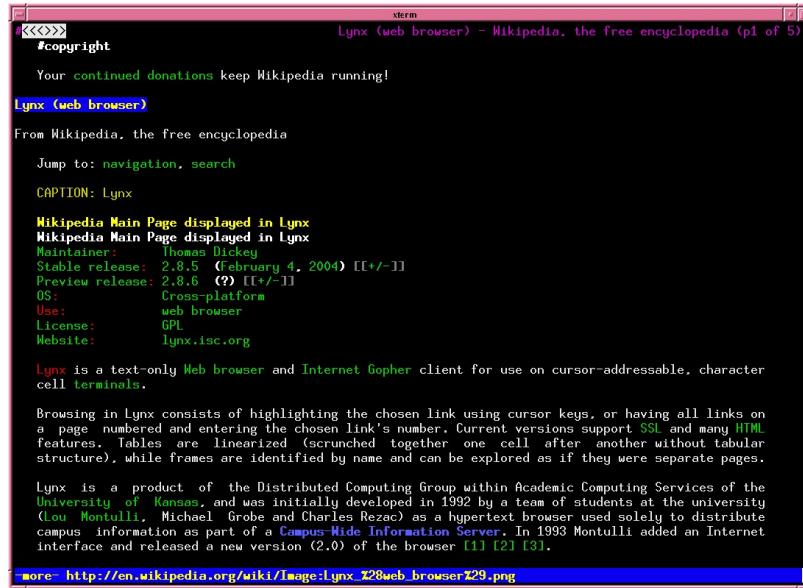
Introduction

The World Wide Web, hereafter referred to as W3, is a global computer networked information system.

The W3 project provides a collaborative information system independent of hardware and software platform, and physical location. The project spans technical design notes, documentation, news, discussion, educational material, personal notes, publicity, bulletin boards, live status information and numerical data as a uniform continuum, seamlessly intergrated with similar information in other disciplines.

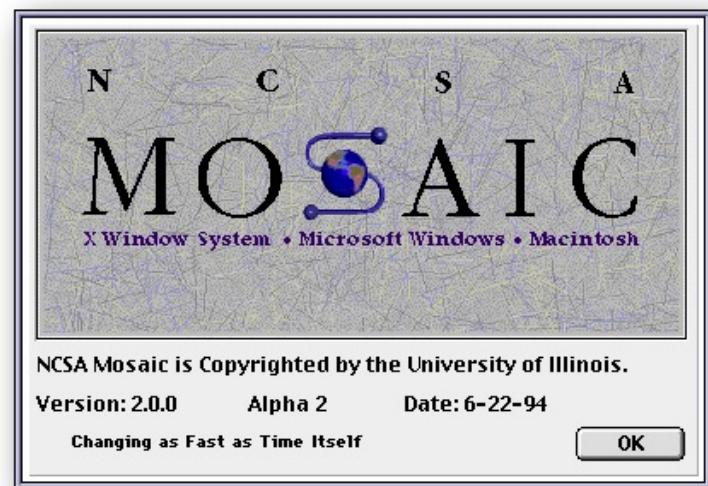
Text Browser – Lynx

- Developed by Distributed Computing Group within Academic Computing Services of the University of Kansas
- Team of students and staff with initial release in 1992
- [https://en.wikipedia.org/wiki/Lynx_\(web_browser\)](https://en.wikipedia.org/wiki/Lynx_(web_browser))



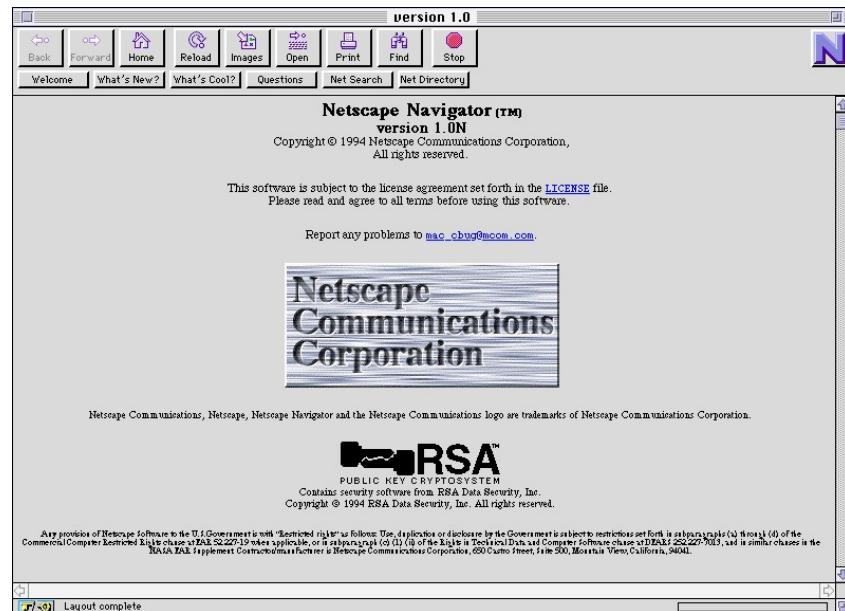
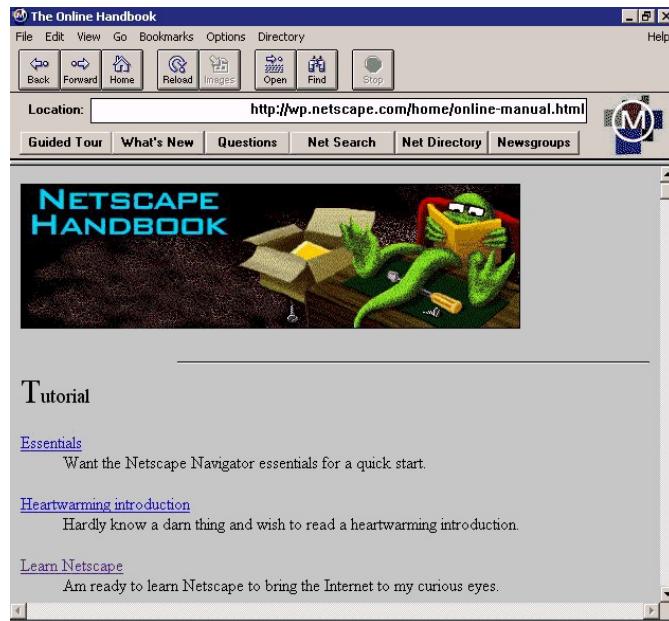
First Graphical Browser – NCSA Mosaic

- Developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign
- Marc Andreessen, lead student developer. NCSA released it in 1993
- <http://www.ncsa.illinois.edu/>



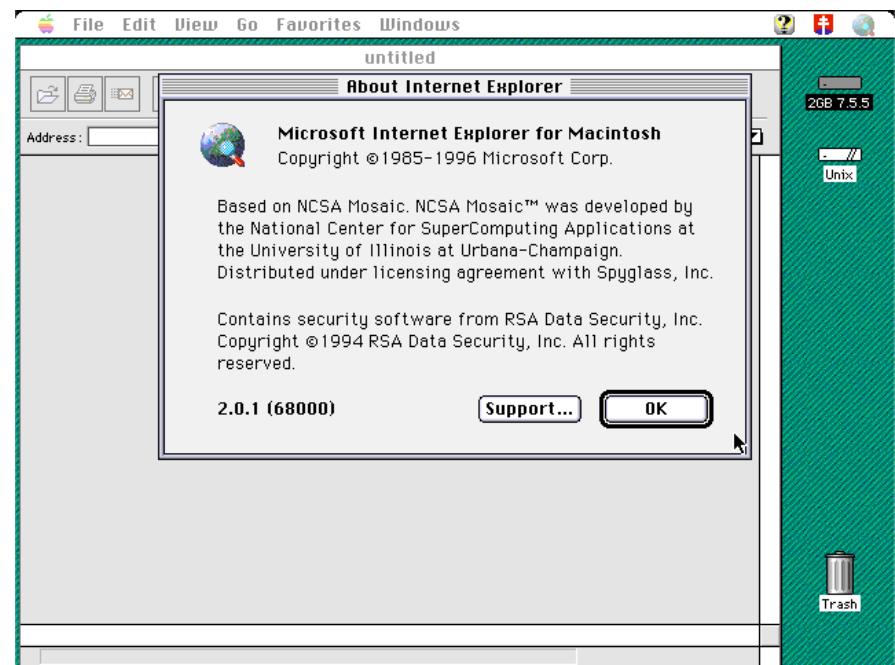
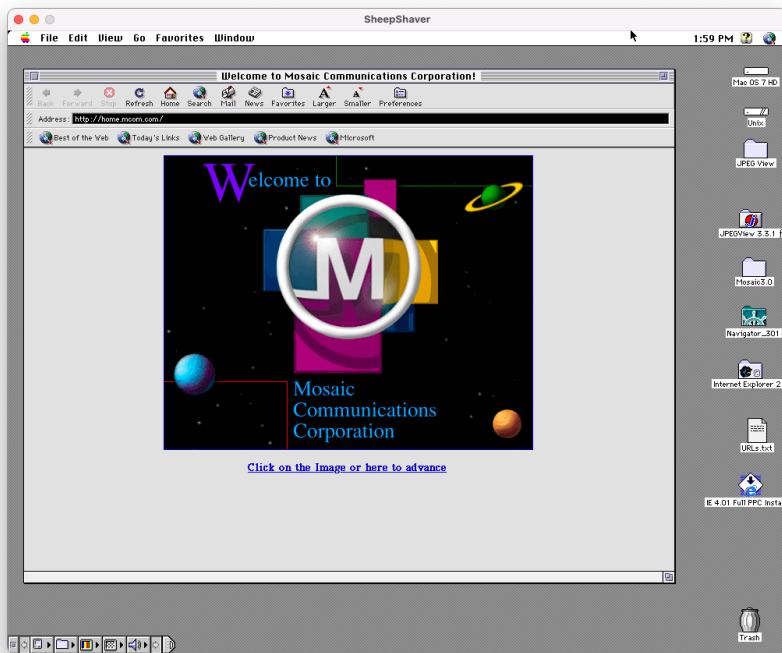
First Commercial Browser – Netscape

- Netscape **Navigator** was a proprietary web browser, and the original browser of the Netscape line
- Co-written by Marc Andreessen, with initial release in Dec. 1994
- <http://home.mcom.com>



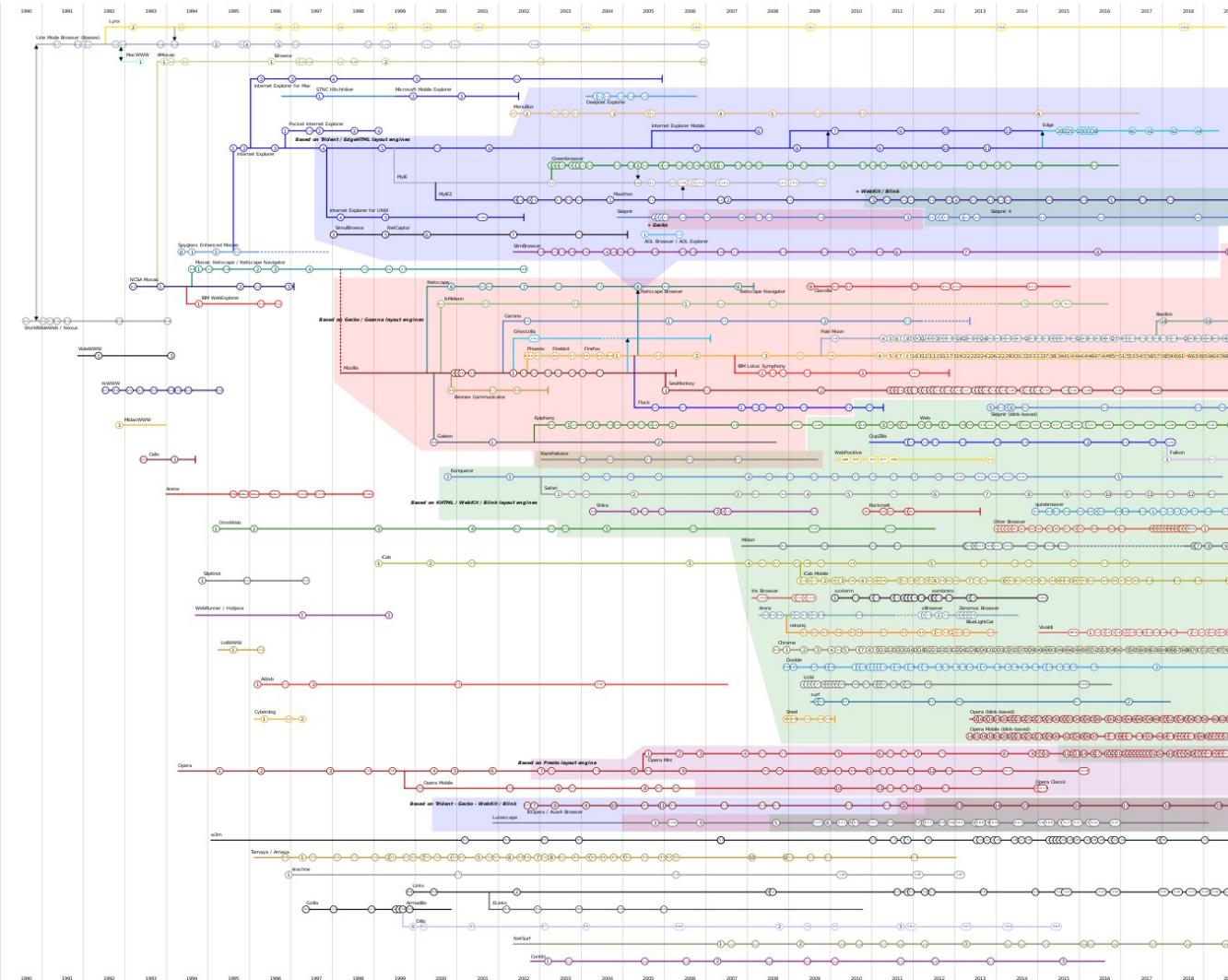
First Free Commercial Browser – IE

- Microsoft Internet Explorer (IE) killed Netscape **Navigator** as it was bundled with Windows 95
- Licensed from NCSA, with initial release in August 1995
- <https://microsoft.com/ie>



Web Browsers Timeline

http://en.wikipedia.org/wiki/List_of_web_browsers



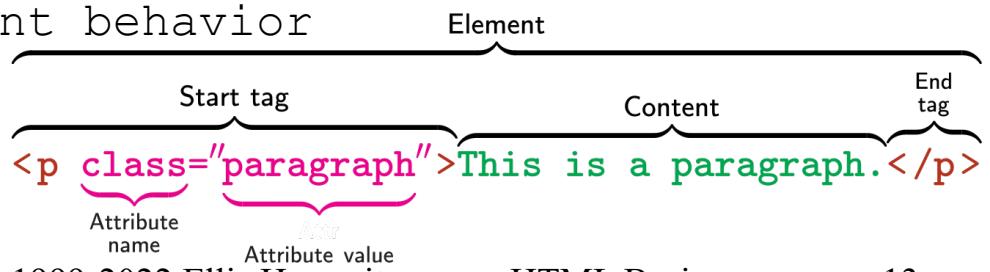
Versions of HTML

- Version 0, 1990, was the original, minimum set of HTML
- Version 1 adds highlighting and images
- Version 2, November 1995, all V.0 and V.1, plus forms
- Version 3.2, January 1997, released by W3C, tables
- HTML 4.01, December 1999 Recommendation:
<http://www.w3.org/TR/html401/>
- HTML5, October 2014 Recommendation, vocabulary and APIs:
<http://www.w3.org/TR/html5/>
- HTML5, December 2017 HTML 5.2 Recommendation:
<https://www.w3.org/TR/html52/>
- **HTML Living Standard, December 2019:**
<https://html.spec.whatwg.org>
- W3C and WHATWG Agreement
<https://www.w3.org/blog/news/archives/7753>
- “W3C stops independent publishing of a designated list of specifications related to HTML and DOM and instead will work to take WHATWG Review Drafts to W3C Recommendations”

NOTE: some of the examples shown here are extracted from the HTML4.0 specification. This document is copyrighted according to: "Copyright © World Wide Web Consortium. All Rights Reserved. <http://www.w3.org/Consortium/Legal/>"

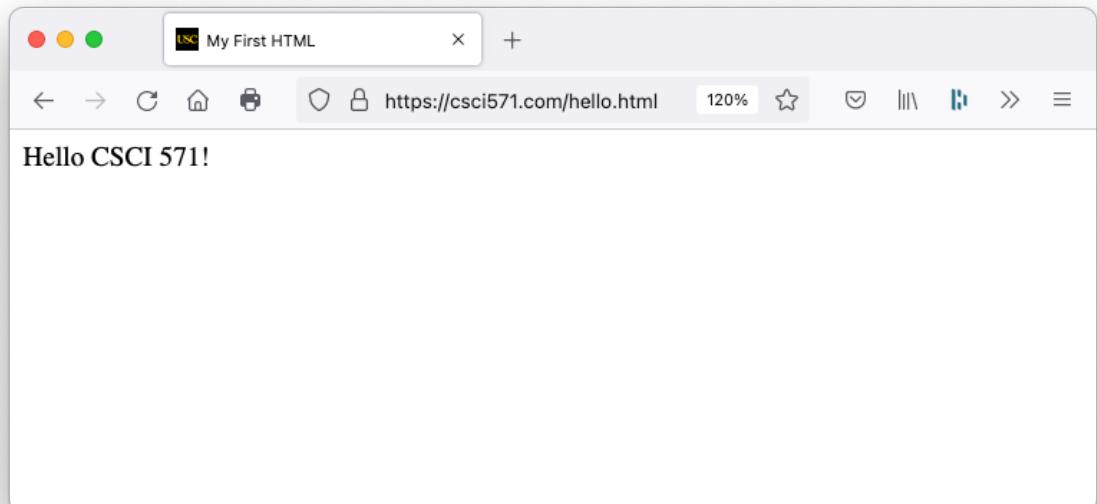
Elements, Tags and Attributes

- An HTML file contains Elements, Tags and Attributes
- HTML Element
 - Basic HTML node that adds semantics
 - Most Elements have a Start Tag, an End tag and content in between
- HTML Tag
 - Composed by the name of the Element surrounded by angle brackets
 - End Tag has a slash after the opening angle bracket (required in HTML5)
- Attribute
 - Specified inside Start Tag
 - Controls the element behavior



My First HTML

```
1 <!DOCTYPE html>
2 <!-- My first HTML comment -->
3 <html>
4   <head>
5     <title>My First HTML</title>
6   </head>
7   <body>
8     Hello CSCI 571!
9   </body>
10  </html>
```



General Structure

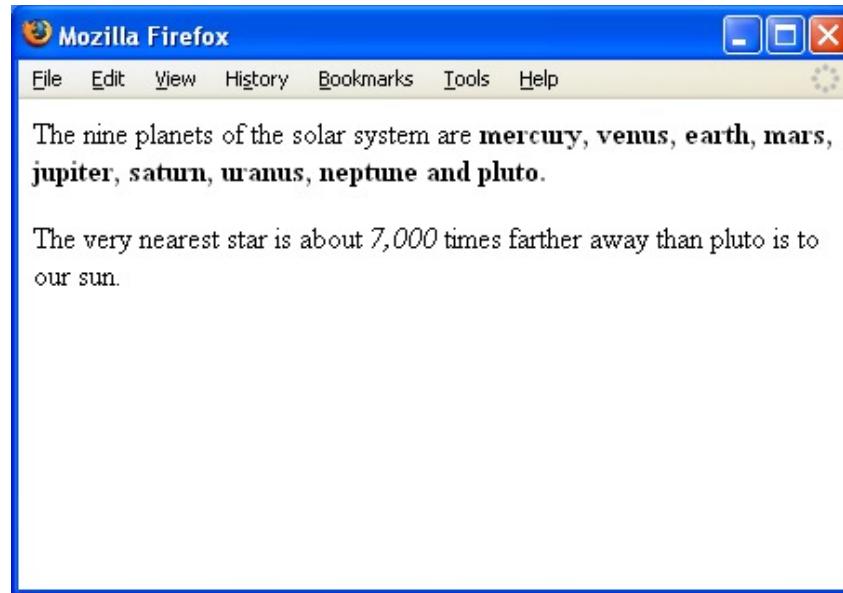
- HTML documents have a head and body
 - head should contain a title
 - body may have paragraphs
- A leading line indicates which version of HTML this document conforms to

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"  
"http://www.w3.org/TR/REC-html40/strict.dtd">  
  
<HTML>  
  
    <HEAD>  
        <TITLE>The Solar System</TITLE>  
    </HEAD>  
  
    <BODY>  
  
        <P>The nine planets of the solar system are...  
    </BODY>  
  
</HTML>
```

Adding Some Markup

```
<P>The nine planets of the solar system are  
<B>mercury, venus, earth, mars, jupiter,  
saturn, uranus, neptune and pluto.</B></P>
```

```
<P>The very nearest star is about  
<I>7,000</I> times farther away than pluto  
is to our sun. </P>
```

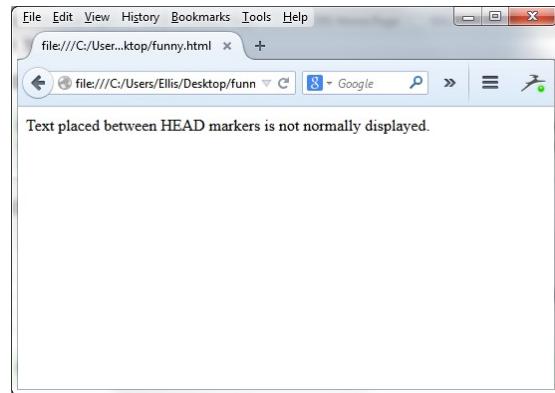


Browsers Are Tolerant

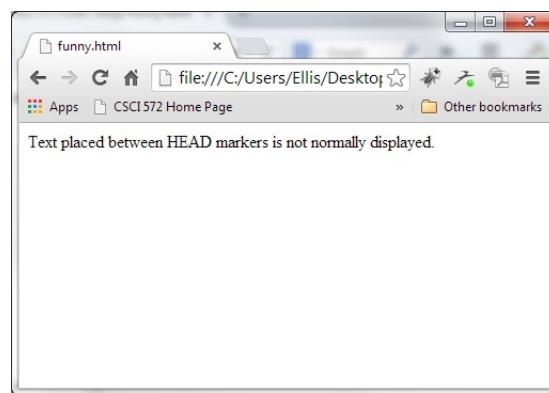
- Browsers follow the rule of being tolerant of mistakes in the input
 - They ignore markup they don't understand
- IE, Edge, Safari, Firefox, Chrome are tolerant browsers
 - They do not insist that the HTML document begin and end with <HTML>
 - <HEAD> and/or <BODY> tags are not required
 - But there is no guarantee that this behavior will be the same for all browsers

Browsers are Tolerant (examples)

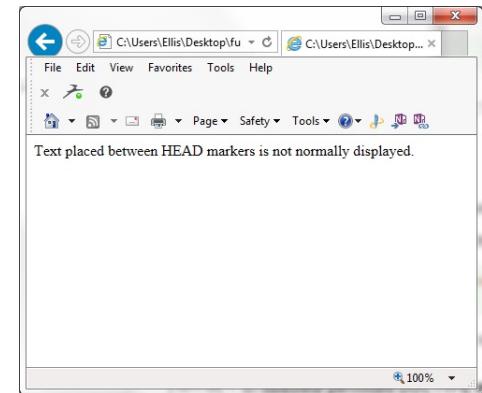
- Suppose the entire document is one line, such as:
<HEAD>Text placed between <ODDTAG> HEAD markers is
not normally displayed.</HEAD>



Firefox



Chrome



Internet Explorer

HTML Elements

- Each element consists of a start tag, content, and an end tag
- E.g., <BODY> some text </BODY>
- A slash (/) after the "<" indicates an end tag
- Some elements do not require end tags, e.g., <P> paragraph tag
- Some elements do not require content, e.g., <HR>

horizontal rule tag places a straight line across the page

Attributes

- Elements may have parameters, called attributes
- Attributes are placed in the start tag before the final ">"
- Attributes have the form name=value

E.g., <H1> is the first heading tag

```
<H1 id="Chapter1">  
Start of Chapter 1  
</H1>
```

- Attribute values are often enclosed in quotes, either double or single
- Quotes are not required when the value contains only letters, digits, hyphens, and periods.
- Attribute names are case insensitive, but not necessarily attribute values

Comments in HTML

- Comments start with

`<!--`

and end with

`-->`

- Comments cannot be nested

- White space is permitted between the `--` and the closing angle bracket, `>`
 - It is not permitted between the opening angle bracket, exclamation point, and the `--`

- E.g.

```
<BODY>
```

```
<!-- This is a comment
```

```
and hence not displayed -->
```

```
But this will be displayed
```

```
</BODY>
```

Complete Set of <BODY> tag attributes

- **id**, assigns a unique name to an element
e.g., <P id=mystart>This is my starting paragraph...
- **class**, assigns one or more names to an element
- **lang**, a language code that identifies a natural language spoken, written, or otherwise used
- **title**, a short description of the body
- **style**, inline display information
- **bgcolor**, background color
- Events include
`onload, onunload onclick, ondblclick, onmousedown,`
`onmouseup, onmouseover, onmousemove, onmouseout,`
`onkeypress onkeydown, onkeyup`
- Deprecated elements in include:
`background, text, link, vlink, alink`
See <http://www.w3.org/TR/html4/struct/global.html#edef-BODY>

<BODY> Using Style Sheets

- Deprecated example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<HTML>
<HEAD>
  <TITLE>A study of population dynamics</TITLE>
</HEAD>
<BODY bgcolor="white" text="black" link="red" alink="fuchsia" vlink="maroon">
  ... document body...
</BODY>
</HTML>
```

- Using Style Sheets

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
  <TITLE>A study of population dynamics</TITLE>
  <STYLE type="text/css">
    BODY { background: white; color: black}
    A:link { color: red } A:visited { color: maroon } A:active { color: fuchsia }
  </STYLE>
</HEAD>
<BODY>
  ... document body...
</BODY>
</HTML>
```

Composing HTML

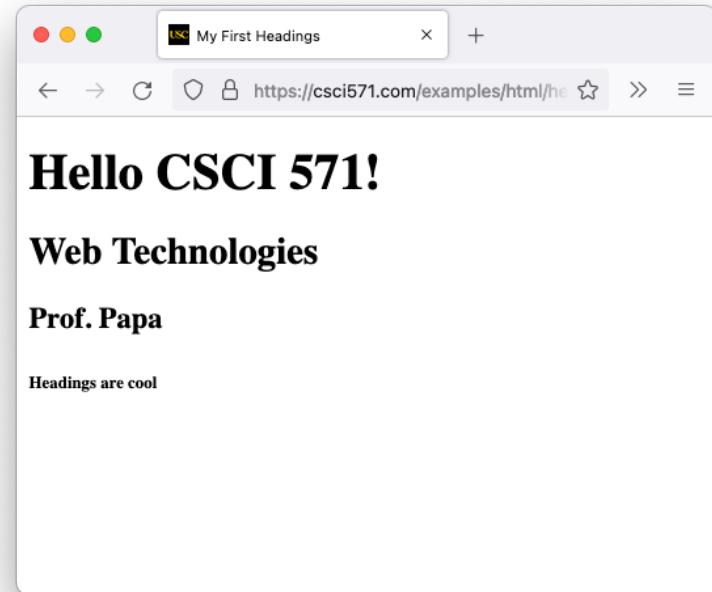
- Conventional editors let you compose HTML directly
 - e.g.,TextEdit,emacs, vi, NotePad, TextPad, etc.
 - use the tools when you are writing HTML directly
- Word Processors include a File SaveAs option which saves your document in HTML format
 - e.g., Microsoft Word
 - **Do not use!**
- There are several free HTML-specific editors, e.g.
 - **Brackets**, <http://brackets.io>
 - **Visual Studio Code**, <https://code.visualstudio.com/>
- There are several commercial HTML/CSS suites, e.g.
 - Adobe Dreamweaver CC (Creative Cloud subscription)
- For a complete list see
http://en.wikipedia.org/wiki/List_of_HTML_editors

HTML Headings

- Heading tags should be used for different levels in a document
 - The size and look can be changed with CSS, so don't use these tags solely to change the size of the font

```
<!DOCTYPE html>

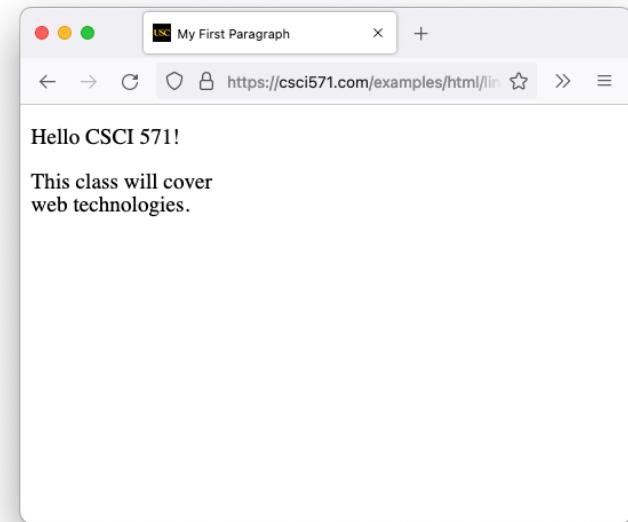
<html>
  <head>
    <title>My First Headings</title>
  </head>
  <body>
    <h1>Hello CSCI 571!</h1>
    <h2>Web Technologies</h2>
    <h3>Prof. Papa</h3>
    <h6>Headings are cool</h6>
  </body>
</html>
```



HTML Line Breaks

- Paragraph tags separate logical display blocks
- Line breaks are used for formatting

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Paragraph</title>
  </head>
  <body>
    <p>Hello CSCI 571!</p>
    <p>This class will cover<br />web technologies.</p>
  </body>
</html>
```



HTML Lists

- HTML lists are very common in everyday web development.
- Perhaps the most used are
 - unordered lists (``) and
 - ordered lists (``)
- but there are a few other list options
 - definition list (`<dl>`), and
 - the menu (`<menu>`) element,
 - both were deprecated in HTML4, but reintroduced in HTML5
- All lists follow the same pattern:
 - `<start tag of list>`
 - `<list item tag>`
 - `<list item tag>`
 - `<list item tag>`
 - `</ end tag of list>`

HTML Definition Lists

<DL>

 <DT>light year<DD>the distance light travels
 in one year

 <DT>asteroids<DD>are small, irregular shaped
 objects, mostly occurring between Mars and
 Jupiter

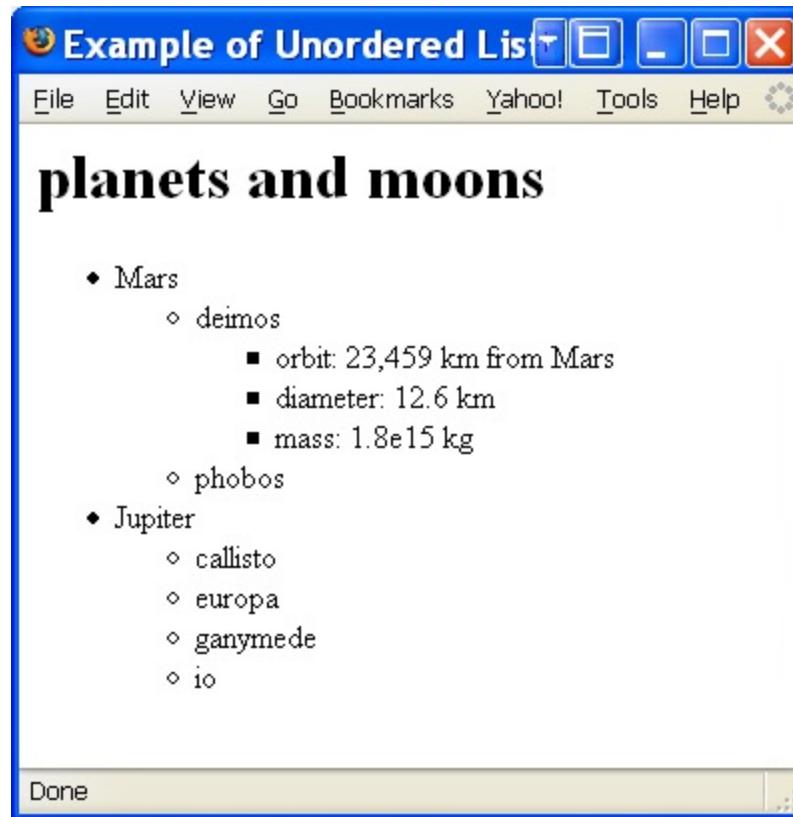
</DL>



HTML Unordered Lists

```
<HTML><HEAD><TITLE>  
Example of Unordered Lists</TITLE></HEAD>  
<BODY><H1>planets and moons</H1>  
<UL>  
<LI>Mars  
    <UL><LI> deimos  
        <UL>    <LI>orbit: 23,459 km from Mars  
            <LI>diameter: 12.6 km  
            <LI>mass: 1.8e15 kg  
        </UL>  
    <LI>phobos  
    </UL>  
<LI>Jupiter  
<UL><LI>callisto<LI>europa<LI>ganymede<LI>io</UL>  
</UL></BODY></HTML>
```

Browser Output of Unordered Lists



HTML Ordered Lists

- Has the general form

```
<OL><LI> first list item<LI> second list item</OL>
```

- START attribute can initialize the sequence to a number other than 1
- TYPE attribute can be used to select the numbering style

Type	Name	Style
1	arabic	1, 2, 3, ...
a	lower alpha	a, b, c, ...
A	upper alpha	A, B, C, ...
i	lower roman	i, ii, iii
I	upper roman	I, II, III,

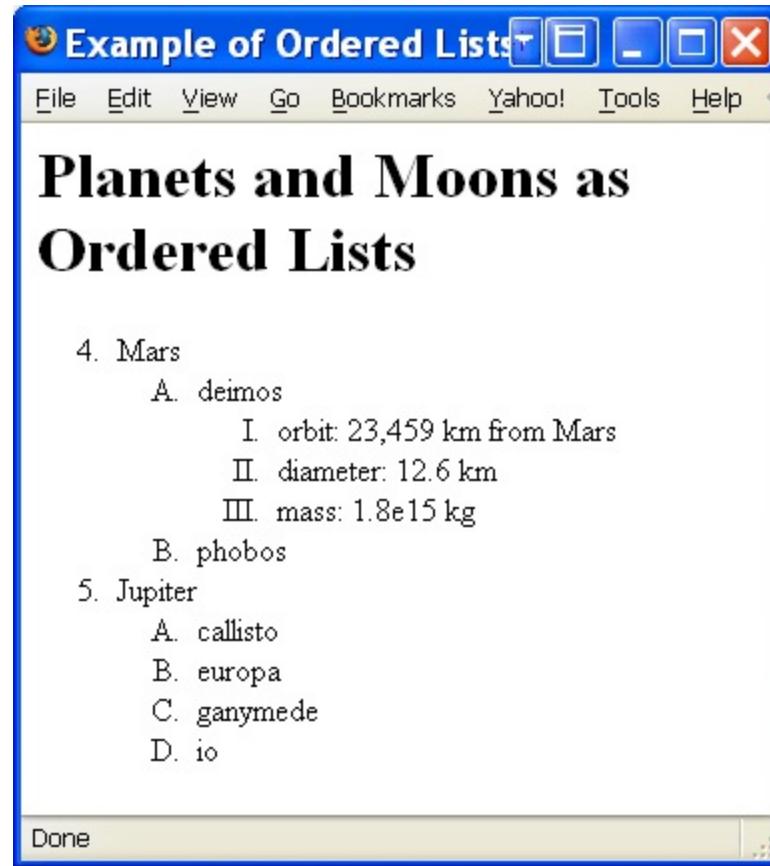
- However, the type attribute is **deprecated**, and list styles should be handled through style sheets, e.g.

```
<STYLE type="text/css">  
OL.withroman {list-style-type: lower-roman}</STYLE>
```

Example - Ordered Lists

```
<HTML><HEAD><TITLE>  
Example of Ordered Lists</TITLE></HEAD>  
<BODY><H1>Planets and Moons as Ordered Lists</H1>  
<OL START=4>  
<LI>Mars  
    <OL type=A><LI>deimos  
        <OL type=I><LI>orbit: 23,459 km from Mars  
            <LI>diameter: 12.6 km  
            <LI>mass: 1.8e15 kg  
        </OL>  
        <LI>phobos  
    </OL>  
<LI>Jupiter  
<OL type=A><LI>callisto<LI>europa<LI>ganymede  
<LI>io</OL></OL></BODY></HTML>
```

Browser Output



HTML Table Elements

- <TABLE>, a tag used to define a table
- <CAPTION>, a tag to label a table
 - Its attributes are ALIGN = top, bottom, left, right
- <TH></TH> or <TD></TD>, tags that identify a table header cell and table data cell
 - Headers are the same as data except they use bold font and are centered
- <TR>, a tag that identifies a container for a row of table cells
 - Same attributes as TH and TD

Facts about Tables

- Table cells can be text, lists, images, forms, figures, or even tables
- Table headers are typically rendered in boldface, and table data is typically rendered in the regular font and point size
- A table has an optional caption followed by rows
- Table rows are said to contain table headers and table data
- The browser will set the number of columns to be the greatest number of columns in all of the rows
- Blank cells are used to fill extra columns in the rows

Example - 3 rows x 2 cols

```
<HTML>
<HEAD>
<TITLE>Table: 3 rows 2 Cols</TITLE>
</HEAD>
<BODY>
<H1>Table: 3 Rows 2 Cols</H1>
<TABLE BORDER="2">
<CAPTION>MIME Content-Types</CAPTION>
<TR><TD>application/postscript</TD><TD>Postscript</TD>
<TR><TD>application/rtf</TD><TD>MS Rich Text Format</TD>
<TR><TD>application/x-pdf</TD><TD>Adobe Acrobat Format</TD>
</TABLE></BODY></HTML>
```

Browser Output

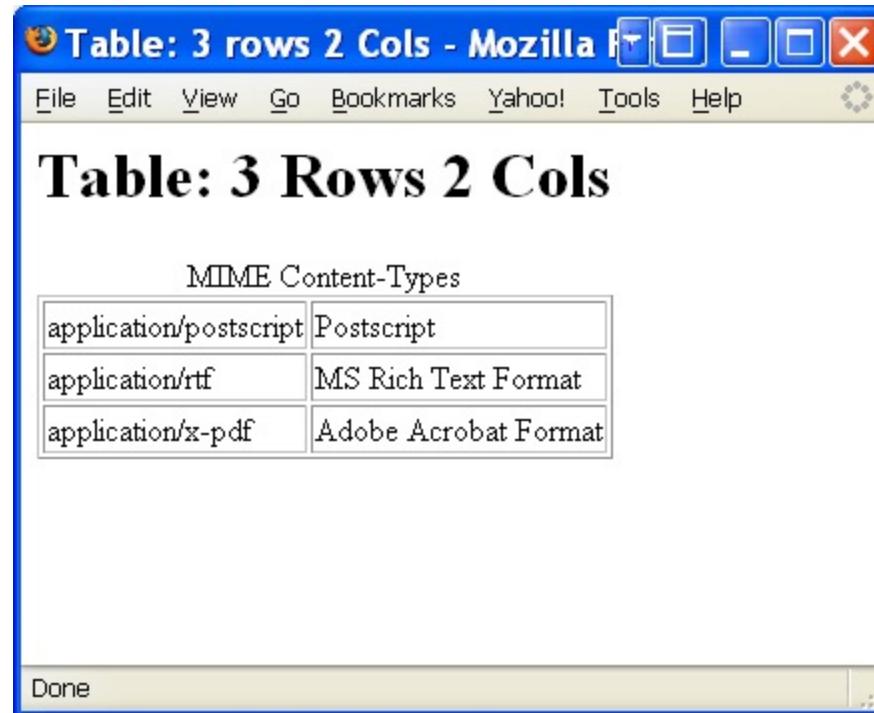


Table Example Rowspan colspan

```
<HTML><HEAD>

<TITLE>Table: Rowspan and Colspan</TITLE>

</HEAD>

<BODY>

<H1>Table: RowSpan and ColSpan</H1>

<TABLE BORDER="2">

<CAPTION>MIME Content-Types</CAPTION>

<TR><TH ROWSPAN=4>Items</TH><TH colspan=2>Types and Their Meaning</TH>

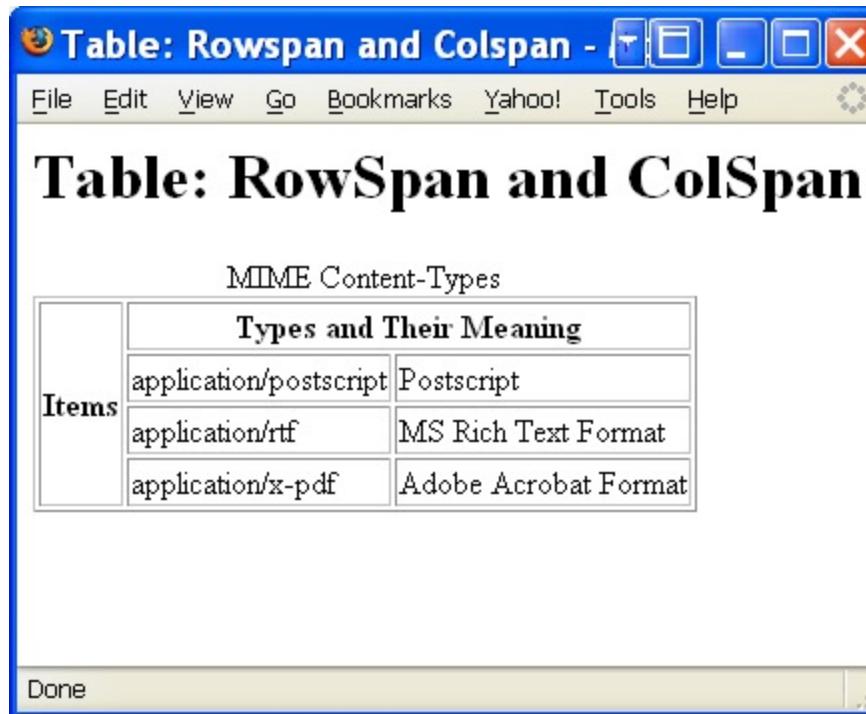
<TR><TD>application/postscript</TD><TD>Postscript</TD>

<TR><TD>application/rtf</TD><TD>MS Rich Text Format</TD>

<TR><TD>application/x-pdf</TD><TD>Adobe Acrobat Format</TD>

</TABLE></BODY></HTML>
```

Browser Output



Arranging Data in a Table

- Originally data in a table could be manipulated using attributes: align left, align right, align center, valign top, valign middle and valign bottom

MIME Content-Types	
Items	Types and Their Meaning
	application/postscript Postscript
	application/rtf MS Rich Text Format
	application/x-pdf Adobe Acrobat Format

- the above attributes are **deprecated** in HTML5 in favor of Cascading Style Sheets (CSS) settings
- What does “deprecated” mean?
- see the slides in the lecture on CSS

HTML Character Set

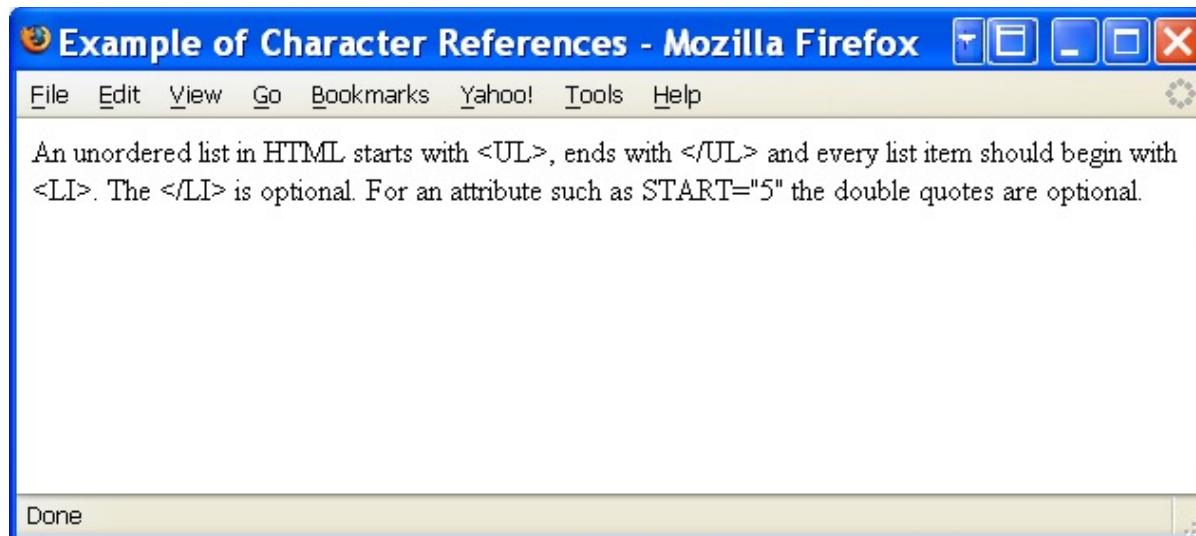
- HTML uses the **Universal Character Set (UCS)**, defined in [ISO10646]. This standard defines a repertoire of thousands of characters used by communities all over the world.
 - Its latest specification (**Unicode 14.0.0**), dated September 14, 2021, can be found at
<https://home.unicode.org/announcing-the-unicode-standard-version-14-0/>
 - Includes Unicode Emoji (with skin tone diversity)
- HTML must also specify how characters are encoded during transmission.
- Commonly used character encodings on the Web include
 - ISO-8859-1 (also referred to as "Latin-1",
 - ISO-8859-5 (which supports Cyrillic),
- A browser is informed of the encoding by a line Content-Type: text/html; charset=EUC-JP

Character references

- Character references in HTML may appear in two forms:
 - **Numeric** character references (either decimal or hexadecimal)
 - `å` (in decimal) represents the letter "a" with a small circle above it (used, for example, in Norwegian).
 - `<` represents left angle bracket
 - `>` represents right angle bracket
 - `&` represents ampersand sign
 - `"` represents double quote
 - Character **entity** references.
 - "`<`" represents the < sign.
 - "`>`" represents the > sign.
 - "`&`" represents the & sign.
 - "`"`" represents the " mark.

Example - Character References

```
<HTML><HEAD><TITLE>Example of Character  
References</TITLE></HEAD>  
  
<BODY>  
  
An unordered list in HTML starts with &lt;UL&gt;,  
ends with &#60;/UL&#62; and every list item should  
begin with &lt;LI&gt;. The &lt;/LI&gt; is optional.  
For an attribute such as START="5"&#34; the double  
quotes are optional.  
  
</BODY></HTML>
```



HTML Hyperlinks (Anchors)

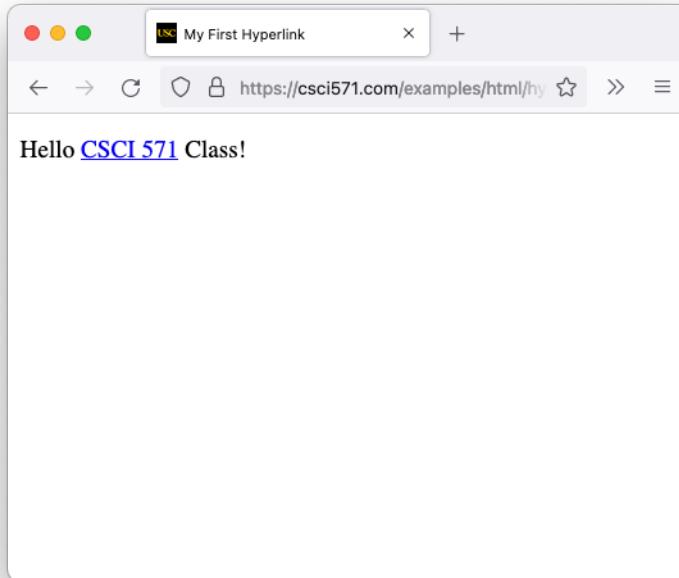
- Hyperlinks are the “novel idea” introduced by Tim Berners-Lee
- Implemented in HTML with Anchor Tags
- An anchor is a way to designate a **link to another document** or to a specific place in the **same document**

- Begins with <A> and ends with
- The link location is given by the “required” **Href attribute** (Hypertext REference); e.g.,
`Class Home Page`
- Hypertext links are displayed using underlining, color, and/or highlighting
 - Depends on the browser defaults or style settings
 - Once a link is taken, it should change color
 - HREF, stands for **Hypertext REference**
- Link destination can be relative or absolute

HTML Hyperlinks (cont'd)

- Anchor tags allow one page to link to another page

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>My First Hyperlink</title>  
  </head>  
  <body>  
    <p>Hello <a href="https://csci571.com">CSCI 571</a>  
Class!</p>  
  </body>  
</html>
```



Syntax of Anchor Names

- An anchor name is the value of either the name or id attribute when used in the context of anchors.
- Anchor names must observe the following rules:
 - **Uniqueness:** Anchor names must be unique within a document. Anchor names that differ only in case may not appear in the same document.
 - **String matching:** Comparisons between fragment identifiers and anchor names must be done by exact (case-sensitive) match.
- See <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a>

Defining Anchors Using the id Attribute

- The id attribute may be used to create an anchor at the start tag of any element (including the A element).
- Example: the **id attribute** places an anchor in an H2 element.

You may read more about this in

```
<A href="#section2">Section Two</A>.
```

. . . more text . . .

```
<H2 id="section2">Section Two</H2>
```

. . . more text

```
<P>Please refer to <A href="#section2">Section Two</A> above for  
more details.
```

- The id and name attributes share the same name space. They cannot both define an anchor with the same name in the same document. Try this example in IE and Firefox:

```
<P><A href="#label1">...</A>
```

. . . more document . . .

```
<P><A name="LABEL1">...</A>
```



**NOTE: browsers work
differently**

Examples of Anchors

- a file in the same directory and same domain as the current page
- a file in directory mydocs on machine nunki.usc.edu, a WWW site
- the newsgroup computers.compilers
- opens an e-mail window for sending a message
-
Download RFC1866 executes the ftp program to fetch a file
- Given the current position, this <HREF> moves up one directory, connects to Docs/Style/ and displays the document Overview.html

 ...
- Connects to lycos and runs pursuit with three arguments
<A HREF="http://www.lycos.com/cgi-bin/pursuit?
Hypertext+Markup+Language"> ...

Anchor Titles

- The title attribute may be set to add information about the nature of a link.
- This information may be spoken by a user agent, rendered as a tool tip, cause a change in cursor image, etc.

```
<BODY>
```

```
    ...some text...
```

```
<P>You'll find a lot more in <A href="chapter2.html"  
        title="Go to chapter two">chapter two</A>.
```

```
See also this <A href="../images/solarsystem.gif"  
        title="GIF image of solar system">view of the  
solar system.</A>
```

```
</BODY>
```

Universal Resource Identifier (URI)

- *URIs* typically consist of three pieces:
 - The **scheme** of the mechanism used to access the resource.
 - The name of the machine **hosting** the resource.
 - The name of the **resource** itself, given as a **path**
E.g., `http://www.usc.edu/dept/cs/index.html`
- Fragment identifiers are URIs that refer to a location within a resource
 - e.g., `http://www.usc.edu/dept/cs/index.html#section2`
- Relative URIs have a path that refers to a resource on the same machine as the current document, e.g., "..." means one level up
- See p. 24 in:
`http://www.ietf.org/rfc/rfc3986.txt`
- See also:
`https://en.wikipedia.org/wiki/Fragment_identifier`

The <LINK> Element

- Should only appear in the HEAD
- It may appear any number of times
- It conveys relationship information that may be rendered in a variety of ways (e.g., a tool-bar with a drop-down menu of links, external files)
- Example - The current document is "Chapter2.html". The rel attribute specifies the relationship of the linked document with the current document.

```
<HTML> <HEAD>

    <TITLE>Chapter 2</TITLE>
    <LINK rel="Index" href="../index.html">
    <LINK rel="Next"   href="Chapter3.html">
    <LINK rel="Prev"   href="Chapter1.html">
</HEAD>
...the rest of the document...
```

How is <LINK> Used

- To provide a variety of information to search engines:
 - Links to **alternate versions** of a document, written in another human language, e.g.

```
<LINK lang="fr" title="La documentation en Fran&ccedil;ais"  
      type="text/html" rel="alternate" hreflang="fr"  
      href="http://domain/manual/french.html">
```

- Links to alternate versions of a document, designed for **different media**

```
<LINK media="print" title="The manual in postscript"  
      type="application/postscript"  
      rel="alternate"  
      href="http://domain/manual/usermanual.ps">
```

- Links to the starting page of a collection of documents.
- Links to style sheets and “media queries” used in Responsive Web Design

Creating Graphics

- Digital cameras & Smartphones
 - Snap and the image is digitized and can be transferred to a computer
 - Typical resolutions are 1280x720, 1920x1080
- Graphic editors
 - Permit the combination of text, drawing, and color
 - For example, Adobe Photoshop
- Scanners
 - Convert text and graphics into machine readable form

Image Formats

- Four image formats are always supported by Web browsers
 - x-pixelmaps (obsolete)
 - Similar to x-bitmaps, but 8 bits are given to each pixel, permitting 256 colors in the image
 - Graphic Interchange Format (**GIF**)
 - Support black and white, grayscale, and color
 - Patented by Unisys (expired, abandoned)
 - Joint Photographic Experts Group (**JPEG**)
 - Designed for photographic images
 - Includes image compression
 - Portable Network Graphics (**PNG**)
 - An open, extensible image format with Lossless Compression
 - Patent-free replacement for GIF and TIFF
 - W3C Recommendation: <http://www.w3.org/TR/PNG/>

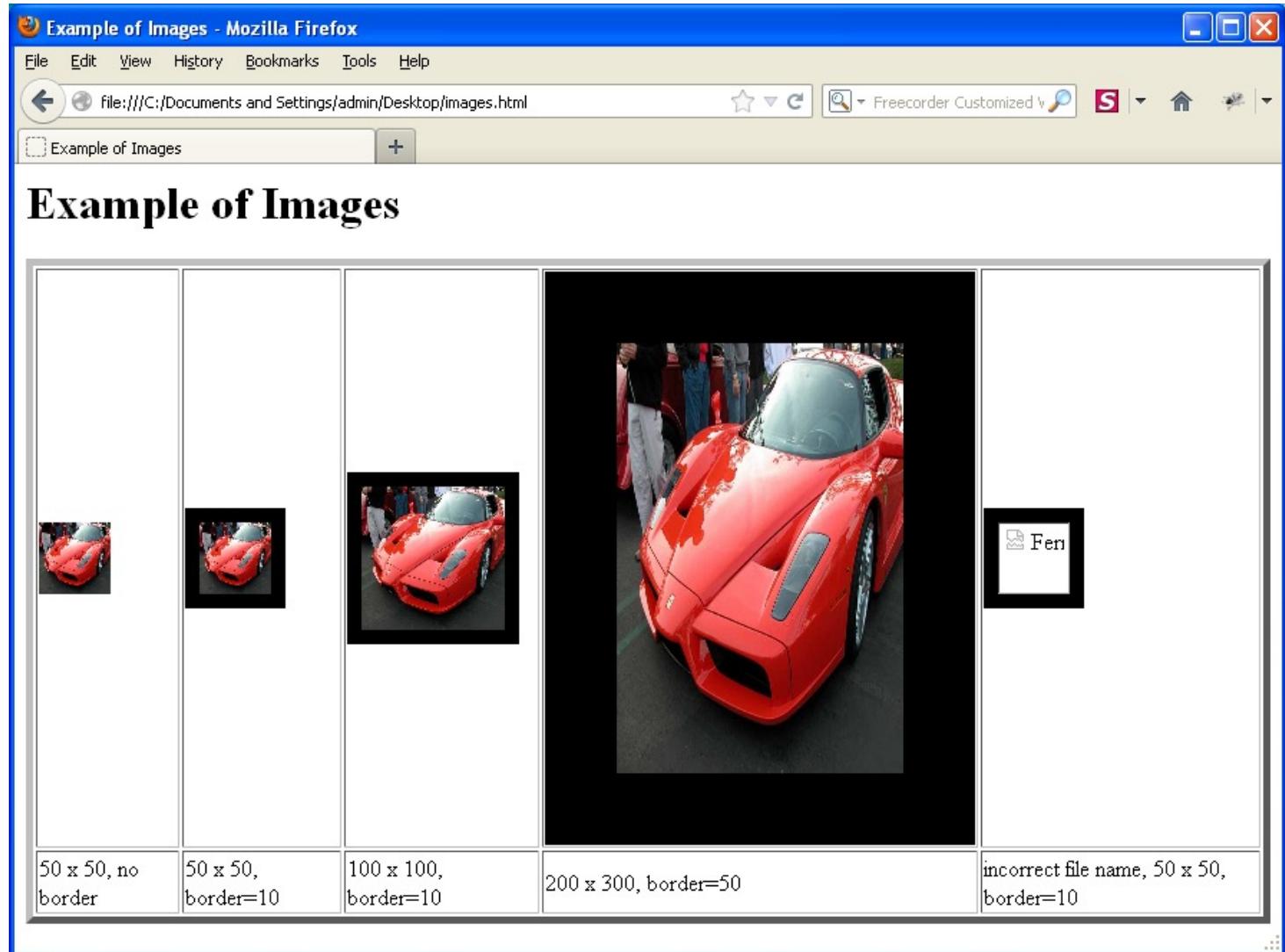
HTML Image Element

- The **IMG** element embeds an image in the current document, e.g.,
- Some attributes of include
 - **SRC**, the HREF or name of the image
 - Align=top, middle, or bottom to align text around an image
 - border to place a border around an image
 - If set to 1 or higher, places a black box around the image
 - If set to 0, leaves no surrounding box
 - **height** and **width** to control the dimensions of the image
 - Alt to replace an image with text, if the image is unavailable or a text browser is used; e.g.,

Example - Images

```
<HTML><HEAD><TITLE>Example of Images</TITLE></HEAD>
<BODY>
<H1>Example of Images</H1>
<table border=5><tr>
<td><IMG SRC="ferrari.jpg" ALT="Ferrari" BORDER=0 WIDTH=50 HEIGHT=50>
<td><IMG SRC="ferrari.jpg" ALT="Ferrari" BORDER=10 WIDTH=50 HEIGHT=50>
<td><IMG SRC="ferrari.jpg" ALT="Ferrari" BORDER=10 WIDTH=100 HEIGHT=100>
<td><IMG SRC="ferrari.jpg" ALT="Ferrari" BORDER=50 WIDTH=200 HEIGHT=300>
<td><IMG SRC="xferrari.jpg" ALT="Ferrari" BORDER=10 WIDTH=50
HEIGHT=50></tr>
<tr>
<td>50 x 50, no border</td>
<td>50 x 50, border=10</td>
<td>100 x 100, border=10</td>
<td>200 x 300, border=50</td>
<td>incorrect file name, 50 x 50, border=10</td>
</tr>
</table>
</BODY></HTML>
```

Browser Output



Active Images

- Active images are images that can be clicked and, just like an anchor, they act as a hypertext link

```
<A HREF="http://sunset.usc.edu:8080/index.html">  
<IMG SRC="USCimage.gif"> </A>
```

- Active images have a border around them and the cursor changes shape when passed over

Image Maps

- Image maps are active images with multiple clickable regions
- each region can be associated with a specific action (e.g., retrieve a document, run a program, etc.)
- When the region is activated by the user, e.g., by a mouse click, the action is taken
 - the pixel coordinates are interpreted by the browser (**USEMAP** attribute).
 - The **<map>** tag is used to associate the image and the regions.

An ImageMap Example from Wikipedia



To see the ImageMap work
go to
http://en.wikipedia.org/wiki/Image_map
and click on each of the individuals
at the table

```
  
  
<map id="ImageMap_1_2013620197"  
name="ImageMap_1_2013620197">  
<area title="Dr Johnson – Dictionary writer"  
alt="Dr Johnson – Dictionary writer"  
coords="133,343,124,287,159,224,189,228,195,291,222,  
311,209,343,209,354,243,362,292,466,250,463"  
shape="poly" href="/wiki/Samuel_Johnson">  
<area title="Boswell – Biographer" alt="Boswell"  
coords="76,224,84,255,43,302,62,400,123,423,121,361,  
137,344,122,290,111,234,96,225"  
shape="poly" href="/wiki/James_Boswell">  
<area title="Sir Joshua Reynolds – Host"  
alt="Sir Joshua Reynolds – Host"  
coords="190,276,208,240,229,228,247,238,250,258,286,  
319,282,323,223,323,220,301,200,295"  
shape="poly" href="/wiki/Joshua_Reynolds">
```

Imagemaps

- Add USEMAP attribute to tag to indicate a client-side imagemap, e.g.

```
<IMG SRC="images/banner.gif" USEMAP="#bannerbar">
```

- Different regions of the image are described using <MAP> tag, e.g.

```
<MAP NAME="bannerbar">
<AREA SHAPE="RECT" COORDS="10,10,50,50" HREF="p1.html">
<AREA SHAPE="RECT" COORDS="50,10,90,50" HREF="p2.html">
<AREA SHAPE="RECT" COORDS="90,10,130,50" HREF="p3.html">
<AREA NOHREF SHAPE=default>
</MAP>
```

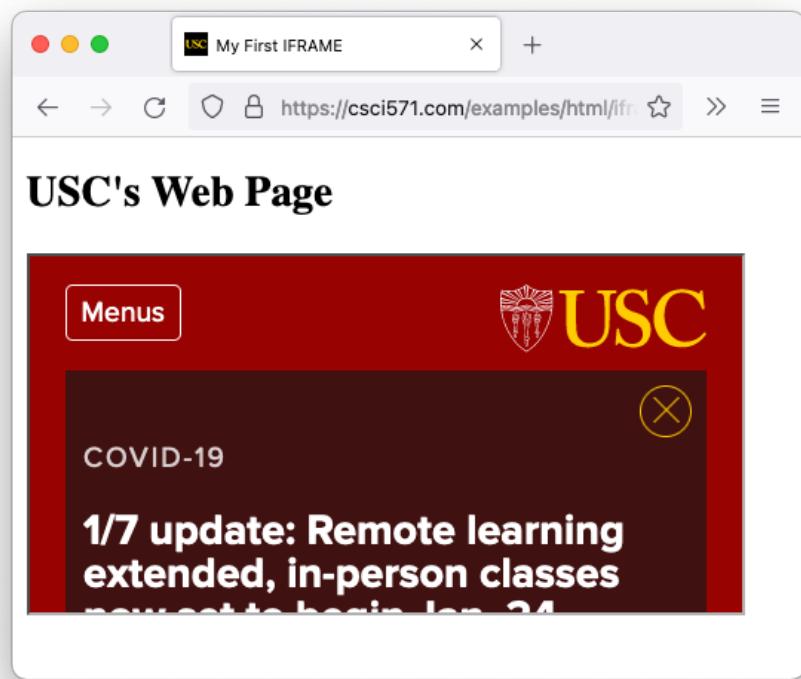
- Possible values for SHAPE are:

- default**: Specifies the entire region.
- rect**: Defines a rectangular region.
- circle**: Defines a circular region.
- poly**: Defines a polygonal region.

HTML Inline Frame

- Used to “embed” a website

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First IFRAME</title>
  </head>
  <body>
    <h2>USC's Web Page</h2>
    <iframe src="https://www.usc.edu" height="200"
width="400" />
  </body>
</html>
```



<META> Element

- Allows you to insert Name/Value pairs describing document properties, e.g.

```
<META NAME="Author" CONTENT="Ellis Horowitz">
```

- USC CS dept home page header

```
<META name = "description" content = "The Computer  
Science Department at the University of Southern  
California, Los Angeles (USC) provides education leading  
to the Bachelors, Masters and Ph.D. degrees in Computer  
Science.">
```

```
<META name = "keywords" content = "USC, computer science,  
computer science research, computer science teaching">
```

```
<META name = "author" content = "Ellis Horowitz">
```

<META> Element

- Moving a Web page to a new site

```
<HTML><HEAD>
```

```
<META HTTP-EQUIV="REFRESH" CONTENT="5;  
URL=http://www.usc.edu/dept/cs/">
```

```
<META NAME="GENERATOR" CONTENT="Mozilla/4.04 [en] (Win95; I)  
[Netscape]">
```

```
<TITLE>This site has moved</TITLE></HEAD>
```

```
<BODY><CENTER>This site has moved to a new location which is:
```

```
<A
```

```
    HREF="http://www.usc.edu/dept/cs/">http://www.usc.edu/dept  
/cs/</A> <BR>
```

Your browser should automatically move to the correct URL in five seconds.

```
</CENTER></BODY></HTML>
```

Meta Tag and Robot Exclusion

```
<meta name="robots" content="noindex,nofollow">  
<title>...</title> </head> <body> ...
```

- The content of the Robots META tag contains directives separated by commas.
- The currently defined directives are
 - [NO] INDEX. The INDEX directive specifies if an indexing robot should index the page.
 - [NO] FOLLOW The FOLLOW directive specifies if a robot is to follow links on the page.
 - The defaults are INDEX and FOLLOW. The values ALL and NONE set all directives on or off: ALL=INDEX, FOLLOW and NONE=NOINDEX, NOFOLLOW.
- Some examples:

```
<meta name="robots" content="index,follow">  
<meta name="robots" content="noindex,follow">  
<meta name="robots" content="index,nofollow">  
<meta name="robots" content="noindex,nofollow"> Note the  
"robots" name of the tag and the content are case  
insensitive.
```

Validating Your HTML

- The reasons for validation
 - Browsers display HTML differently
 - Browsers treat HTML errors differently
- What validators do
 - Flag syntax errors with respect to HTML DTD
 - Compare your pages to HTML 4.x, XHTML, and even HTML 5 (experimental)
- Some tools are downloaded to your site; others read your Web page from a URL
- HTML validation tools can be found at:
<http://search.yahoo.com/bin/search?p=html+validation>
- W3C Markup Validation Service:
<http://validator.w3.org/>

W3C Markup Validation Service

- Options: character encoding & Document type

The screenshot shows a web browser window with multiple tabs open. The active tab is 'validator.w3.org/#validate_by_uri+with_options'. The page title is 'Markup Validation Service' with the subtitle 'Check the markup (HTML, XHTML, ...) of Web documents'. There are three tabs at the top: 'Validate by URI' (selected), 'Validate by File Upload', and 'Validate by Direct Input'. The main form is titled 'Validate by URI' and contains a field 'Address:' with the value 'http://www.usc.edu'. Below this is a section titled 'More Options' with several configuration options:

- Character Encoding: dropdown set to '(detect automatically)', with a checkbox 'Only if missing'.
- Document Type: dropdown set to '(detect automatically)', with a checkbox 'Only if missing'.
- Radio buttons: 'List Messages Sequentially' (selected) and 'Group Error Messages by Type'.
- Checkboxes: 'Show Source', 'Clean up Markup with HTML-Tidy', 'Show Outline', 'Validate error pages', and 'Verbose Output'.

A large 'Check' button is centered below these options. At the bottom of the main form, a note states: 'This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available.' Below this note is a Mozilla logo and a message about community support. A footer navigation bar includes links for Home, About..., News, Docs, Help & FAQ, Feedback, and Contribute. The W3C logo is also present. The page footer contains copyright information and links to privacy statements.

Validating Your HTML

- Sample output

The screenshot shows a web browser window displaying the W3C Markup Validator results for a document. The browser tabs include 'validator.w3.org/check?uri=http%3A%2F%2Fwww.usc.edu&charset=%28detect+automatically...'. The main content area is titled 'Notes and Potential Issues' and contains a note about using an experimental feature: 'Using experimental feature: HTML5 Conformance Checker'. It explains that the validator checked the document with this feature and provides a link to report issues. Below this is the 'Validation Output: 1 Error' section, which lists a single error: 'Line 70, Column 77: Element style is missing required attribute scoped.' The error details show the problematic code snippet: `_tomic="true" aria-live="polite">><style>.feature{background:url(http://www.usc...`. It also lists attributes for the element: `style`, `Global attributes`, `media`, `type`, and `scoped`. A note states that the `title` attribute has special semantics on this element. At the bottom of the page, there are links for 'Home', 'About...', 'News', 'Docs', 'Help & FAQ', 'Feedback', and 'Contribute'. The W3C logo is visible on the left, and a footer note at the bottom right states: 'This service runs the W3C Markup Validator, v1.2+hg. COPYRIGHT © 1994-2011 W3C® (MIT, ERCIM, KEIO). ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.'

Helpful Links to Play with

- W3 Schools
<https://www.w3schools.com/html/>
- CODEPEN
<https://codepen.io/>
- JS Fiddle
<https://jsfiddle.net/>
- JS BIN
<https://jsbin.com/?html,output>

Lecture

HTML: Style Sheets

Style Sheets

- Starting with HTML version 4.x there was an effort to **separate** the specification of **style** (presentation) from the specification of **content**
- Style sheets are the mechanism HTML assumes will be used to specify
 - the amount of white space between text or between lines,
 - the amount lines are indented,
 - the colors used for the text and the backgrounds
 - the font size and style of text
 - the precise position of text and graphics
 - How front matter (preface, figure list, title page, and so forth) should look
 - How all or individual sections should be laid out in terms of space (for example, two newspaper columns, one column with headings having hanging heads, and so forth)

CSS

- CSS stands for Cascading Style Sheets
 - It is not a markup language. It is a list of Rules
- CSS is like JSON Objects, with keyword: value pairs
 - CSS keywords are already defined though

Version	Year
CSS 1	1996
CSS 2	1998
CSS 2.1	2011
CSS 3 modules	2012-2022
CSS3	ongoing

CSS



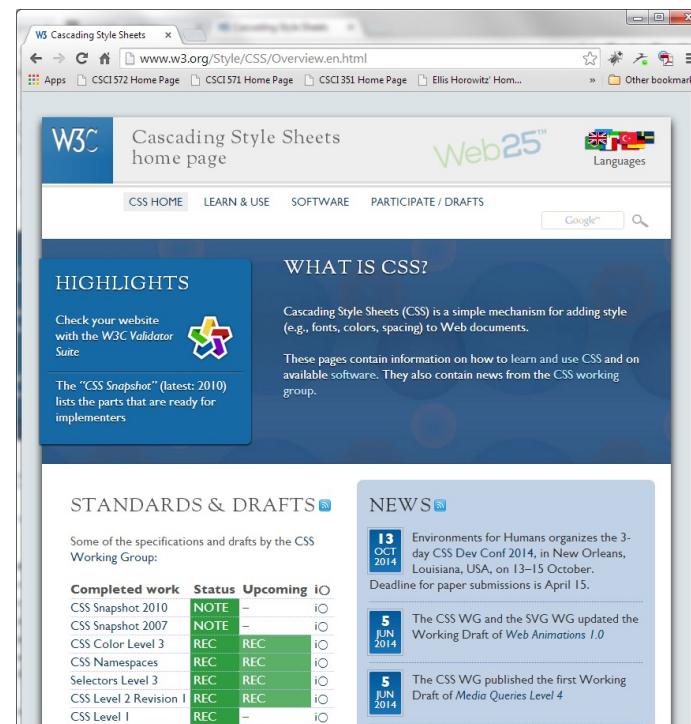
Style Sheet Languages

- These notes use the style language "Cascading Style Sheets" ([CSS1] and [CSS2]), abbreviated CSS, and described in:

<http://www.w3.org/TR/REC-CSS1> (level 1)
<http://www.w3.org/TR/REC-CSS2> (level 2)
<http://www.w3.org/TR/CSS21/> (level 2 Revision 1)
<http://www.w3.org/Style/CSS/current-work> (Level 3, in process)

<http://www.w3.org/Style/CSS/Overview.en.html> (see Status)

- There are other style sheet languages, e.g., XSL defined at
<http://www.w3.org/Style/XSL>



The screenshot shows the W3C Cascading Style Sheets (CSS) home page. The header features the W3C logo and navigation links for CSS HOME, LEARN & USE, SOFTWARE, and PARTICIPATE / DRAFTS. A search bar is also present. The main content area includes sections for HIGHLIGHTS (Check your website with the W3C Validator Suite, CSS Snapshot 2010), WHAT IS CSS? (a brief description of CSS), STANDARDS & DRAFTS (listing completed work like CSS Snapshot 2010, CSS Snapshot 2007, CSS Color Level 3, etc.), and NEWS (announcements for CSS Dev Conf 2014 and the Working Draft of Web Animations 1.0).

Expressing Style Within HTML

- CSS can be included in three ways
 1. **Inline** in an HTML element through the **style attribute**
 2. In the **<style> element**, contained in the `<head>` of an HTML document
 3. In an external file that is included in an HTML document in the **<link> tag**
- Combining style information from multiple sources, called “cascading”
- There is a defined order of precedence where the definitions of a style element conflict
- These HTML extensions permit
 - flexible placement of style information
 - independence from any particular style sheet language

Simple Example

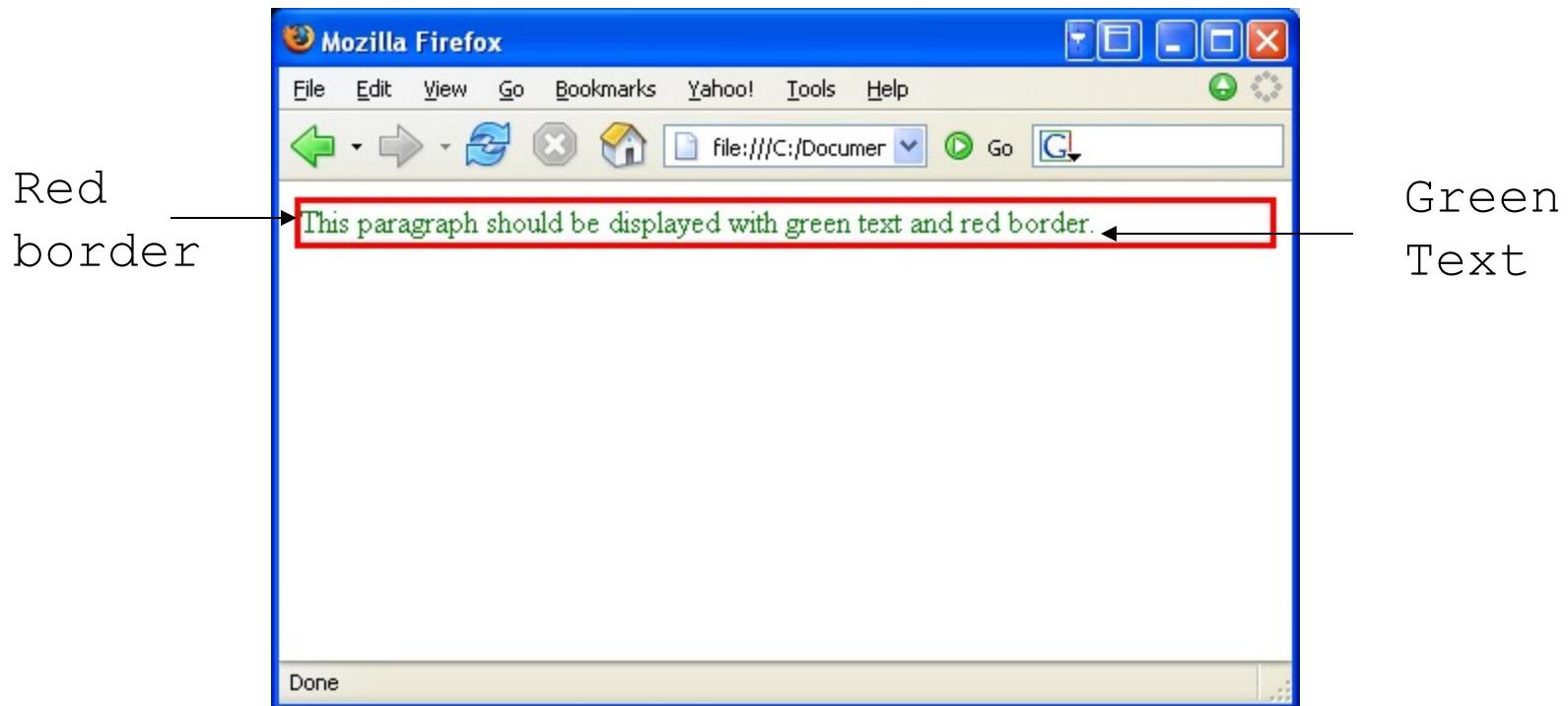
- Suppose a file, special.css contains 3 lines that sets the text color of a paragraph to green and surrounds it with a solid red border:

```
P.special {  
    color : green;  
    border: solid red;  
}
```

- Authors may link this style sheet to their source HTML document with the LINK element:

```
<HTML> <HEAD>  
    <LINK href="special.css" rel="stylesheet"  
          type="text/css">  
</HEAD> <BODY>  
    <P class="special">This paragraph should be  
        displayed with green text and red border.  
</BODY> </HTML>
```

Browser Output



Setting Color

- Expressed either as Red, Green, Blue (RGB) triple or using predefined color names
- Four equivalent forms of specifying color

```
EM { color: #f00 }      /* #rgb */  
EM { color: #ff0000 }    /* #rrggbb */  
EM { color: rgb(255,0,0) } /* integer [0,255] */  
EM { color: rgb(100%, 0%, 0%) } /* float range  
0.0% - 100.0% */
```

- Pre-defined color** names

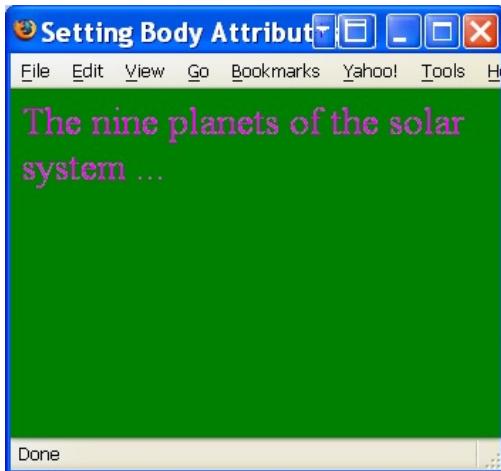
Black="#000000"	Green="#008000"
Silver="#C0C0C0"	Lime="#00FF00"
Gray="#808080"	Olive="#808000"
White="#FFFFFF"	Yellow="#FFFF00"
Maroon="#800000"	Navy="#000080"
Red="#FF0000"	Blue="#0000FF"
Purple="#800080"	Teal="#008080"
Fuschia="#FF00FF"	Aqua="#00FFFF"

Option 1: **Inline style Attribute**

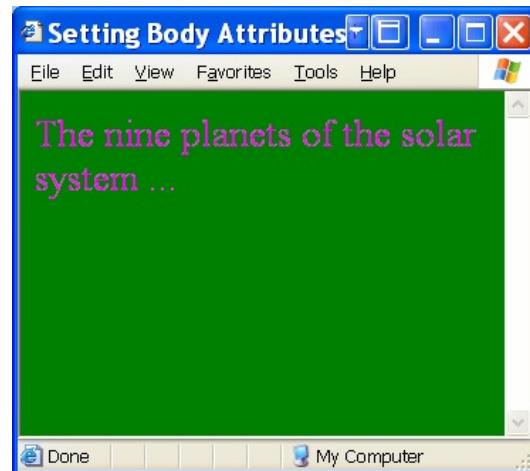
- Using inline style settings to set the font size, background and color of text

```
<HTML>
  <HEAD>
    <TITLE>Setting Body Attributes</TITLE>
  </HEAD>
  <BODY style="font-size: 20pt;
                background: green; color: fuchsia">
    The nine planets of the solar system ...
  </BODY>
</HTML>
```

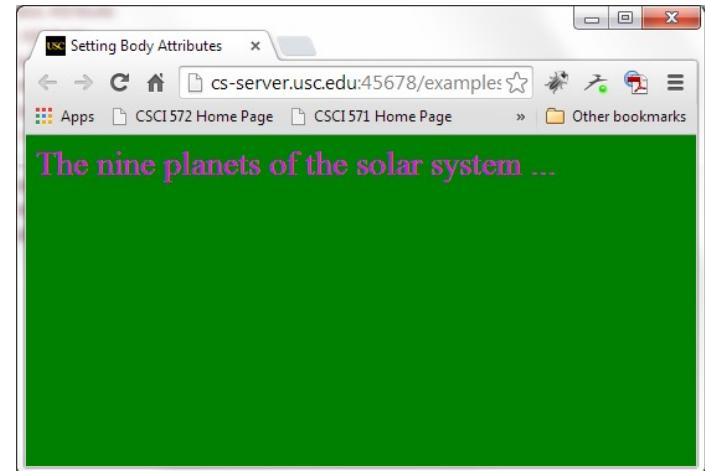
Browser Output



Firefox



Internet Explorer



Chrome

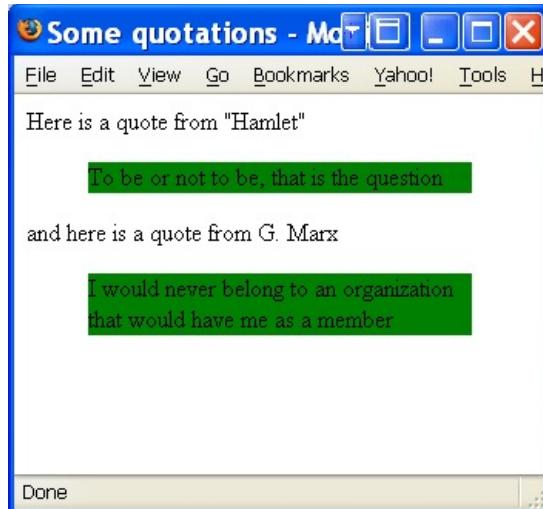
Option 2: Using the <STYLE> Element

- <STYLE> element is placed in the <HEAD>
- Example using the <STYLE> element to assign a fuchsia background to all QUOTATIONS belonging to class named example

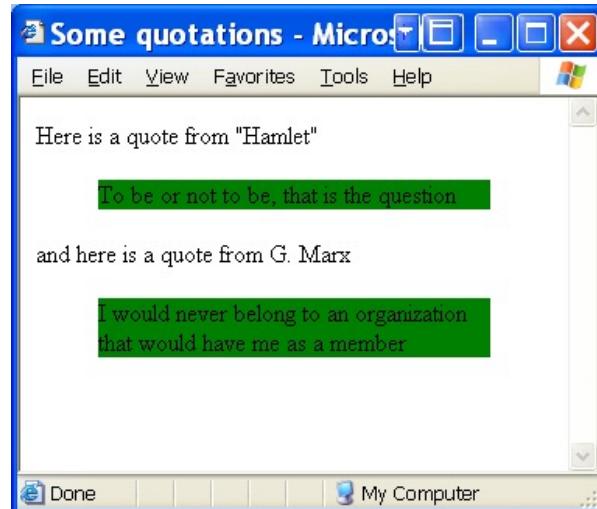
```
<HTML><HEAD><TITLE>Some quotations</TITLE>
<STYLE type="text/css">
BLOCKQUOTE.example { background : green }
</STYLE></HEAD>

<BODY>Here is a quote from "Hamlet"
<BLOCKQUOTE class="example" id="example-1"> To be or
not to be, that is the question</BLOCKQUOTE>
and here is a quote from G. Marx
<BLOCKQUOTE class="example" id="example-2"> I would
never belong to an organization that would have me as
a member</BLOCKQUOTE>
</BODY>
</HTML>
```

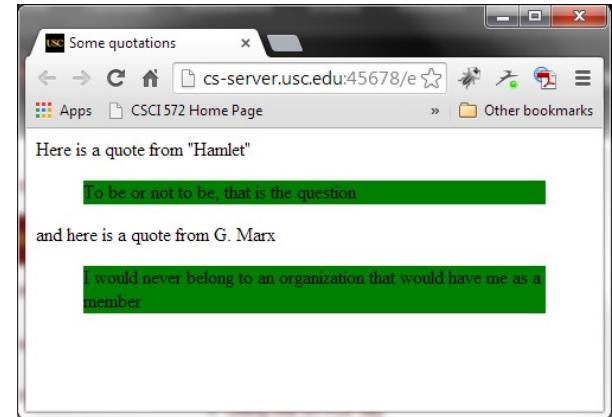
Browser Output



Firefox



Internet Explorer



Chrome

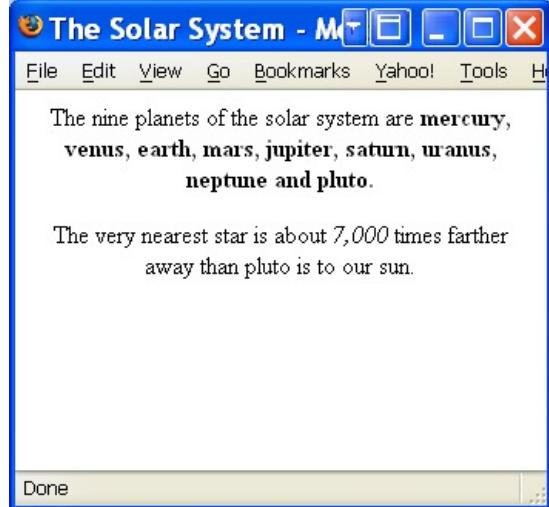
Example - Center Entire Document

```
<HTML>
<HEAD>
    <TITLE>The Solar System</TITLE>
    <STYLE type="text/css">
        BODY {text-align: center}
    </STYLE>
</HEAD>
<BODY>
    <P>The nine planets of the solar system are <B>mercury, venus, earth, mars, jupiter, saturn, uranus, neptune and pluto.</B></P>
    <P>The very nearest star is about <I>7,000</I> times farther away than pluto is to our sun.</P>
</BODY> </HTML>
```

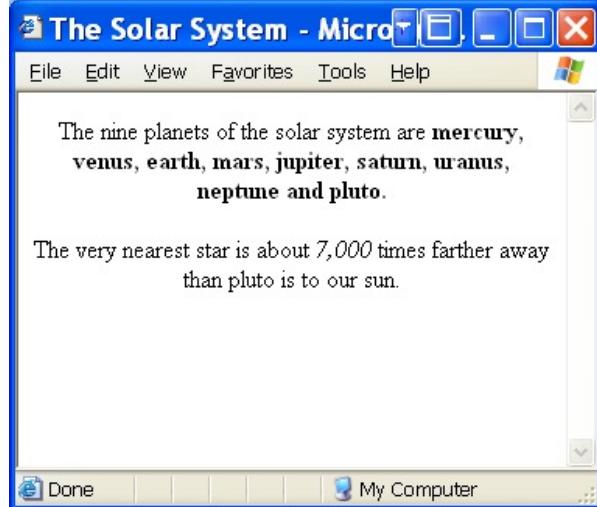
Note <CENTER> element is deprecated, removed in HTML5

```
<style>p {text-align:center;}</style>
```

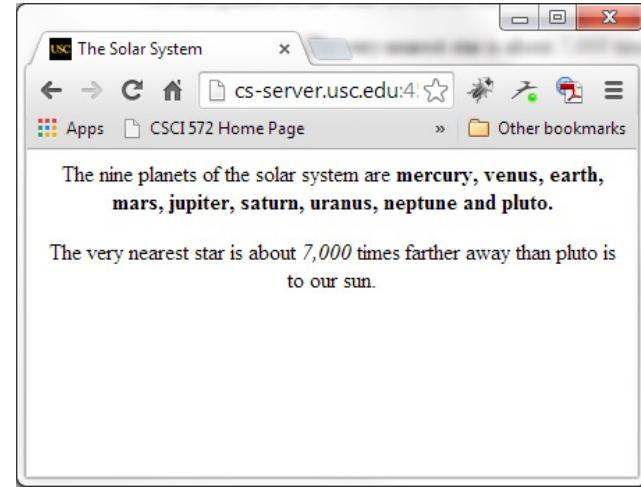
Browser Output



Firefox



Internet Explorer



Chrome

Option 3: Selectors (. #)

- The **CLASS attribute** assigns a name to one or more elements
- The **ID attribute** also can be used in a similar way.
- CLASS and ID can be used as a selectors of style properties

Ex1: P.redtext {color: #FF0000}

would be applied when <P> contains the CLASS name
<P CLASS="redtext">some text</P>

Ex2: .redtext{color: #FF0000}

applies to all elements that use CLASS=redtext

- Context sensitive selectors

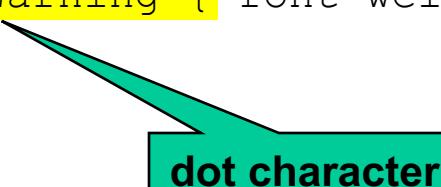
STRONG { color: #00FF00}

defines all occurrences of as green, but

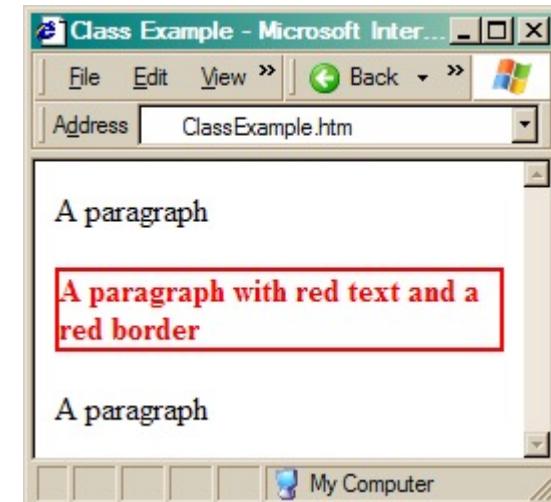
Class Attribute

- The **class attribute** is used for optional styles

```
<head>
  <style>
    .Warning { font-weight: bold; color: red;
               border:2px solid red; }
  </style>
</head>
<body>
  <p>A paragraph</p>
  <p class="Warning">A paragraph with
               red text and a red border</p>
  <p>A paragraph</p>
</body>
```



dot character

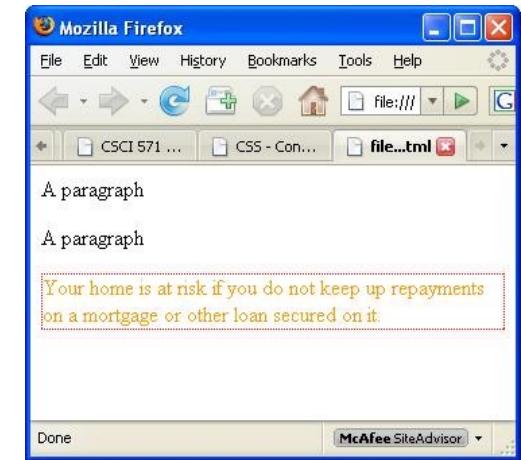
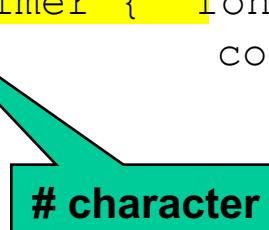


Id Attribute

- The **id attribute** is also used for optional styles
 - but it can only be used **ONCE** in the entire document
- Normally used for major document sections
 - Header, Footer, TopNav, LeftNav,
 - Content, etc.

```
<head>
    <style>
        #Disclaimer { font-size:medium;
                      color: #ff9900;
                      border:1px dotted red; }

    </style>
</head>
<body>
    <p>A paragraph</p>
    <p>A paragraph</p>
    <p id="Disclaimer">Your home is at risk if you
       do not keep up repayments on a mortgage
       or other loan secured on it.</p>
</body>
```

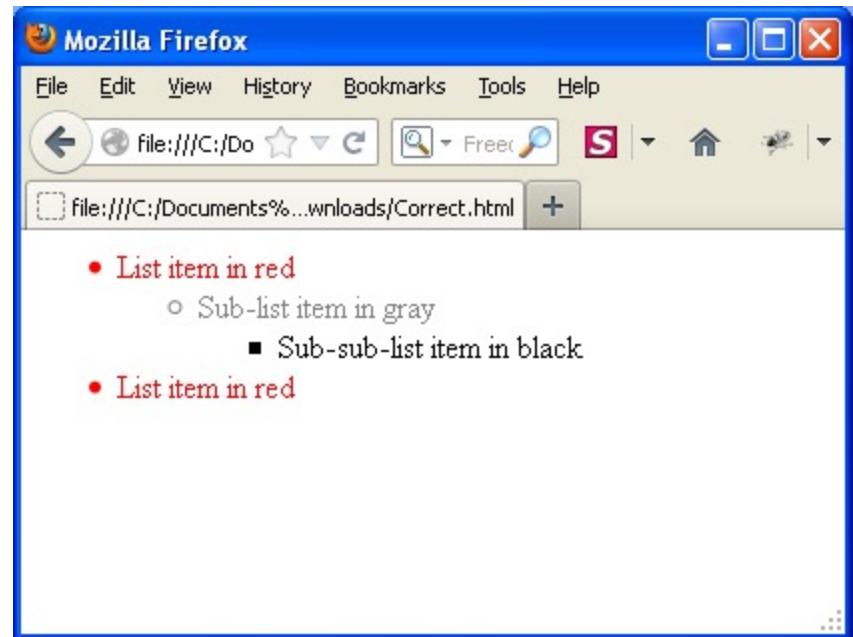


Specific Styles

- Styles aimed at specific tags

```
<head>
<style>
    ul { color: red; }
    ul ul { color: gray; }
    ul ul ul { color: black; }
</style> </head>

<body>
    <ul>
        <li>List item in red</li>
        <ul> <li>Sub-list item in gray</li>
            <ul> <li>Sub-sub-list item in black</li>
            </ul>
        </ul>
        <li>List item in red</li>
    </ul>
</body>
```

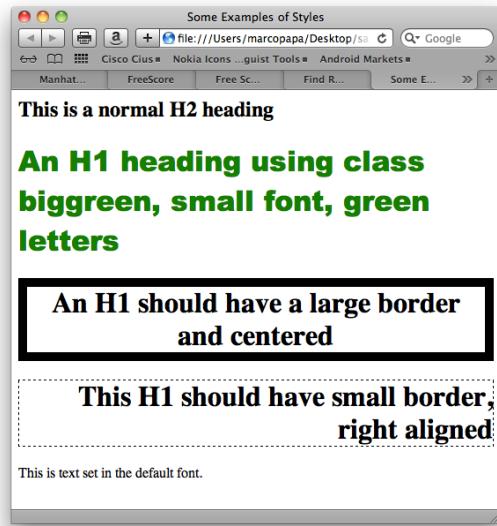


More <STYLE> Element Examples

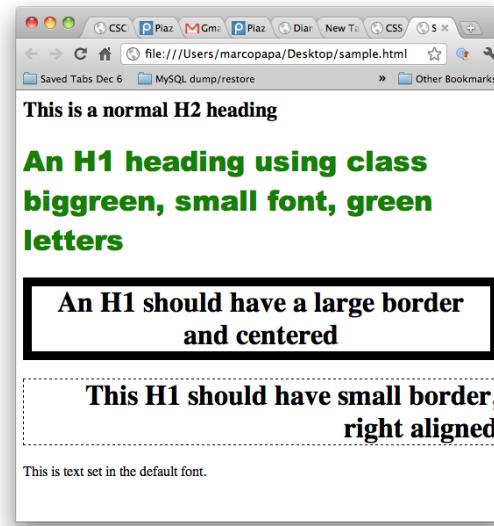
- Review
 - CLASS rule preceded by “.” and applied to multiple elements
 - ID rule preceded by “#” and applied to single elements
 - Values assigned to ID and CLASS are case sensitive

```
<HTML><HEAD><TITLE>Some Examples of Styles</TITLE>
<STYLE type=text/css>
.biggreen {font-family: "Arial Black"; font size:30; color:green}
H1.myh1 {border-width:10; border-style:solid; text-align:center}
#myid {border-width:1; border-style:dashed; text-
    align:right}</STYLE></HEAD><BODY>
<H2>This is a normal H2 heading</H2>
<H1 CLASS="biggreen">An H1 heading using class biggreen, small
    font, green letters</H1>
<H1 class=myh1>
    An H1 should have a large border and centered</H1>
<H1 id=myid> This H1 should have small border, right aligned
    </H1>This is text set in the default font.
</BODY></HTML>
```

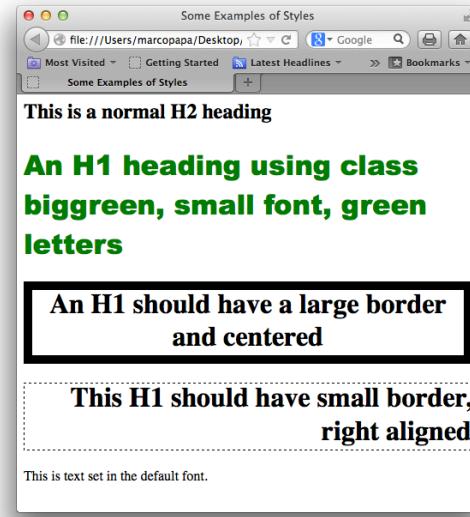
Browser Output



Safari



Chrome



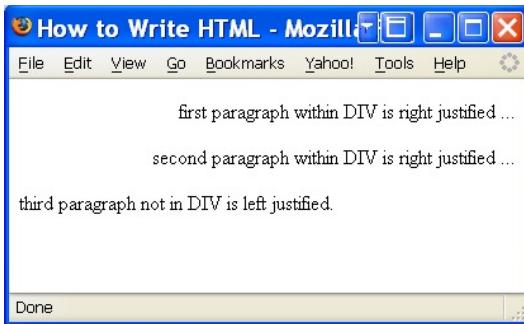
Firefox

HTML Block Tags

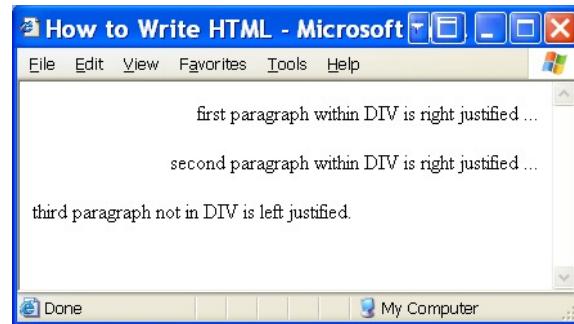
- <DIV> and tags have no initial presentation properties
 - **exception**, line break before and after a <DIV> tag
 - applies to **inline** elements (example:)
 - <DIV> applies to **block** elements (example: <p>)
- With CSS, properties such as text-align are “inherited” from the parent element

```
<HTML><HEAD><TITLE>How to Write HTML</TITLE>
<STYLE type="text/css">
DIV.mypars {text-align: right} </STYLE>  <BODY>
<DIV class="mypars">
<P>first paragraph within DIV is right justified ...
<P>second paragraph within DIV is right justified ...</DIV>
<P>third paragraph not in DIV is left justified.
</BODY></HTML>
```

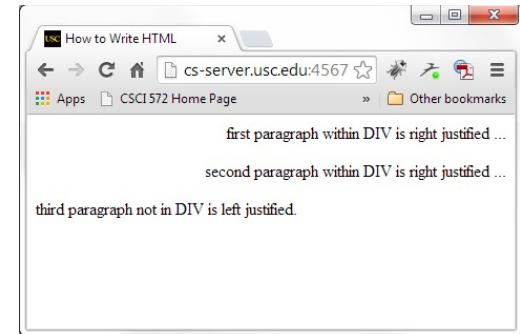
Browser Output



Firefox



Internet Explorer



Chrome

Precedence (specificity) of Style Settings

- The more precise a specification is, the higher the precedence, i.e., the more likely it is used
 - H1.mypars {text-align: center} [highest]
 - .biggreen {text-align: center} [next]
 - H1 { text-align: center} [lowest]
- a style for tag.class has higher precedence than one for .class, which has higher precedence than a style for the tag itself
- styles defined using a “style” attribute (inline) have highest precedence
- styles defined using <STYLE> element have next highest precedence
- styles defined in a separate file, e.g., special.css, have lowest precedence
- **Precedence** is guided by the **CSS cascading order**:

<https://www.w3.org/TR/CSS22/cascade.html#cascading-order>

Summary

- **Cascade**
 - At a very simple level this means that the **order of CSS rules matter**; when two rules apply that have equal specificity the one that comes **last** in the CSS is the one that will be used.
- **Specificity**
 - Specificity is how the browser decides which rule applies **if multiple rules have different selectors** but could still apply to the same element.
- **Inheritance**
 - Some CSS property values set on parent elements are **inherited by their child elements**, and some aren't.
- See Tutorial at:
https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

Composite Styles

- Many styles can be combined in one selector:

```
font: normal small-caps bold small/2em Verdana,  
Arial, Helvetica, sans-serif;
```

- Or you could specify:

```
font-family: Verdana, Arial, Helvetica, sans-serif;  
font-size:small;  
font-style:normal;  
font-variant:small-caps;  
font-weight:bold;  
line-height:2em;
```

- Common uses include:

- Font
- Background
- Margin & Padding

The DOCTYPE Directive

- One of the most important tags on your page
- Instructs modern browsers to work in 'standards compliant mode'
 - Your web page will look the same in **all** browsers
 - Browsers turn off their proprietary extensions
 - Fonts are rendered in the same way
 - For example, **font-size: small**, is rendered the same size on all browsers
- **For HTML 5:**

```
<!DOCTYPE html>
```
- **For HTML 4.01**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

DOCTYPE Examples

- **HTML4 Loose** compliance (minimum requirement) :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

- **HTML4 Strict** compliance:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

- **XHTML Transitional** compliance (less strict) :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional-dtd">
```

- **HTML5 :**

```
<!DOCTYPE html>
```

- Compound doctype declarations also exist for MathML and SVG
 - Math markup language, see <http://www.w3.org/Math/>

Style Sheet Media Types

- Enable authors to create documents for different media types
- **Example: all H1 displayed on a projector are in blue; when printed they are centered**

```
<HEAD>

<STYLE type=text/css media=projection
```

- **Example: add sound effects to anchors for speech output**

```
<STYLE type=text/css media=aural
```

Style Sheet Media Types

- Used in CSS3 for **media queries**
- See Responsive Design later in the course
- **Example: Target specific physical characteristics of device.**

```
<link rel="stylesheet" type="text/css" media="screen  
and (max-device-width: 480px)" href="min.css" />
```

- **Example: two equivalent pair of media queries**

```
<style>  
...  
@media all and (min-width:500px) { ... }  
@media (min-width:500px) { ... }  
</style>
```

Recognized Media types

- **all** - Suitable for all devices (**CSS3**)
- **braille** - Intended for braille tactile feedback devices.
- **embossed** - Intended for paged braille printers.
- **handheld** - Intended for handheld devices (typically small screen, monochrome, limited bandwidth).
- **print** - Intended for paged, opaque material and for documents viewed on screen in print preview mode. (**CSS3**)
- **projection** - Intended for projected presentations (
- **screen** - Intended primarily for color computer screens. (**CSS3**)
- **speech** - intended for speech synthesizers (**CSS3**)
- **tty** - Intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities.
- **tv** - Intended for television-type devices (low resolution, color, limited-scrollability screens, sound available).
- **3d-glasses** - Intended for 3D Glasses like Oculus VR and Google Cardboard.

Pseudo Elements and Pseudo Classes

- pseudo elements and pseudo classes are ways of assigning style properties independent of the document tree
- pseudo-classes
 - **:link** - a normal, un-visited link
 - **:visited** - a link the user has visited
 - **:hover** - a link when the user mouses over it
 - **:active** - a link the moment it is clicked
- Example, given the style definition

```
<html><head>  
<style>  
a:link {color:#FF0000;}          /* unvisited link (red) */  
a:visited {color:#00FF00;}      /* visited link (green) */  
a:hover {color:#FF00FF;}        /* mouse over link (pink) */  
a:active {color:#0000FF;}       /* selected link (blue) */  
</style></head><body>
```

- See

https://www.w3schools.com/css/css_pseudo_classes.asp

https://www.w3schools.com/css/css_pseudo_elements.asp

Pseudo Elements and Pseudo Classes (cont'd)

- Look up the meaning of these other pseudo elements and pseudo classes

<http://www.w3.org/TR/CSS2/selector.html#pseudo-elements>

- **pseudo classes**

- **:first-child**, Selects every <p> elements that is the first child of its parent
- **:hover**, Selects links on mouse over
- **:active**, selects the active link
- **:focus**, selects the input element which has the focus
- **:lang**, selects every <p> element with a lang attribute

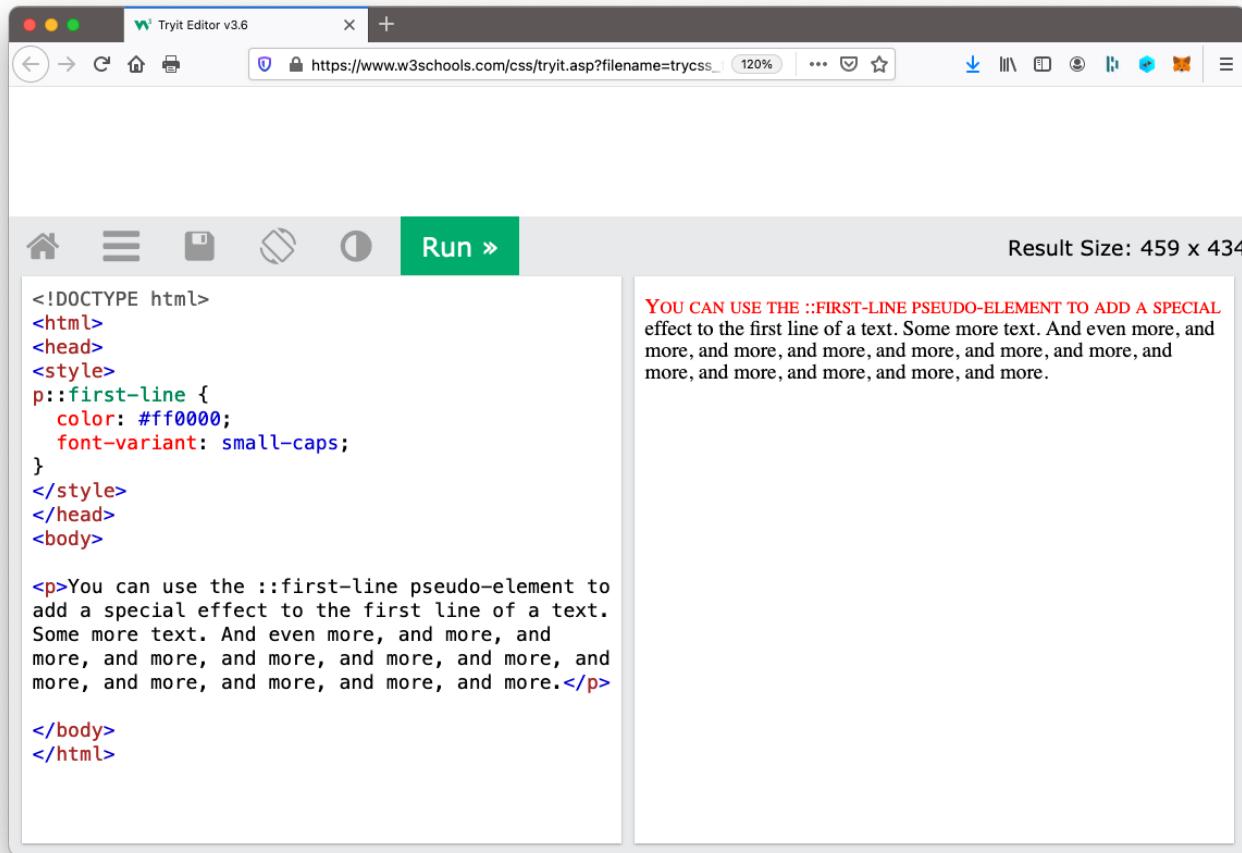
- **pseudo elements**

- **:first-line**, add a special style to the first line of a text
- **:first-letter**, add a special style to the first letter of a text
- **:before**, to insert some content before the content of an element
- **:after**, to insert some content after the content of an element

Pseudo Elements Example

- From W3Schools

https://www.w3schools.com/css/tryit.asp?filename=trycss_firstline



The screenshot shows a web browser window titled "Tryit Editor v3.6" with the URL "https://www.w3schools.com/css/tryit.asp?filename=trycss_firstline". The browser interface includes a toolbar with icons for home, search, and refresh, and a status bar indicating "Result Size: 459 x 434". The main area is divided into two panes. The left pane contains the following HTML and CSS code:

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-line {
    color: #ff0000;
    font-variant: small-caps;
}
</style>
</head>
<body>

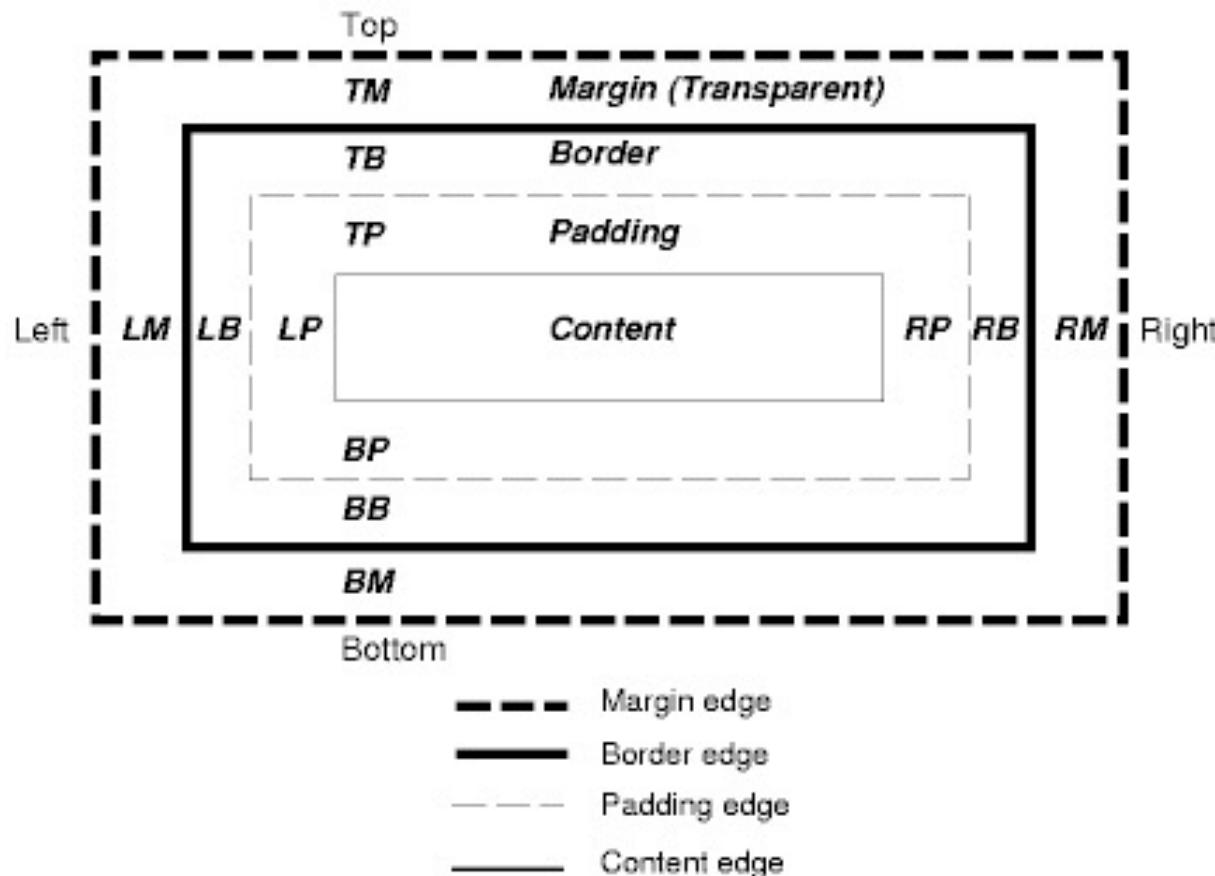
<p>You can use the ::first-line pseudo-element to
add a special effect to the first line of a text.
Some more text. And even more, and more, and
more, and more, and more, and more, and more, and
more, and more, and more, and more, and more.</p>

</body>
</html>
```

The right pane displays the rendered output of the code. The first line of the paragraph is colored red and has a small-caps font variant, while the subsequent lines are black. The text content is identical to the code in the left pane.

Box Model

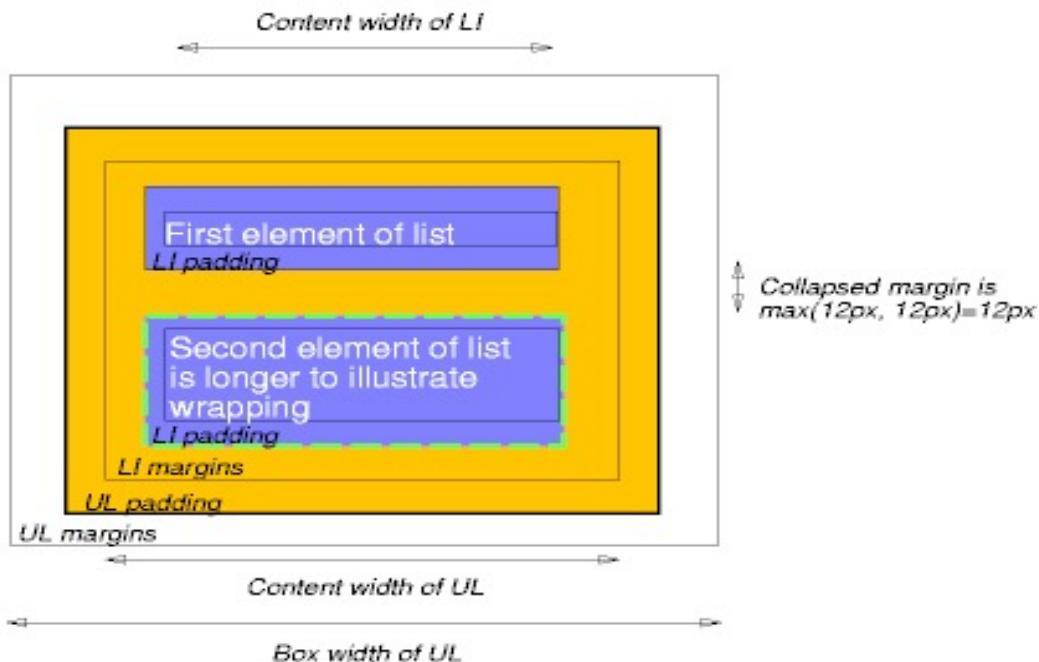
Each box has a **content** area (e.g., text, an image, etc.) and optional surrounding **padding**, **border**, and **margin** areas; the size of each area is specified by properties defined below. The following diagram shows how these areas relate, and the terminology used to refer to pieces of margin, border, and padding



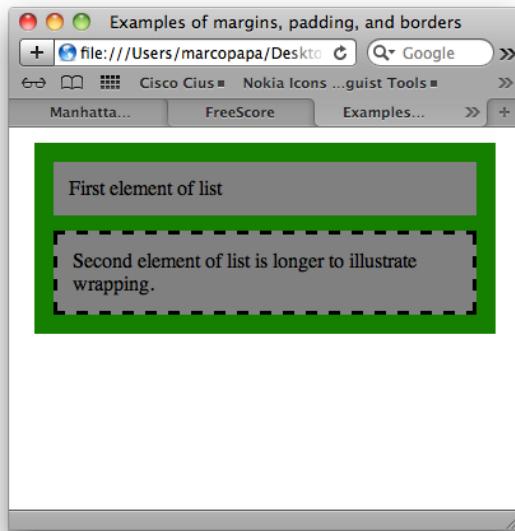
Example of Margins, Padding, Borders

```
<HTML><TITLE>Examples of margins, padding, and borders</TITLE>
<STYLE type="text/css">
UL { background: green;
margin: 12px 12px 12px 12px;
padding: 3px 3px 3px 3px; /* No borders set */ }
LI { color: black;      /* text color is black */
    background: gray; /* Content, padding will be gray */
    margin: 12px 12px 12px 12px;
    padding: 12px 0px 12px 12px; /* Note 0px padding right */
list-style: none /* no glyphs before a list item */
    /* No borders set */ }

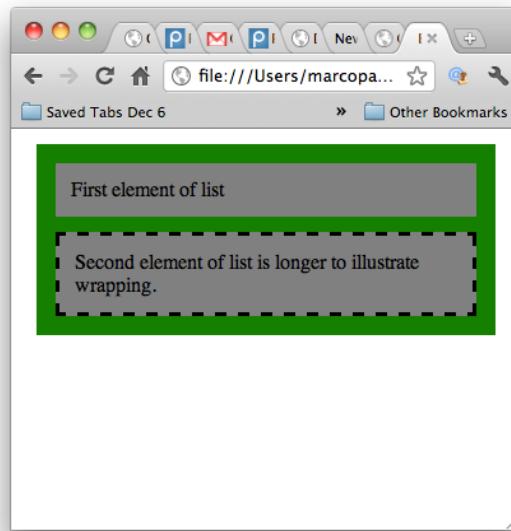
LI.withborder { border-style: dashed;
border-width: medium; /* sets border width on all sides */
border-color: black; } </STYLE> </HEAD>
<BODY> <UL> <LI>First element of list
<LI class="withborder">Second element of list is longer to
    illustrate wrapping.
</UL>
</BODY>
</HTML>
```



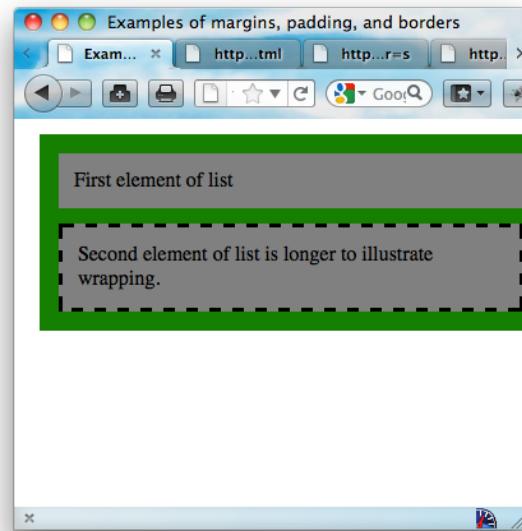
Browser Output



Safari



Chrome



Firefox

<https://csci571.com/examples/css/margins.html>

CSS Vendor Prefixes

- CSS vendor prefixes or CSS browser prefixes are a way for browser makers to add support for new CSS features during a testing and experimentation period.
- Browser prefixes are used to add new features that may not be part of a formal specification and to implement features in a specification that hasn't been finalized
- The CSS browser prefixes are:
 - Android: **-webkit-**
 - Chrome: -webkit-
 - Firefox: **-moz-**
 - Internet Explorer: **-ms-**
 - iOS: -webkit-
 - Opera: **-o-**
 - Safari: -webkit-
- E.g., before HTML 5, to set a rounded corner on a box one would have to write

```
-moz-border-radius: 10px 5px  
-webkit-border-top-left-radius: 10px;  
-webkit-border-top-right-radius: 5px;  
-webkit-border-bottom-right-radius: 10px;  
-webkit-border-bottom-left-radius: 5px;  
border-radius: 10px 5px;
```

Style Sheets Are Pervasive

- **espn.com:**
 - page.css:
- **cbsnews.com:**
 - videocontrol.css:
- **microsoft.com:**
 - <inline style sheet #1-#10>
- Style sheets are often used for "branding" & for changing the look-and-feel
- **Use Firefox Web Developer Inspector or Chrome Developer Tools to view CSS in Style Editor**

Reset CSS

- A **CSS Reset** is a short, often compressed (minified) set of CSS rules that *resets* the styling of all HTML elements to a consistent baseline.
- Every browser has its own default ‘user agent’ stylesheet, that it uses to make unstyled websites appear more legible.
 - For example, most browsers by default make links blue and visited links purple, give tables a certain amount of border and padding, apply variable font-sizes to H1, H2, H3 etc. and a certain amount of padding to almost everything.
- The goal of a reset stylesheet is to reduce browser inconsistencies in things like default line heights, margins and font sizes of headings, and so on.
- Reset styles quite often appear in CSS frameworks
- See <http://www.cssreset.com/> for several actual code examples

One Sample of css reset

```
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126 License: none (public domain) */

html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6,
p, blockquote, pre, a, abbr, acronym, address, big, cite, code, del, dfn,
em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt,
var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label,
legend, table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed, figure, figcaption, footer,
header, hgroup, menu, nav, output, ruby, section, summary, time, mark,
audio, video {

    margin: 0; padding: 0; border: 0; font-size: 100%; font:
    inherit; vertical-align: baseline; }

/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {display: block; }
body {line-height: 1; }

ol, ul {list-style: none; }

blockquote, q {quotes: none; }

blockquote:before, blockquote:after,
q:before, q:after {content: ''; content: none; }

table {border-collapse: collapse; border-spacing: 0; }
```

Style Sheet Examples

Style Sheet File

- Assume the style rules are stored in the file **mystyle.css**
- These rules re-define the <H1-H5> tags, set background to gray, alter the <P>, , tags, and name some color, font and text styles

```
h1, h2, h3, h4, h5 { color: red }
body { background-color: #cccccc; font-family:
    Arial, Helvetica, sans-serif }
p { line-height: 200% }
ul li { font-size: 70% }
.red { color: red }
.green { color: green }
.blue { color: blue }
#big { font-size: 120% }
#upper { text-transform: uppercase }
```

Example - Using the Style Sheet

- This example links to an external style sheet and refers to named selectors to produce different colored text and line spacing

```
<HTML>
<HEAD><LINK rel="stylesheet" href="mystyle.css"
    type="text/css">
</HEAD>
<BODY>
<H2>Let's have fun with layout!</H2>
<STRONG id="big">This text is big,</STRONG>
    whereas <SPAN id="upper" class="blue">this text
        is uppercase and blue!</SPAN>
<P>
I like documents to be double-spaced. It covers
    more space, which makes people think I've
        written a whole lot more.
</P>
</BODY> </HTML>
```

Browser Output



Firefox



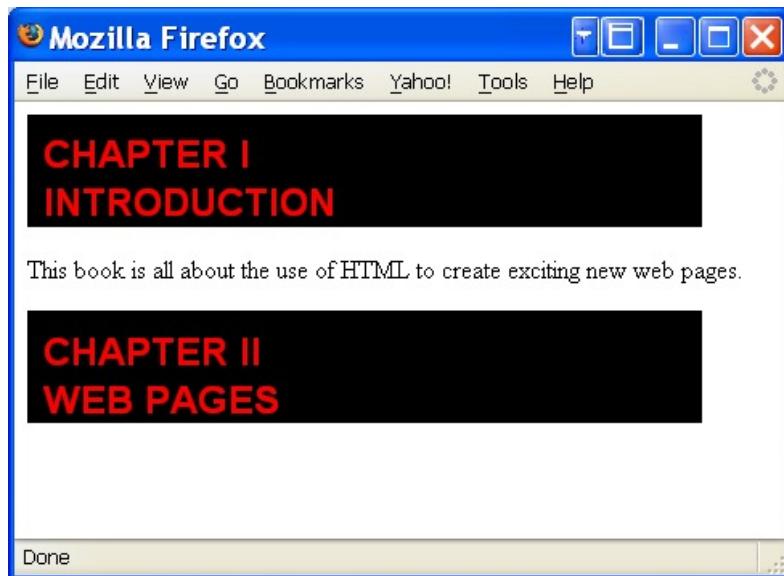
Internet Explorer

Example - Headlines with Graphic Backgrounds

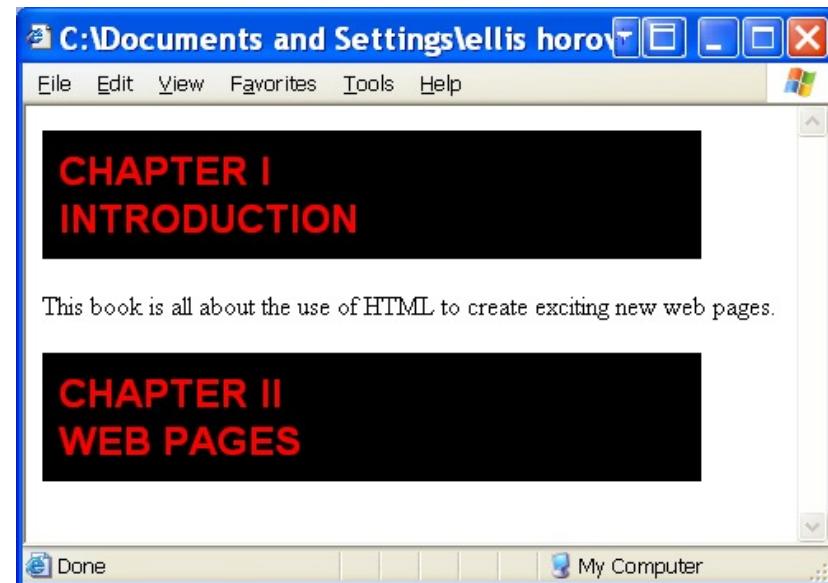
- Using CSS background property, one can create different titles using the same background

```
<HTML>
<HEAD><STYLE TYPE=text/css>
H1 {background:black url(black5.gif) no-repeat;
    font:bold 18pt helvetica, sans-serif; color:white;
    height:50px; width:400px; padding:10px; }
</STYLE></HEAD>
<BODY>
<H1><Font color=#FF0000>
    CHAPTER I<BR>
    INTRODUCTION</FONT></H1>
<P>This book is all about the use of HTML to create
exciting new web pages.
<H1><Font color=#FF0000>
    CHAPTER II<BR>
    WEB PAGES</FONT></H1> </BODY></HTML>
```

Browser Output



Firefox



Internet Explorer

Example – Creating Drop Caps

- A traditional form of book style

```
<HTML><HEAD>

<STYLE TYPE=text/css>
P {font: normal 10pt helvetica, arial, sans-serif; }
.dropcap {font:bold 300% times, serif; color:red;
  float:left; }

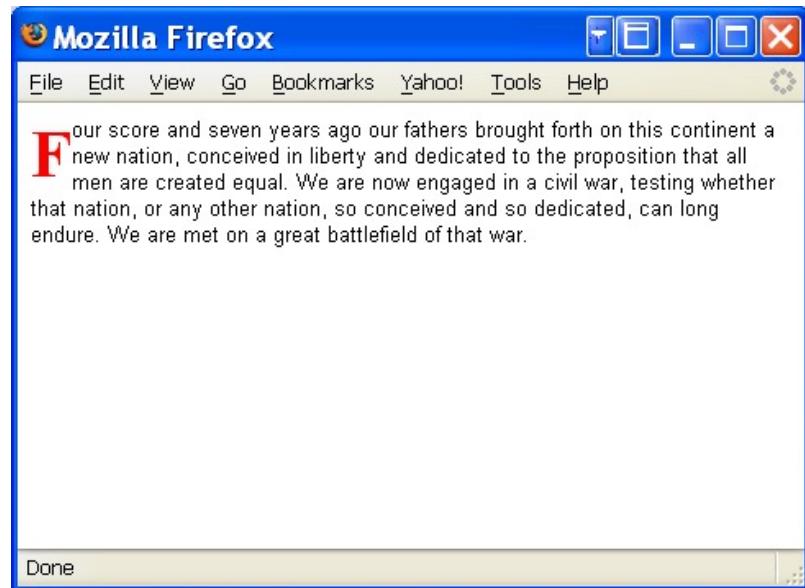
</STYLE></HEAD>

<BODY>

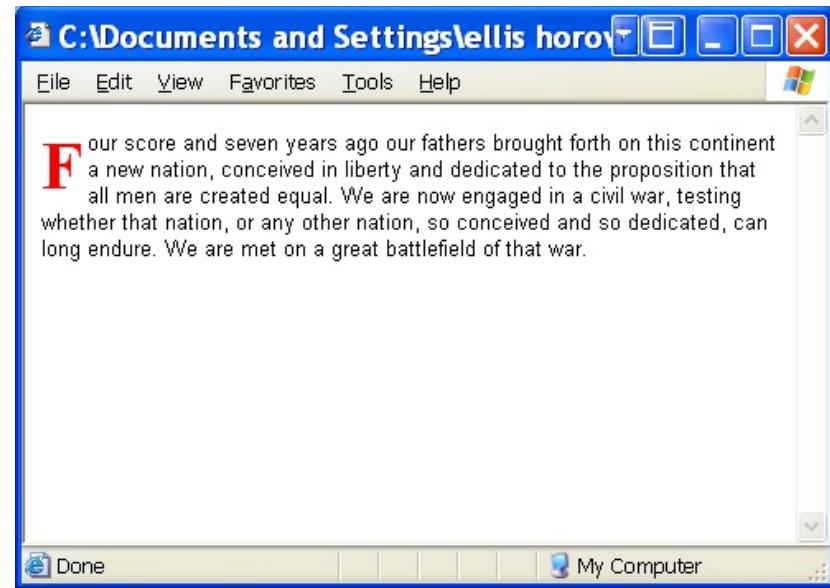
<P><SPAN CLASS=dropcap>F</SPAN>our score and seven
  years ago our fathers brought forth on this
  continent a new nation, conceived in liberty and
  dedicated to the proposition that all men are
  created equal. We are now engaged in a civil war,
  testing whether that nation, or any other nation,
  so conceived and so dedicated, can long endure. We
  are met on a great battlefield of that war.

</BODY></HTML>
```

Browser Output



Firefox



Internet Explorer

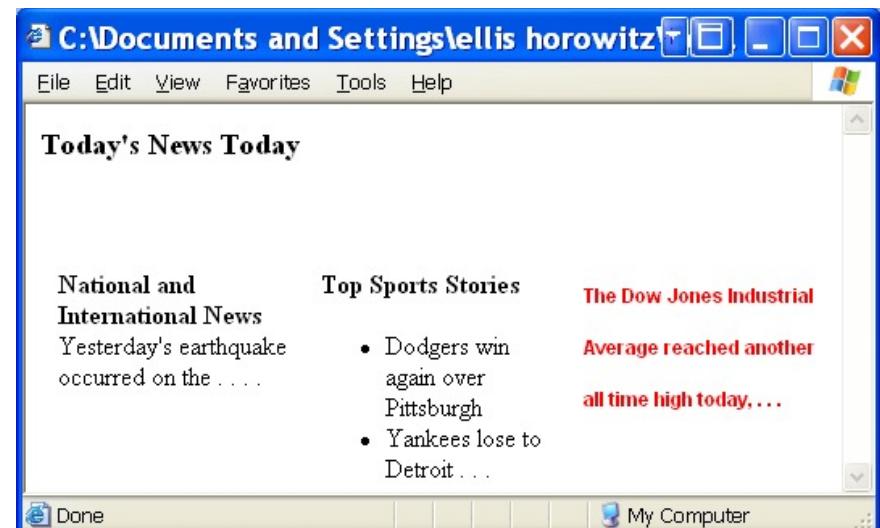
Creating Multiple Columns

```
<HTML><HEAD><STYLE TYPE=text/css>
#column1 {position : absolute; top:.5in; left: .1in;
           width:1.5in; font:12pt/14pt time,serif;}
#column2 {position : absolute; top:.5in; left: 1.8in;
           width:1.5in; font:12pt/14pt time,serif;}
#column3 {position : absolute; top:.5in; left: 3.5in;
           width:1.5in; font: bold 9pt/24pt helvetica, sans-serif;
           color:red;}
.relElement {position: relative; margin: 10px; }
</STYLE></HEAD><BODY> <H3>Today's News Today</H3>
<DIV CLASS=relElement>
<SPAN ID=column1><b>National and International News</B>
Yesterday's earthquake occurred on the . . . .
<SPAN ID=column2><B>Top Sports Stories</B><BR>
<UL><LI>Dodgers win again over Pittsburgh
<LI>Yankees lose to Detroit . . .
<UL></SPAN>
<SPAN ID=column3>The Dow Jones Industrial Average reached
another all-time high today, . . .
<SPAN>
</DIV></BODY></HTML>
```

Browser Output



Firefox



Internet Explorer

Note: Obsolete in HTML 5. Replaced by semantic elements

CSS3 Additional Features

- The W3C has announced the creation of four new modules for CSS Level 3.
- The modules add entirely new functionality and do not extend any previous CSS Level 1 or Level 2 functionality.
- They are based on proposals from Apple's WebKit team, and the current Working Drafts and Recommendations are available at the following URLs:

<http://www.w3.org/TR/css3-transitions/> (WD 2021)

<http://www.w3.org/TR/css3-transforms-1/> (2D/3D CR 2019)

<http://www.w3.org/TR/css3-mediaqueries/> (**REC**)

<http://www.w3.org/TR/css3-namespace/> (**REC**)

<http://www.w3.org/TR/css3-selectors/> (**REC**)

<http://www.w3.org/TR/css3-color/> (**REC**)

CSS 3 New Features

- The new CSS3 features can be organized according to the following categories
 - Borders
 - Backgrounds
 - Text Effects
 - Fonts
 - 2D Transforms
 - 3D Transforms
 - Transitions
 - Animations
 - Multiple Columns
 - User Interface

CSS3 Features supported by All modern Browsers

- Define “modern browsers”: **IE9+, Edge, Opera 10+, Firefox 3.5+, Chrome, Safari 3+**
- box-sizing
- border-radius
- box-shadow
- RGBA Colors
- HSLA Colors
- Multiple Backgrounds
- background-clip
- background-origin
- background-size
- Transforms
- Media Queries

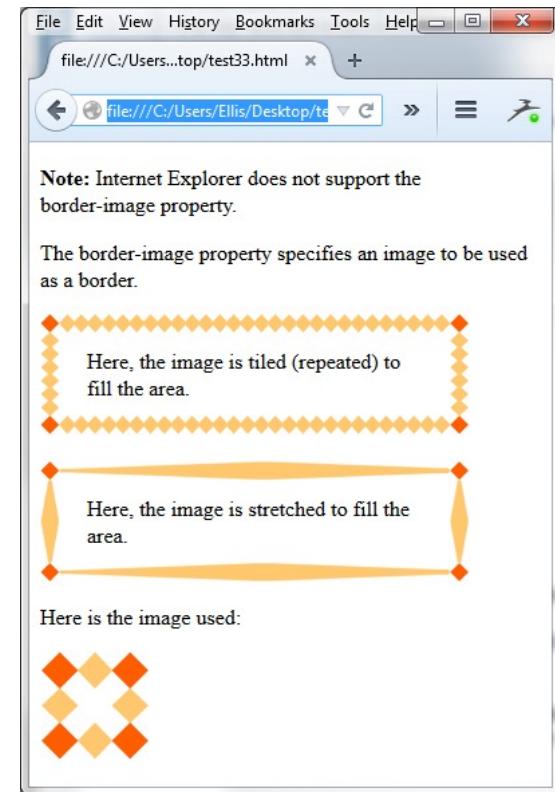
Border Images

```
<style>
div {
    border: 15px solid transparent;
    width: 250px;
    padding: 10px 20px; }

#round {
    -webkit-border-image: url(border.png) 30 30 round; /* Safari */
    -o-border-image: url(border.png) 30 30 round; /* Opera */
    border-image: url(border.png) 30 30 round; }

#stretch {
    -webkit-border-image: url(border.png) 30 30 stretch; /* Safari */
    -o-border-image: url(border.png) 30 30 stretch; /* Opera */
    border-image: url(border.png) 30 30 stretch; }

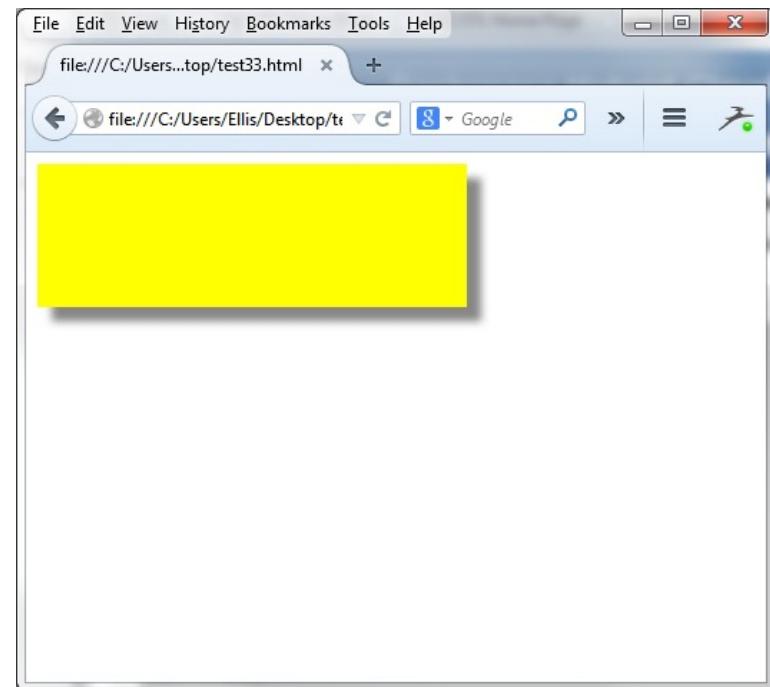
</style></head><body>
<p><b>Note:</b> Internet Explorer does not support the border-image property.</p>
<p>The border-image property specifies an image to be used as a border.</p>
<div id="round">Here, the image is tiled (repeated) to fill 'the area.</div><br>
<div id="stretch">Here, the image is stretched to fill the area.</div>
<p>Here is the image used:</p>
</body></html>
```



Prefix no longer needed: https://www.w3schools.com/cssref/css3_pr_border-image.asp

Box Shadow

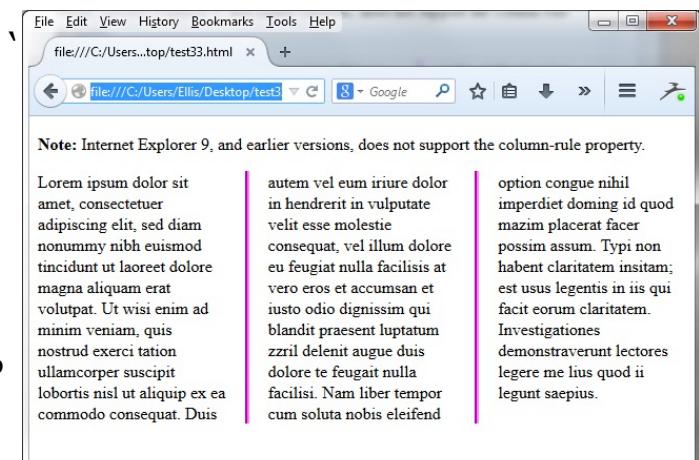
```
<style>
div {
    width: 300px;
    height: 100px;
    background-color: yellow;
    box-shadow: 10px 10px 5px #888888;
}
</style></head>
<body>
<div></div>
</body>
```



Multi-Column Layout

```
<!DOCTYPE html><html><head><style>
.newspaper {
-webkit-column-count: 3; /* Chrome, Safari, Opera */
-moz-column-count: 3; /* Firefox */
column-count: 3;
-webkit-column-gap: 40px; /* Chrome, Safari, Opera */
-moz-column-gap: 40px; /* Firefox */
column-gap: 40px;
-webkit-column-rule: 4px outset #ff00ff; /* Chrome, Safari, Opera */
-moz-column-rule: 4px outset #ff00ff; /* Firefox */
column-rule: 4px outset #ff00ff; }
</style></head><body>
<p><b>Note:</b> Internet Explorer 9, and earlier versions, does not support the column-rule property.</p>
<div class="newspaper">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis autem velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis. . . .
</div></body></html>
```

[Prefix no longer needed](https://www.w3schools.com/cssref/css3_pr_column-count.asp): https://www.w3schools.com/cssref/css3_pr_column-count.asp



Advanced CSS

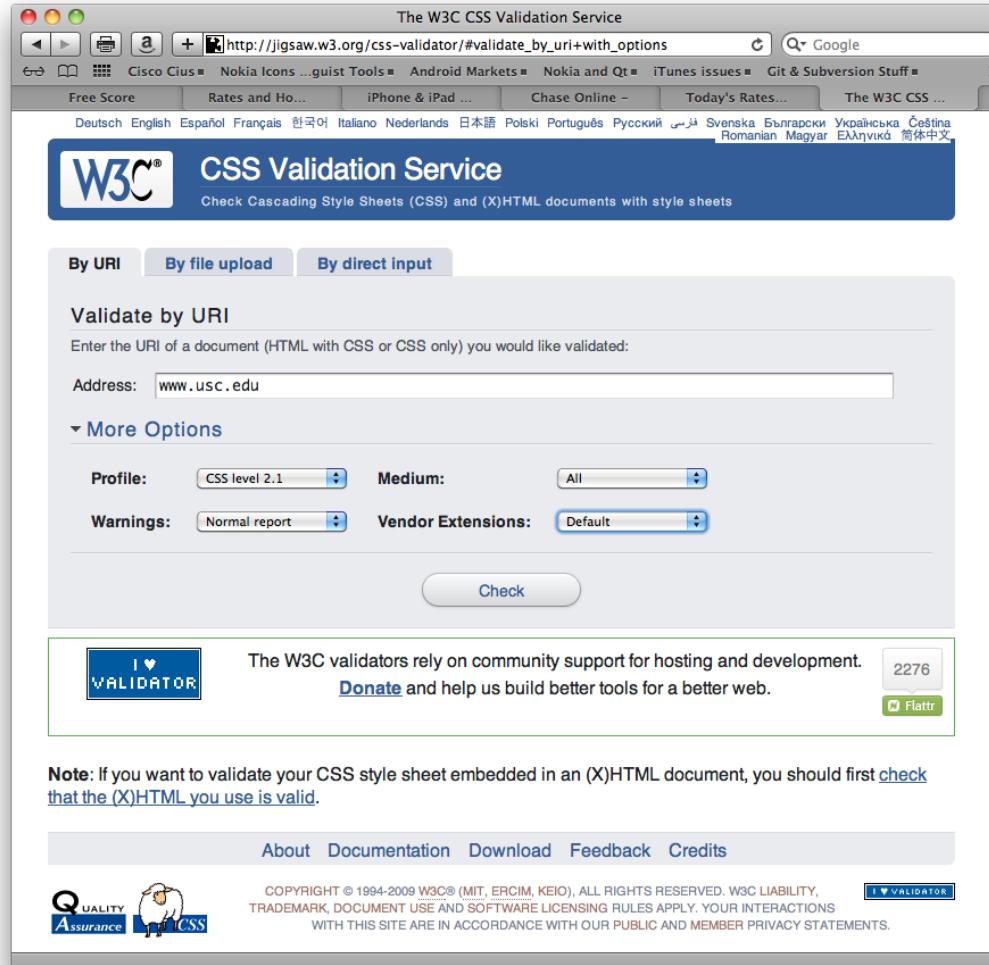
- Try the samples at W3Schools CSS Advanced:
<https://www.w3schools.com/css/>
 - CSS Rounded Corners
 - CSS Border Images
 - CSS Backgrounds
 - CSS Gradients
 - CSS Shadows
 - CSS Transitions
 - CSS Animations
 - CSS Tooltips
 - CSS Flexbox

Discussion Section

Validations and Sample CSS

W3C CSS Validation Service

- Available at:
<http://jigsaw.w3.org/css-validator/>



Discussion on Style

- For more examples of working with style sheets see
<http://www.w3.org/Style/Examples/007/>
<http://www.csszengarden.com/>



A screenshot of the CSS Zen Garden homepage version 215. The design is minimalist with a light blue background. It features a large circular logo with the text "CSS ZEN GARDEN" and "The Beauty of CSS Design". Below the logo is a "SELECT A DESIGN" button and a "VIEW ALL DESIGNS" link. The main content area contains text about the purpose of the garden and links to various design styles.

A screenshot of the CSS Zen Garden homepage version 215. The design is minimalist with a light blue background. It features a large circular logo with the text "CSS ZEN GARDEN" and "The Beauty of CSS Design". Below the logo is a "SELECT A DESIGN" button and a "VIEW ALL DESIGNS" link. The main content area contains text about the purpose of the garden and links to various design styles.

Here are eight versions of the same HTML page created using 8 different css files

CSS Implementations

The screenshot shows the CanIuse.com website interface. At the top, there's a navigation bar with links like 'About', 'News', and 'Index'. Below that is a header with the text 'Can I use' followed by a search bar and some settings options. The main content area is divided into several sections:

- CSS**: Lists features like @font-face, calc(), background-blend-mode, etc.
- HTML5**: Lists features like Audio element, Canvas, Color input type, etc.
- SVG**: Lists features like Inline SVG, SVG effects for HTML, SVG filters, etc.
- Other**: Lists features like async attribute, classList, Content Security Policy, etc.

At the bottom of the page, there's a footer with the text 'July 27, 2014 - Added indicator for disabled by ...' and a link to 'Compare browsers'.

caniuse.com home page



CSS 2.1 selectors example

CSS3 Animation Demos

- Apple Safari Technology Preview
<https://developer.apple.com/safari/technology-preview/>
- For examples of CSS3 Animation Demos see:
<https://www.internic.com/blog/30-creative-website-examples-of-css-animation/>
- More examples of CSS3 animations see
<https://www.creativebloq.com/web-design/examples-css-912710>
(see Type Terms example)

Codepen.io

- Codepen.io: “The best place to built, test and discover front-end code.”
- CSS Transform, animations on CODEPEN
<https://codepen.io/pork00chops/pen/vYXmbQW>
<https://codepen.io/michalporag/pen/MWjRJBO>
- SCSS on CODEPEN
<https://codepen.io/bennettfeely/pen/NWRmGYb>
- More CODEPEN examples:
<https://www.google.com/search?q=codepen+examples>

Lecture

JavaScript – Basics

What is JavaScript

- JavaScript is a "simple", interpreted, programming language with elementary object-oriented capabilities
- JavaScript has two distinct systems
 - client-side JavaScript runs on Web browsers
 - server-side JavaScript runs on Web servers
- JavaScript syntax resembles C, C++, and Java
- Developed in 10 days by **Brendan Eich** at Netscape in **May 1995** (now CEO at Brave Software after serving as cofounder and CTO at Mozilla)
- The original name was **Mocha**, chosen by Marc Andreessen, founder of Netscape
- Was renamed as **LiveScript**, then **JavaScript**
- See "A Short History of JavaScript":
https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

JavaScript is Embedded in HTML

- **In the <body>**

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
//the Javascript here produces content for the BODY on
loading
</SCRIPT>
</BODY>
</HTML>
```

- **or in the <head> as a deferred script**

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
//the Javascript here creates functions for later use
</SCRIPT>
</HEAD>
<BODY>
</BODY></HTML>
```

A Simple Example

```
<HTML>  
<HEAD>  
<TITLE>Simple Javascript</TITLE>  
</HEAD>  
<BODY>  
<H1>First Example of JavaScript</H1>  
<SCRIPT LANGUAGE="JavaScript">  
document.write("Last updated on " + document.lastModified + ". ")  
</SCRIPT>  
</BODY>  
</HTML>
```

Example 1: Browser Output



Another Example

```
<HTML>

<HEAD><TITLE>Computing Factorials</TITLE></HEAD>

<BODY>

<H1>Another Example of JavaScript</H1>

<SCRIPT LANGUAGE="JavaScript">

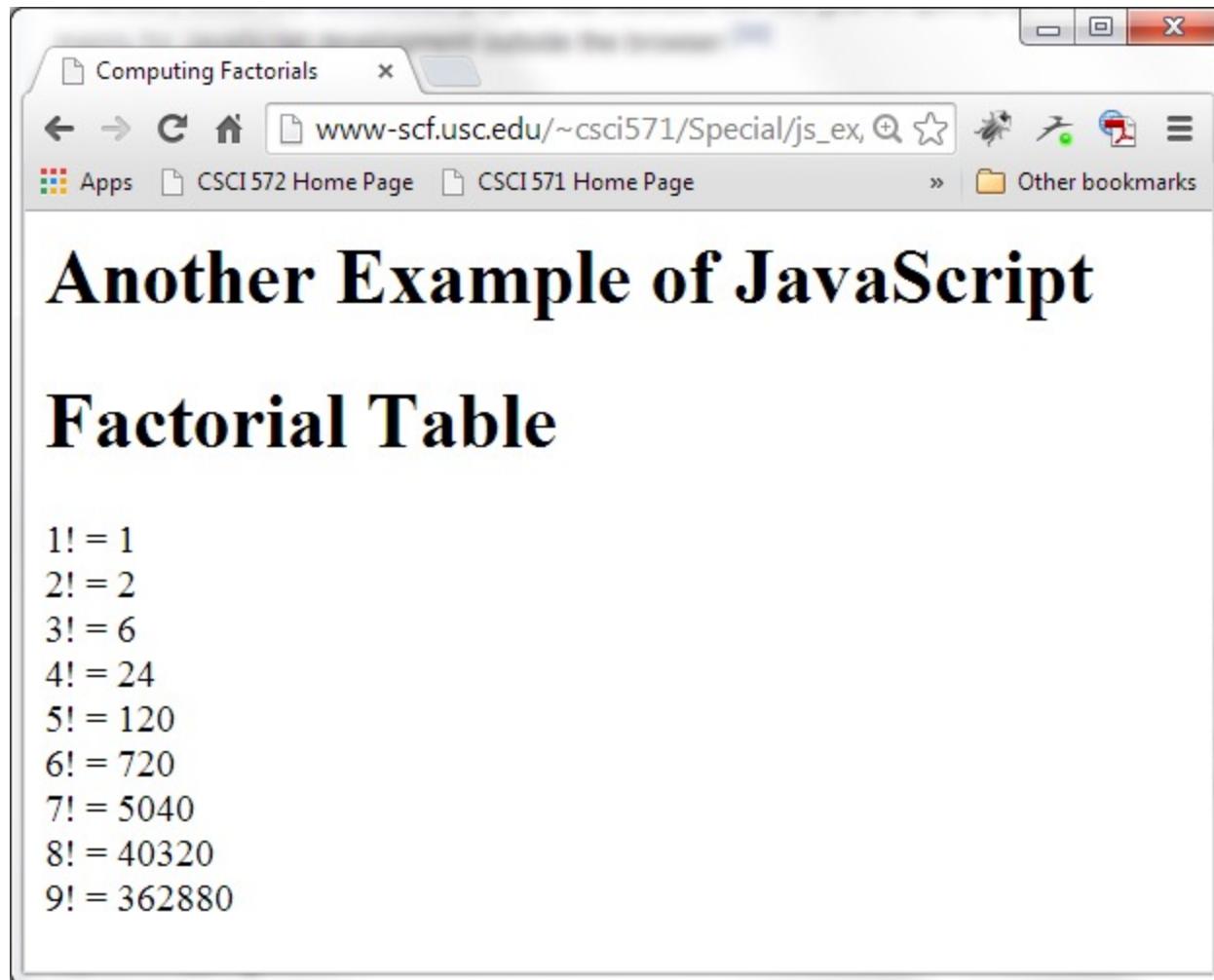
document.write("<H1>Factorial Table</H1>") ;
for ( i = 1, fact = 1; i < 10; i++, fact = fact * i) {
    document.write(i + " ! = " + fact);
    document.write("<BR>");
}

</SCRIPT>
</BODY>

</HTML>
```

Classical C for statement

Example 2: Browser Output



JavaScript has Event Handlers

```
<HTML>  
  
<HEAD><TITLE>Handling Events Example</TITLE></HEAD>  
  
<BODY>  
  
<H1>Handling Events in JavaScript</H1>  
  
<INPUT TYPE="button" VALUE="Click me"  
      onClick="alert('You clicked me')">  
  
</BODY>  
  
</HTML>
```



Some Common Events

- **Mouse Events**

- onclick user clicks an HTML element
- ondblclick user double-clicks an element
- onmouseover user moves the mouse over an HTML element
- onmouseout user moves the mouse away from an HTML element

- **Keyboard Events**

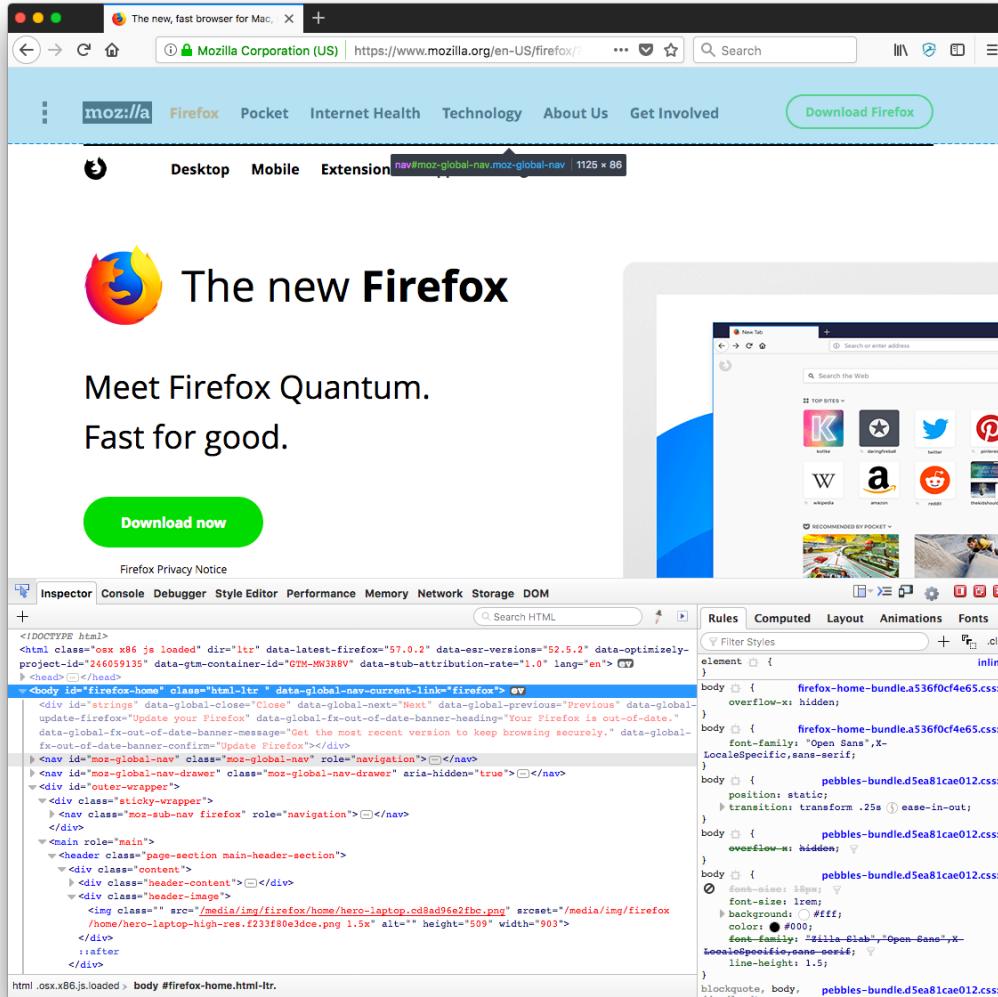
- onkeydown user presses a key
- onkeyup user releases a key

- **Object Events**

- onload browser has finished loading the page
- onunload a page has unloaded
- onresize a document view is resized
- onscroll a document view is scrolled

Debugging JavaScript

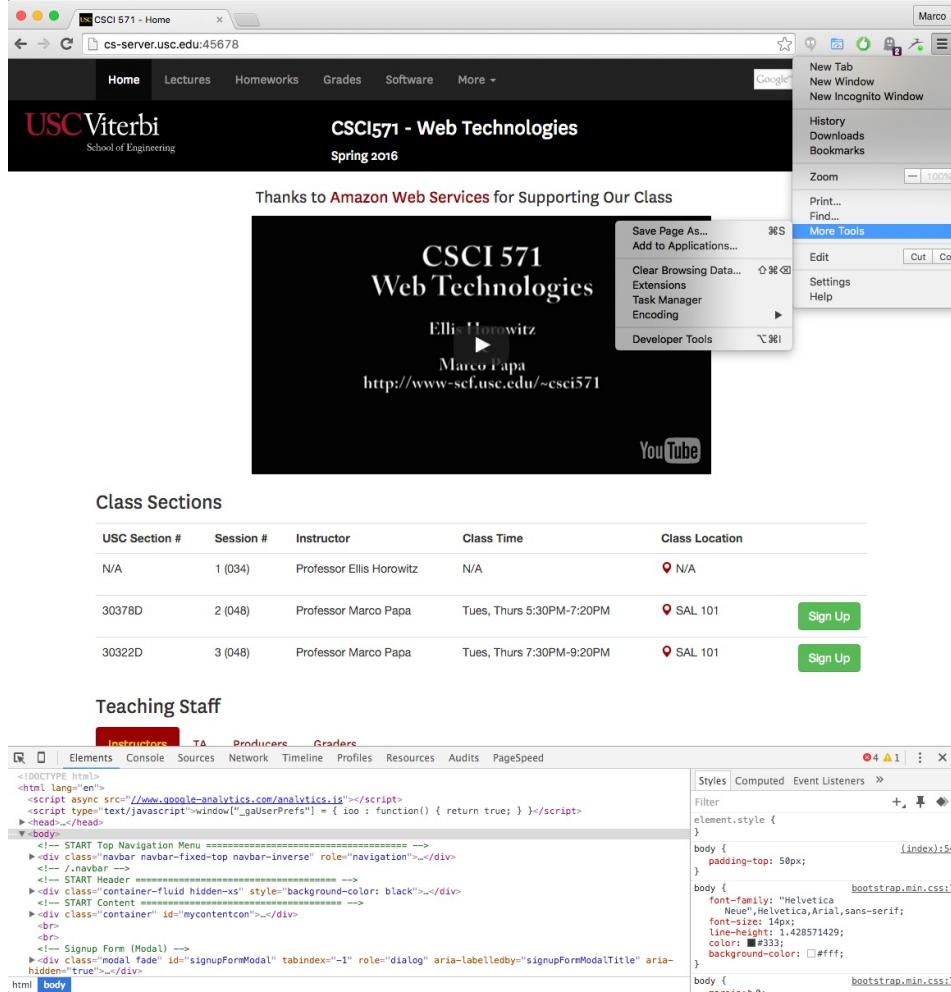
Firefox Developer Tools



- Built into Firefox
- To invoke, select Firefox “burger” menu > More tools > Web Developer Tools

Debugging JavaScript

Chrome Developer Tools



- The Chrome Developer Tools are built into Google Chrome.
- They provide deep access into the internals of the browser and their web application.

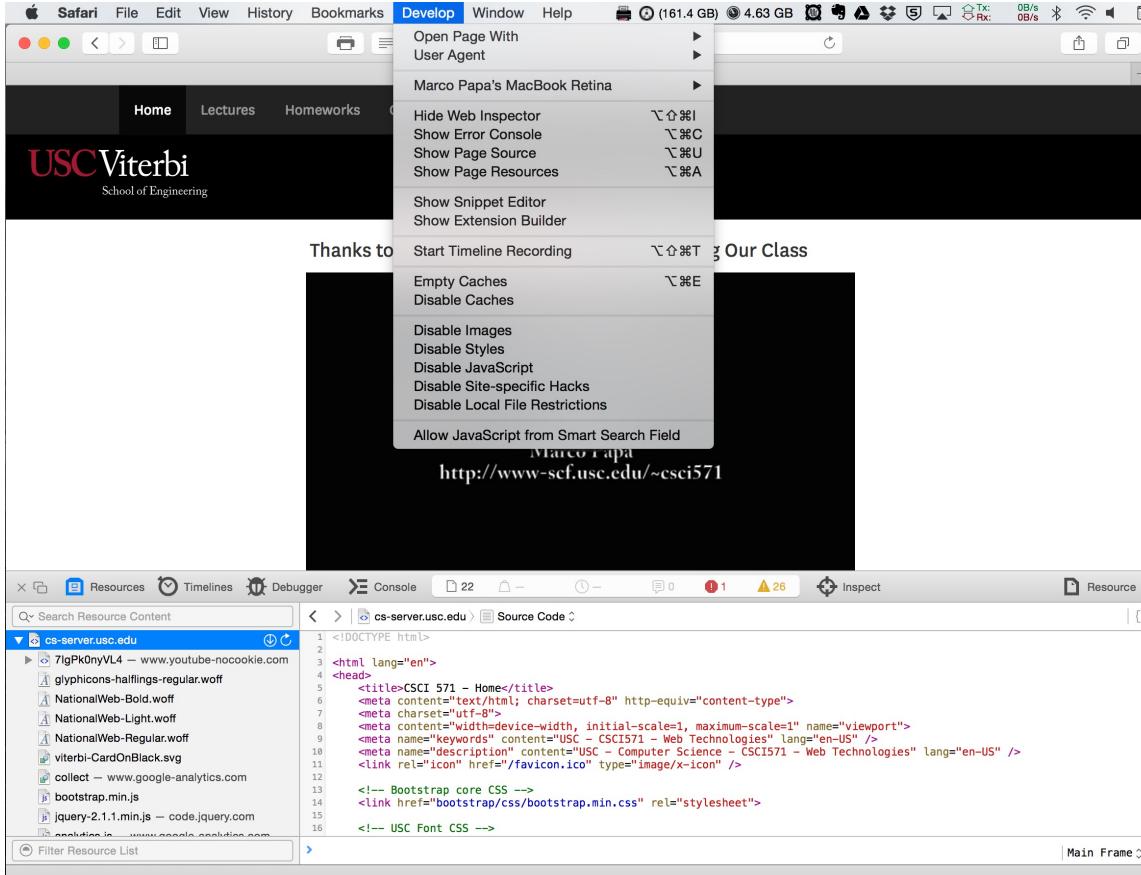
Use the DevTools to

- track down layout issues
- set JavaScript breakpoints
- get insights for code optimization.

- Go to Customize and Control Chrome > More Tools > Developer Tools

Debugging JavaScript

Safari Developer Tools



- The Safari Developer Tools are built into Safari.
- Developer Tools include
 - Web Inspector
 - Error Console
 - Page Source
 - Page Resources
 - Snippet Editor
 - Extension Builder
 - Debugger
 - Timelines
 - Timeline recording
- Turn on Develop menu:
Preferences > Advanced > check
“Show Develop menu in menu bar”

Debugging JavaScript

Internet Explorer F12 Developer Tools

The screenshot shows a Microsoft Internet Explorer window displaying the "Introduction to F12 Developer Tools" page from MSDN. The URL in the address bar is [msdn.microsoft.com/en-us/library/ie/gg589512\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ie/gg589512(v=vs.85).aspx). The page content includes a sidebar with links for F12 developer tools in IE9, such as "Getting started with the F12 developer tools" and "Using the F12 tools console to view errors and status". The main content area features a large heading "Introduction to F12 Developer T" (partially cut off) and a paragraph about F12 developer tools. A list of topics is provided at the bottom of the main content area.

F12 developer tools in IE9

- Getting started with the F12 developer tools
- Using the F12 tools console to view errors and status
- F12 developer tools console error messages
- Using the F12 developer tools to debug JavaScript errors
- Using F12 developer tools to debug HTML and CSS
- Using the profiler tool to analyze the performance of your code

Introduction to F12 Developer T

F12 developer tools is a suite of tools to help you build and debug your webpage.

This content refers to an older version of F12 tools. Please visit our [latest F12 tools documentation](#).

Writing great webpages requires coding expertise, as well as the right tools to find and debug errors that crop up. Windows Internet Explorer 9 provides a view of your rendered code, and F12 tools pages are interpreted on a code level by Internet Explorer 9. F12 tools also help identify and analyze the page such as links and image reports.

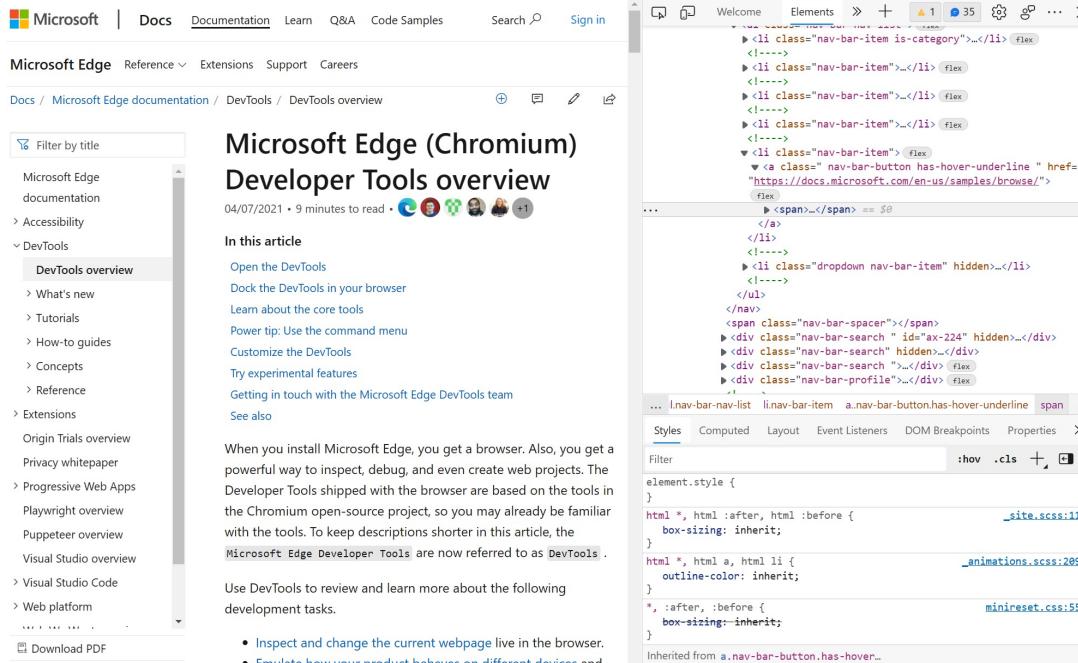
- [What F12 tools does](#)
- [Opening F12 tools](#)
- [Navigating your webpage and code in F12 tools](#)
 - [Change attributes and variables](#)
 - [Search for items](#)
 - [Select element by click](#)

A similar set of capabilities as Firebug and Chrome Developer Tools.

Available in IE9, IE10 and IE11.

Debugging JavaScript

Edge F12 Developer Tools



The screenshot shows the Microsoft Edge DevTools Elements panel. The main pane displays the DOM structure of the Microsoft Edge documentation page. A specific element, a link to 'DevTools overview', is selected. The bottom pane shows the Styles tab with CSS rules applied to the selected element, such as 'element.style { }' and 'html *, html :after, html :before { }'. The left sidebar lists various developer tools like Accessibility, DevTools, and Network.

Includes seven tools:

- DOM Explorer tool
- Console tool
- Debugger tool
- Network tool
- Performance tool
- Memory tool
- Emulation tool

To open and explore the DevTools, use one any of the following actions.

- Hover on any element on the webpage, open the contextual menu (right-click), and then select Inspect. This action opens the Elements tool.
- Select F12.
- Select Ctrl+Shift+I on Windows/Linux or Command+Option+I on macOS.

Documentation available at: <https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/>

What JavaScript Programs Can Do

- Write programs to perform any computation; it is equivalent in power to a general-purpose programming language
- But it is specifically designed for **manipulating web pages**
 - Control Web page **appearance** and **content** (this is its intended use)
 - Control the **Web browser**, open windows, test for browser properties
 - **Interact** with document **content**
 - Retrieve and manipulate all **hyperlinks**
 - **Interact** with the **user**, sensing mouse clicks, mouse moves, keyboard actions
 - Read/write client state with **cookies**

Limitations of Client-Side JavaScript

- [Was] Extremely difficult to explicitly draw graphics.
 - **This has been dramatically improved in the latest versions**
- No access to the underlying file system or operating system
- Unable to open and use arbitrary network connections
- No support for multithreading
- [Was] Not suitable for computationally intensive applications
 - **This has been dramatically improved in the latest versions**

JavaScript – Basics of the Language

- JavaScript is **case-sensitive**
 - sum, SUM and Sum are 3 different identifiers
 - HTML is NOT case-sensitive
- JavaScript **ignores** spaces, tabs, newlines
 - So it can be **minified**
- Semicolon is optional
 - but multiple statements on a line require a semicolon

```
i = 1; j = 2
```

- C and C++ style comments are supported

```
//comment to end of line
```

```
/* this can be a  
multiple line comment */
```

JavaScript Literals

- **literals** are fixed values, (not variables) that you literally provide in your script
 - **numbers**, 42, -5, 3.14159, -7.5E6
 - All numbers are treated as floating point
 - Octal (begin with a zero), 01234
 - Hexadecimal (begin with zero and x), 0xFF
 - **boolean**, true, false, (also null and undefined)
 - **strings**, any sequence of zero or more characters enclosed within single or double quotes
 - Examples
 - 'a single quoted string'
 - "a double quoted string"
 - """
 - "alert('Please Click OK')"

JavaScript Strings

- **Strings** are immutable, once created they can never be changed
- You can search a string and extract substrings, but you cannot modify a string
- “Immutable” means that once you instantiate the object, you can't change its properties.
- So, when calling methods on a string, JavaScript will return the modified string, but it won't change the initial string
- Now this doesn't mean that you can't assign a new string object to the str variable. You just can't change the current object that str references.
- One generally builds a string through concatenation

```
var newWebPage = ""  
  
newWebPage += "<HTML><HEAD>"  
  
newWebPage += "<TITLE>A Sample Page</TITLE></HEAD>"  
  
newWebPage += "<BODY>My Home Page</BODY>"  
  
newWebPage += "</HTML>"
```

Properties of Strings

- Strings have a length property

```
"Lincoln".length // result = 7  
"Four score".length //result = 10  
"One\nTwo".length // result = 7  
"".length // result = 0
```

- Some String methods

```
string.toLowerCase(); string.toUpperCase()
```

```
string.indexOf(searchString [, startIndex]) //returns index value  
of char within string where searchString begins
```

```
string.charAt(index) //returns the one char at position index
```

```
string.substring(indexA, indexB) //returns characters of string  
between indexA and indexB
```

JavaScript Escape Notation

- Escape sequences are used to embed special characters in a string

\b backspace

\t tab

\f form feed

\' single quote

\n newline

\\" double quote

\r carriage return

\\ backslash

- Example of escape characters in strings

```
msg = 'You\'re using an embedded single quote here.'
```

```
msg = "This is on the first line \n and this is on  
the second line."
```

```
msg = document.title + "\n" + document.links.length  
+ "links present"
```

JavaScript Reserved Words

JavaScript identifiers start with a letter, \$, or underscore followed by zero or more letters or digits;

JavaScript reserved words; you cannot use these reserved words as variables, labels, or function names

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

More JavaScript Words to Avoid

You should also avoid using the name of JavaScript built-in objects, properties, and methods including:

Array	Date	eval	function	hasOwnProperty
Infinity	isFinite	isNaN	isPrototypeOf	length
Math	NaN	name	Number	Object
prototype	String	toString	undefined	valueOf

Some notes:

1. The **NaN** property represents "Not-a-Number" value.
This property indicates that a value is not a legal number
2. **Infinity** is a numeric value that represents positive infinity.
-Infinity is a numeric value that represents negative infinity.
3. The **valueOf()** method returns the primitive value of the specified object.

JavaScript Variables

- Variables should be declared, but not their type

```
var i, sum;      //declaration
```

```
var zero = 0;    //declaration and initialization
```

```
var myName = "Ellis"
```

- The **type** of value a variable can hold during execution may change.

- **Scope**

- Any variable outside a function is a **global variable** and can be referenced by any statement in the document

- Variables declared in a function as "var" are **local** to the function

- if var is omitted, the variable becomes global

- In a multi-frame or multi-window set up of the browser, scripts can access global variables from any other document currently loaded

JavaScript Data Types

JavaScript has a limited set, with some notable ones lacking, e.g., integer, and some notable ones not usually found in programming languages, e.g., function

Type	Example	Description
String	"a string"	A series of characters inside quote marks
Number	123.45	Any number not inside quote marks
Boolean	true	A logical true and false
Null	null	Completely devoid of any value, not a number, not a string, different than 0 in C/C++
Object		All properties and methods belonging to the object
Function		A function

JavaScript Arrays

- Though not an official data type, arrays are included in the language using a traditional array notation, i.e., **square brackets**
- However, they differ from conventional arrays in many ways
- **array properties**
 - one dimensional, **indexed** from **zero**
 - array elements can contain any type of data including references to other arrays, to objects, to functions
 - array elements can have different types
- An **array literal** is a list of zero or more expressions, each of which represents an array element, enclosed in square brackets ([]), e.g.
 - `var coffees = ["French Roast", "Columbian", "Kona"];`
 - `var fish = ["Tuna", , "Cod"];` (one empty element)
 - `var myArray = ["Richard", 10, getPhoto()];` (string, number, function)
 - `var items = [[1,2],[3,4],[5,6]];` (two-D array, items[0][0] is 1)

More on Arrays

- Every array has a length property
- The length property is the **largest integer** property name in the array **plus one**

```
var myArray = [];
myArray.length          //0
myArray[100000] = true;
myArray.length          //100001
```

- Arrays are **sparse**, in the above example only one index is allocated
- JavaScript does NOT provide a way to declare the size (dimension) of an array, but we can add one

```
Array.dim = function (dimension, initial) {
    var a = [], i;
    for (i = 0; i < dimension; i += 1) {
        a[i] = initial; }
    return a; }

var myArray = Array.dim(10,0); //makes an array of ten zeros
```

More on Arrays

- There are many ways to iterate over an array
 - for loop; `for (i=0; i < len; i++) { . . . }`
 - for in loop; `for (x in person) { . . . }`
 - while loop; `while (condition) { . . . }`
- There are many built-in methods for working with arrays, here are just a few:
 - `concat()`, joins two or more arrays
 - `indexOf()`, search the array for an element and return its position
 - `pop()`, remove the last element
 - `push()` add a new element at the end
 - `reverse()`, reverses the order of elements
- See also *JavaScript Guide*:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Predefined_Core_Objects
http://www.w3schools.com/jsref/jsref_obj_array.asp

Arrays and Objects are Semantically Identical

- The **typeof()** function returns a string which is the type of its argument ("number", "string", "boolean", "object", "function", "undefined")
- In JavaScript objects and arrays are really identical, **typeof(array) = typeof(object) = object**
- JavaScript does **NOT** support associative arrays, e.g.

```
var person = new Array();
person["firstname"] = "John";
person["age"] = 41;
```

- `person["firstName"]` returns "John", but `person[0]` returns undefined and `person.length` returns 0
- Array elements are object properties in the same way that `toString` is a property, but trying to access an element of an array as follows throws a syntax error, because the property name is not valid:

```
console.log(arr.0); // a syntax error
```

- There is nothing special about JavaScript arrays and the properties that cause this. JavaScript properties that begin with a digit cannot be referenced with dot notation; and must be accessed using bracket notation.

Objects

- An object literal is a list of zero or more **pairs** of **property names** and associated **values** of an object, enclosed in curly braces ({}), e.g.

```
var person = {firstName:"John",
              lastName:"Doe",
              age:50,
              eyeColor:"blue"};
```

- Object **properties** are like JavaScript **variables**
- the “dot” operator is used to access the value of an object’s property or to assign it a value, e.g.

```
lname = person.lastName          // returns "Doe"
person.lastName = "Smith";
```

- Objects can be nested within objects, e.g.

```
var myHonda = {color: "red",
               wheels: 4,
               engine: {cylinders: 4,
                         size: 2.2}
             };
```

Object Constructors

- It is often useful to define an “object type” that can be used multiple times to create object instances having the same structure
- To do this one creates an object constructor, which is a JavaScript function that is called with the **new** operator, e.g.

```
function cat(name, meow) { this.name = name;  
    this.talk = function() { alert(this.name + " says " + meow) } }  
cat1 = new cat("felix", "purr");  
cat1.talk();  
cat2 = new cat("ginger", "hiss");  
cat2.talk();
```

- `cat()` is an object constructor with properties and methods declared inside

Predefined JavaScript Objects

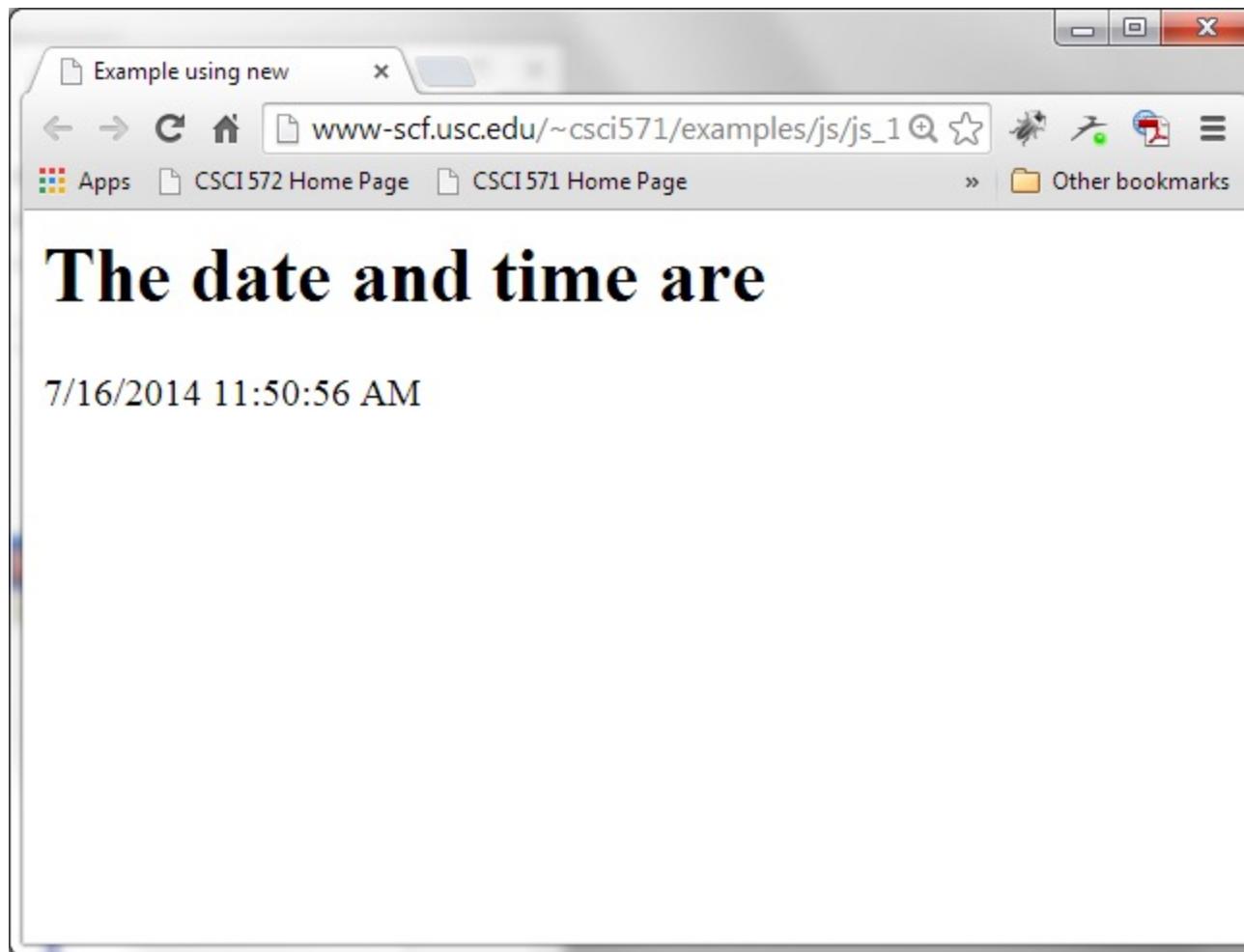
- There are a set of predefined objects that include:
 - **Array object**, we have already seen
 - **Boolean object**, a wrapper around the primitive Boolean data type,
 - `var booleanObject = new Boolean(value);`
 - Any object whose value is not (`undefined`, `null`, `0`, `Nan` or the empty string, including a Boolean object whose value is `false`), evaluates to `true`
 - **Date Object**, like Java it stores dates as the number of milliseconds since Jan. 1, 1970, 00:00:00
 - `var Xmas95 = new Date("December 25, 1995");`
 - `Xmas95.getMonth()` returns 11, `Xmas95.getFullYear()` returns 1995
 - **Function object**
 - `Var functionObjectName = new Function([arg1, ..., argn], functionbody);`
 - **Math object** includes properties and methods for mathematical constants, e.g., `sin()`, `cos()`, `ceil()`, `floor()`
 - **RegExp object** (discussed later)
 - **String object** (discussed later)

Example Using Date Object

```
<HTML>
<HEAD>
<TITLE>Example using new</TITLE>
<SCRIPT LANGUAGE=JavaScript>
function outputDate() {
    var d = new Date(); //creates today's date and time
    document.write(d.toLocaleString()); }
                                // converts a date to a string
</SCRIPT></HEAD>
<BODY>
<H1>The date and time are</H1>
<SCRIPT LANGUAGE=JavaScript>
    outputDate();
</SCRIPT>
</BODY>
</HTML>
```

See examples at: <https://csci571.com/examples.html#js>

Example 4: Browser Output



JavaScript Popup Boxes

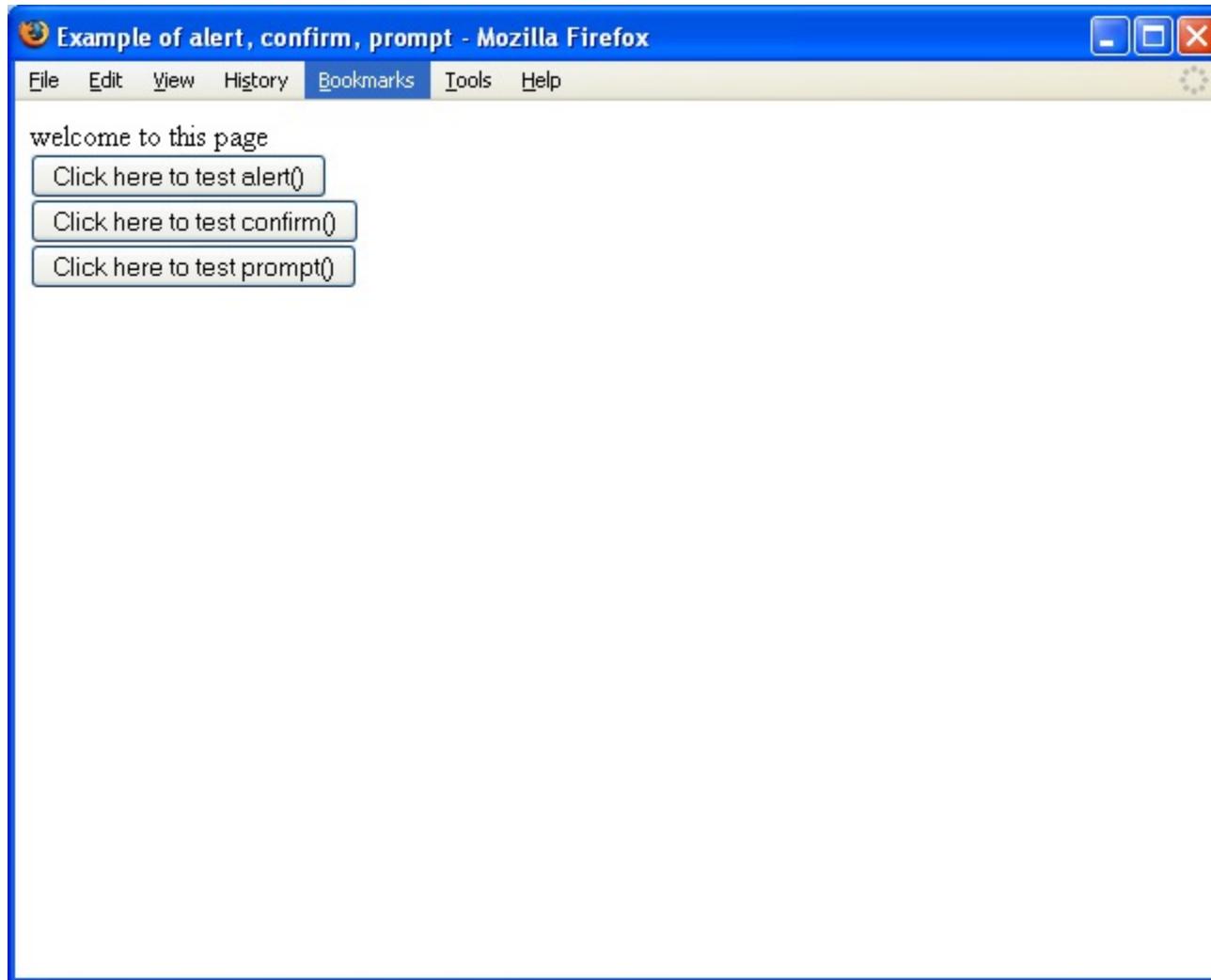
alert(), **confirm()**, and **prompt()**

```
<HTML><HEAD>
<TITLE>Example of alert, confirm, prompt</TITLE>
<SCRIPT LANGUAGE=JavaScript>
function alertUser() {
    alert("An alert box contains an exclamation mark");
}
function confirmUser() {
    var msg = "\n please confirm that you want\n" +
              "to test another button?";
    if (confirm(msg))
        document.write("<h2>You selected OK</h2>");
    else document.write("<h2>You selected Cancel</h2>"); }
function promptUser() {
    name1=prompt("What is your name? ", " ");
    document.write("<h2>welcome to this page " + name1 + "</h2>");
}
</SCRIPT>
</HEAD>
```

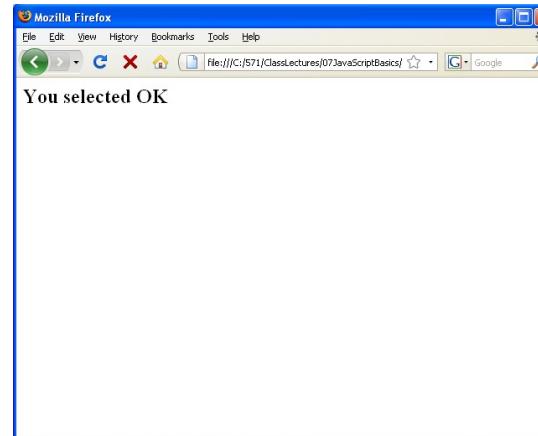
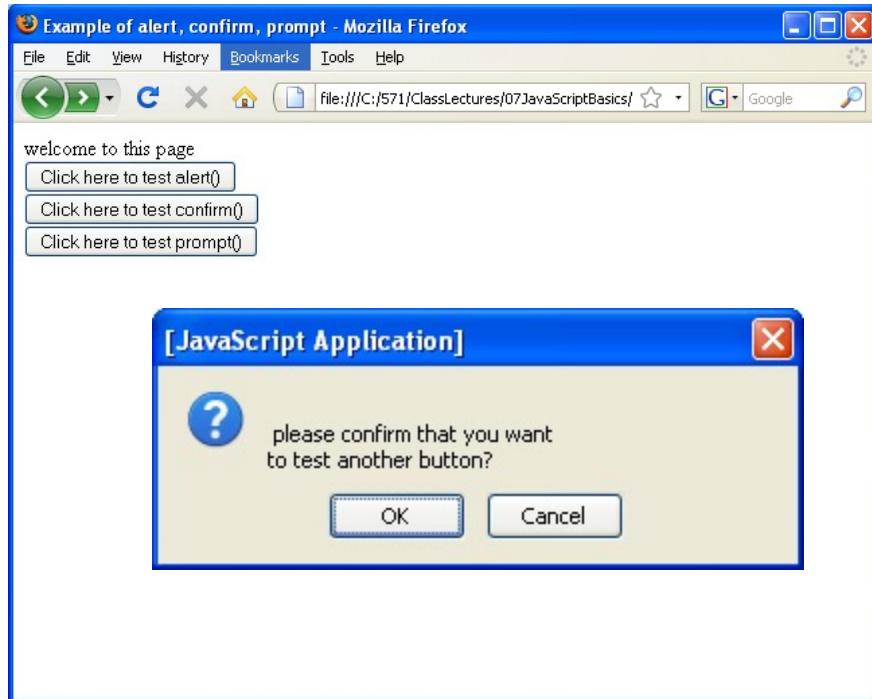
Using alert() , confirm() , and prompt()

```
<BODY>welcome to this page<br>
<FORM>
<INPUT TYPE=button VALUE="Click here to test alert()" 
    onClick="alertUser()"><BR>
<INPUT TYPE=button VALUE="Click here to test confirm()" 
    onClick="confirmUser()"><BR>
<INPUT TYPE=button VALUE="Click here to test prompt()" 
    onClick="promptUser()">
</FORM>
</BODY>
</HTML>
```

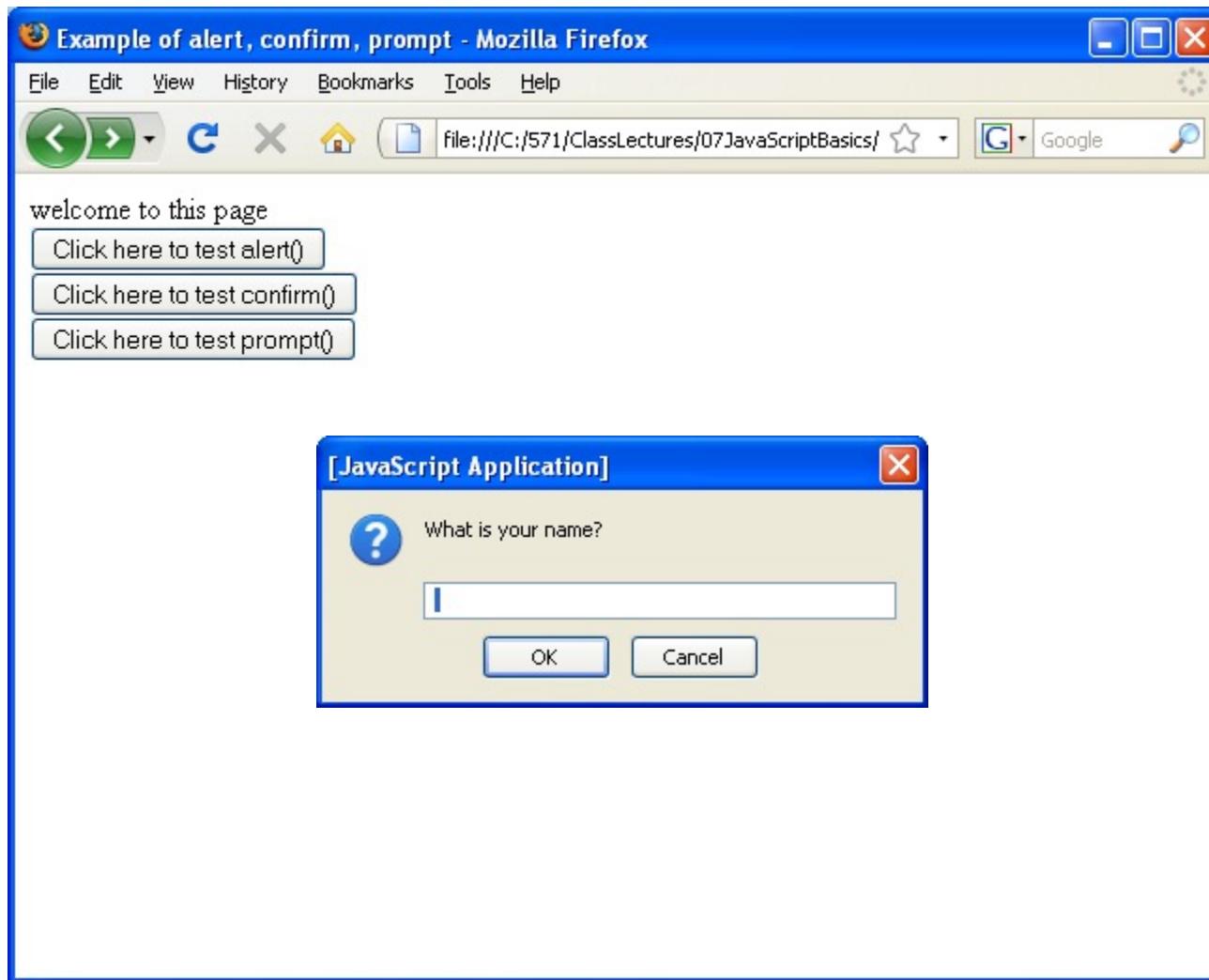
Example 5: Browser Output



Clicking on confirm()



Clicking on prompt()



Final Thoughts - Common Mistakes

1. **Undefined may not be null**

- In JavaScript something that has not been assigned to is not null, but undefined. Undefined is different from null when using != = but not when using the weaker != because JavaScript does some implicit casting in the later case
- For details see
<http://javascript.about.com/od/hintsandtips/a/Null-And-Undefined.htm>

2. **You cannot overload a function**

- If you try to define two different functions with the same name but with different arguments, assuming the proper function will be called when the number of arguments match (this is overloading), then your assumption is incorrect. JavaScript will simply use the latest-defined version of the function and call it;
- If a parameter is omitted, it is undefined

3. **Undeclared variables are global**

- If a variable is NOT declared using var, then it is global. Two variables of the same name, both undeclared will create conflicts that are hard to debug

Netscape JavaScript Versions

Netscape Browser	JavaScript Version	Comments
2.0	1.0	a.k.a. LiveScript
3.0	1.1	Adds images, arrays, applets, plug-ins
4.0-4.05	1.2	More enhancements
4.06-4.7	1.3	
	1.4	Server only
6-7, Firefox 1.0-4	1.5 (1999)	ECMAScript-262 compliant
Firefox 1.5	1.6	ECMAScript-262 Edition 3 compliant
Firefox 2.0	1.7	Generators, iterators and let statement
Firefox 3.0-3.5	1.8-1.8.1	Some ECMAScript 4 updates, JSON codec
Firefox 4.0-6.0	1.8.5 (5)	ECMAScript 5 partial support
Firefox 17-90+	6 - 9	ECMAScript 5.1 - ECMAScript 2015 (ES6) - ECMAScript 2018

MSIE JavaScript Versions

MSIE Browser	JScript	Comments
3.X/1	1.0	More or less compatible with NS JavaScript 1.0
3.X/2	2.0	More or less compatible with NS JavaScript 1.1
4.0	3.0	More or less compatible with NS JavaScript 1.2
5.0	5.0	More or less compatible with NS JavaScript 1.5
5.1	5.1	Minor update
5.5	5.5	Minor update
6.0	5.6	ECMA-262 3 rd edition
7.0	5.7	ECMA-262 3 rd edition + ECMA-327
8.0	5.8	ECMA-262 3 rd edition + ECMA-327 + JSON (RFC 4627)
9.0-11.0	10.0	Features from ECMA-262 5 th edition (ES 5), .NET
Edge	1.0	ECMAScript 2015 (aka ES6)

ECMAScript

- JavaScript now controlled by the ECMA standard body
- **ECMA** stands for **European Computer Manufacturers Association**
- First language specification, ECMA-262, a.k.a. ECMAScript, approved in 1997, closely resembles Netscape JavaScript 1.1
- Current language specification is **ECMA-262, 12th Edition, June 2021, ECMAScript © 2021**
- ECMA-262 12th Ed. language specification found at:
 - <http://www.ecma-international.org/ecma-262/>
- Mozilla JavaScript Docs
 - <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Helpful Links to Play with

- As usual you can use CodePen, JSFiddle, JS Bin etc. to learn JavaScript

<https://codepen.io>

<https://jsfiddle.net/>

<https://jsbin.com/>

- “JavaScript vs Other Popular Languages of the 21st Century”

<https://dzone.com/articles/javascript-vs-other-popular-languages-of-21st-cent>

- “JavaScript 20 Years Old Today”

<https://www.i-programmer.info/news/167-javascript/8595-javascript-20-years-old-today.html>

- Free software at “wtfjs” ☺

<https://wtfjs.com/>

- JavaScript WTF

<https://javascriptwtf.com/>

JSON – JavaScript Object Notation

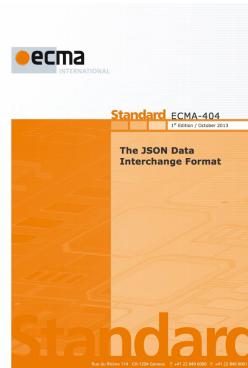
Some of the slides taken from
Douglas Crockford / Paypal Inc.

What is JSON

- **JSON**, short for **JavaScript Object Notation**, is a lightweight data interchange format.
 - It is a text-based, human-readable format for representing simple data structures and associative arrays (called objects).
- The JSON format is specified in RFC 4627 (July **2006**) authored by Douglas Crockford.
 - <https://tools.ietf.org/rfc/rfc4627.txt>
 - The official MIME type for JSON is application/json. The JSON file extension is .json.
- The JSON format is often used for transmitting structured data over a network connection in a process called serialization.
 - Its main application is in Ajax web application programming, where it serves as an alternative to the use of the XML format.
- Code for parsing and generating JSON data is readily available for a large variety of programming languages. The www.json.org website provides a comprehensive listing of existing JSON bindings, organized by language.

Brief History

- JSON was based on a subset of the JavaScript programming language (specifically, Standard ECMA-262 - now in its 12th Edition)
 - however, it is a language-independent data format.
 - For the complete specification see
<https://www.ecma-international.org/publications/standards/Ecma-262.htm>
- The JavaScript ECMA standard is based upon Netscape's JavaScript and Microsoft's Jscript



- Douglas Crockford was the original developer of JSON while he was at State Software, Inc. He is now Senior JavaScript Architect at Paypal
- <http://www.json.org/>, is a website devoted to JSON discussions and includes many JSON parsers

How to use the JSON format

- A JSON file allows one to load data from the server or to send data to it.
- Working with JSON involves three steps: (i) the browser processing, (ii) the server processing, and (iii) the data exchange between them.

1. Client side (browser)

- The content of a JSON file (or stream), or the definition of JSON data is assigned to a variable, and this variable becomes an object of the program.

2. Server side

- a JSON file (or stream) on the server can be operated upon by various programming languages, including PHP and Java thanks to parsers that process the file and may even convert it into classes and attributes of the language.

3. Data exchange

- Loading a JSON file from the server may be accomplished in JavaScript in several ways:
 - directly including the file into the HTML page, as a JavaScript **.json external file**.
 - loading by a JavaScript command
 - using **XMLHttpRequest()**
- To convert JSON into an object, it can be passed to the JavaScript **eval()** function.
- Sending the file to the server may be accomplished by XMLHttpRequest(). The file is sent as a text file and processed by the parser of the programming language that uses it.

JSON and XMLHttpRequest Example

- XMLHttpRequest will be covered in more detail in the AJAX lecture

- The XMLHttpRequest code:

```
var req = new XMLHttpRequest();
req.open("GET", "file.json", true); // "asynchronous" operation
req.onreadystatechange = myCode; // the callback
req.send(null);
```

- The JavaScript callback: eval() parses JSON, creates an object and assigns it to variable doc

```
function myCode() {
    if (req.readyState == 4) {
        if (req.Status == 200) {
            var doc = eval('(' + req.responseText + ')');
        }
    }
}
```

- Using the data:

```
var menuName = doc.getElementById('menu'); // finding a field menu
doc.menu.value = "my name is"; // assigning a value to the field
```

- How to access data:

```
doc.commands[0].title // read value of the "title" field in the array
doc.commands[0].action // read value of the "action" field in the array
```

JavaScript eval()

- The JavaScript **eval()** is a function property of the Global Object, and evaluates a string and executes it as if it was JavaScript code, e.g.

```
<script type="text/javascript">  
eval("x=10;y=20;document.write(x*y)");  
document.write("<br />");  
document.write(eval("2+2"));  
document.write("<br />");  
var x=10;  
document.write(eval(x+17));  
document.write("<br />"); </script>
```

- produces the output

200
4
27

- Because JSON-formatted text is also syntactically legal JavaScript code, an easy way for a JavaScript program to parse JSON-formatted data is to use the built-in JavaScript eval() function
- the JavaScript interpreter itself is used to *execute* the JSON data to produce native JavaScript objects.
- The eval() technique is subject to security vulnerabilities if the data and the entire JavaScript environment is not within the control of a single trusted source; See:
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval

JSON Basic Data Types

- ***String*** (double-quoted unicode with backslash escaping)
- ***Numbers*** (integer, real, or floating point)
- ***Booleans*** (true and false)
- ***Object*** (collection of key:value pairs, comma-separated and enclosed in curly brackets)
- ***Array*** (an ordered sequence of values, comma-separated and enclosed in square brackets)
- ***Null*** (a value that isn't anything)

String

- Sequence of 0 or more Unicode characters
- No separate character type
 - A character is represented as a string with a length of 1
- Wrapped in "double quotes"
- Backslash escapement

Object

- Objects are unordered containers of key/value pairs
- Objects are wrapped in { }
- , separates key/value pairs
- : separates keys and values
- Keys are strings (unlike in JavaScript)
- Values are JSON values
- Can be used to represent struct, record, hashtable, object

Example Object

```
{"name": "Jack B. Nimble", "at large":  
true, "grade": "A", "level": 3,  
"format": {"type": "rect", "width": 1920,  
"height": 1080, "interlace": false,  
"framerate": 24} }
```

Example Object Formatted

```
{  
  "name": "Jack B. Nimble",  
  "at large": true,  
  "grade": "A",  
  "level": 3,  
  "format": {  
    "type": "rect",  
    "width": 1920,  
    "height": 1080,  
    "interlace": false,  
    "framerate": 24  
  }  
}
```

Array

- Arrays are ordered sequences of values
- Arrays are wrapped in [] (square brackets)
- , separates values
- JSON does not talk about indexing.
 - An implementation can start array indexing at 0 or 1.

Two Examples of JSON Arrays

- **One dimensional**

```
[ "Sunday", "Monday", "Tuesday", "Wednesday",
  "Thursday", "Friday", "Saturday"]
```

- **Two dimensional**

```
[  
  [0, -1, 0],  
  [1, 0, 0],  
  [0, 0, 1]  
]
```

Arrays vs Objects

- Use objects when the key names are arbitrary strings
- Use arrays when the key names are sequential integers

JSON is Not XML

JSON

- Objects
- Arrays
- Strings
- Numbers
- Booleans
- null

XML

- element
- attribute
- Attribute string
- content
- <![CDATA[]]>
- Entities
- Declarations
- Schema
- Stylesheets
- Comments
- Version
- namespace

JSON vs. XML Example

JSON

```
{  
  "menu": "File",  
  "commands": [  
    {  
      "title": "New",  
      "action": "CreateDoc"  
    },  
    {  
      "title": "Open",  
      "action": "OpenDoc"  
    },  
    {  
      "title": "Close",  
      "action": "CloseDoc"  
    }  
  ]  
}
```

XML Equivalent

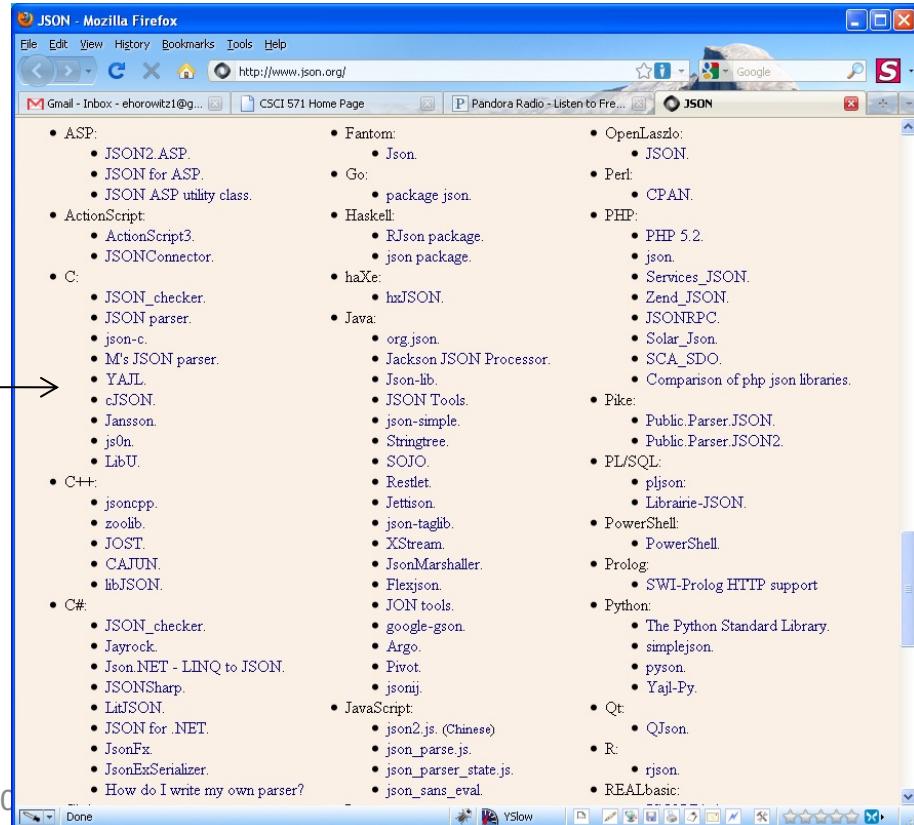
```
<?xml version="1.0" ?>  
<root>  
  <menu>File</menu>  
  <commands>  
    <item>  
      <title>New</value>  
      <action>CreateDoc</action>  
    </item>  
    <item>  
      <title>Open</value>  
      <action>OpenDoc</action>  
    </item>  
    <item>  
      <title>Close</value>  
      <action>CloseDoc</action>  
    </item>  
  </commands>  
</root>
```

Rules for JSON Parsers

- A JSON decoder must accept all well-formed JSON text
- A JSON decoder may also accept non-JSON text
- A JSON encoder must only produce well-formed JSON text
- A list of decoders for JSON can be found at

<http://www.json.org/>

JSON parsers for programming languages include C, C++, C#, Java, JavaScript, Perl, PHP



Same Origin Policy

- Same origin policy is a security feature that browsers apply to client-side scripts
- It prevents a document or script loaded from one “origin” from getting or setting properties of a document from a different “origin”
 - Rationale: the browser should not trust content loaded from arbitrary websites
- Given the URL: <http://www.example.com/dir/page.html>

URL	Outcome	Reason
http://www.example.com/dir2/other.json	Success	Same protocol and host
http://www.example.com/dir/inner/other.json	Success	Same protocol and host
http://www.example.com:81/dir2/other.json	Failure	Same protocol and host but different port
https://www.example.com/dir2/other.json	Failure	Different protocol
http://en.example.com/dir2/other.json	Failure	Different host
http://example.com/dir2/other.json	Failure	Different host

JSON: The Cross-Domain Hack

- JSON and the **<script> tag** provide a way to get around the Same Origin Policy

```
<script src=http://otherdomain.com/data.js>  
</script>
```

- The **src** attribute of a script tag can be set to a URL from any server, and every browser will go and retrieve it, and read it into your page
- So, a script tag can be set to point at a URL on another server with JSON data in it, and that JSON will become a global variable in the webpage
- So JSON can be used to grab data from other servers, without the use of a server-side proxy
- Available in HTML since 1994

JSON and Dynamic Script Tag “Hack”

- Using JSON it is possible to get around the limitation that data can only come from a single domain (bypasses cross-domain)
- To do this one needs to
 - to find a website that returns JSON data, and
 - A JavaScript program that contains a JSONScriptRequest class that creates a **dynamic <script> tag** and its contents
- The implementation of this class can be found at the class website:

http://csci571.com/examples/js/jsr_class.js

- The most important line in the script (the "hack") is the following one:

```
this.scriptObj.setAttribute("src", this.fullUrl + this.noCacheIE);
```

which sets the src attribute of the <script> tag to a new URL

Source Code for jsr_class.js

```
// Constructor -- pass a REST request URL to the constructor
function JSONscriptRequest(fullUrl) {
    // REST request path
    this.fullUrl = fullUrl;
    // Keep IE from caching requests
    this.noCacheIE = '&noCacheIE=' + (new Date()).getTime();
    // Get the DOM location to put the script tag
    this.headLoc = document.getElementsByTagName("head").item(0);
    // Generate a unique script tag id
    this.scriptId = 'JscriptId' + JSONscriptRequest.scriptCounter++;
}

// Static script ID counter
JSONscriptRequest.scriptCounter = 1;

// buildScriptTag method
JSONscriptRequest.prototype.buildScriptTag = function () {
    // Create the script tag
    this.scriptObj = document.createElement("script");
    // Add script object attributes
    this.scriptObj.setAttribute("type", "text/javascript");
    this.scriptObj.setAttribute("charset", "utf-8");
    this.scriptObj.setAttribute("src", this.fullUrl + this.noCacheIE);
    this.scriptObj.setAttribute("id", this.scriptId);
}

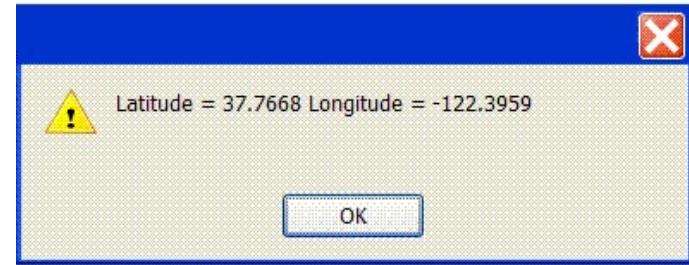
// removeScriptTag method
JSONscriptRequest.prototype.removeScriptTag = function () {
    // Destroy the script tag
    this.headLoc.removeChild(this.scriptObj);
}

// addScriptTag method
JSONscriptRequest.prototype.addScriptTag = function () {
    // Create the script tag
    this.headLoc.appendChild(this.scriptObj);
}
```

Critical line

Example Using JSON and JSONscriptRequest class

```
<html><body>  
// Include the JSONscriptRequest class  
<script type="text/javascript" src="jsr_class.js"> </script>  
<script type="text/javascript">  
// Define the callback function  
function getGeo(jsonData) {  
    alert('Latitude = ' + jsonData.ResultSet.Result[0].Latitude + ' Longitude = ' +  
        jsonData.ResultSet.Result[0].Longitude);  
    bObj.removeScriptTag();  
}  
// The web service call  
var req = 'http://api.local.yahoo.com/MapsService/V1/geocode?appid=YahooDemo&output=json&callback=getGeo&location=94107';  
// Create a new request object  
bObj = new JSONscriptRequest(req);  
// Build the dynamic script tag  
bObj.buildScriptTag();  
// Add the script tag to the page  
bObj.addScriptTag();  
</script></body></html>
```



output

buildScriptTag creates

<script src="getGeo({"ResultSet": {"Result": [{"precision": "zip", ...}]} })>

Adding the <script> tag to the page causes getGeo to be called
And the JSON-encoded data to be passed to the getGeo function;
The JavaScript interpreter automatically turns JSON into a
JavaScript object, and the returned data can be referenced
Immediately. See:

<https://www.xml.com/pub/a/2005/12/21/json-dynamic-script-tag.html>

Tiingo.com Example

- Tiingo provides Stock Lookup and Stock Quote APIs. Results are in JSON format.
- To access the service, you do not need an Application ID.
- Stock End-of-Day Quote JSON REST call looks like this:

```
https://api.tiingo.com/tiingo/daily/aapl/prices?startDate=2019-01-02&token=<token>
```

- Response:

```
[ { "date": "2019-01-02T00:00:00.000Z", "close": 157.92, "high": 158.85, "low": 154.23, "open": 154.89, "volume": 37039737, "adjClose": 157.92, "adjHigh": 158.85, "adjLow": 154.23, "adjOpen": 154.89, "adjVolume": 37039737, "divCash": 0.0, "splitFactor": 1.0 }, ... ]
```

- Crypto Top-of-book JSON REST call looks like this:

```
https://api.tiingo.com/tiingo/crypto/top?tickers=curebtc&token=<token>
```

- Response:

```
[ { "ticker": "curebtc", "baseCurrency": "cure", "quoteCurrency": "btc", "topOfBookData": [ { "askSize": 21.55601545, "bidSize": 726.29848588, "lastSaleTimestamp": "2019-01-30T00:19:34.777000+00:00", "lastPrice": 1.894e-05, "askPrice": 1.9e-05, "quoteTimestamp": "2019-01-30T00:44:34.209957+00:00", "bidExchange": "BITTREX", "lastSizeNotional": 0.0010885247347072, "lastExchange": "BITTREX", "askExchange": "BITTREX", "bidPrice": 1.894e-05, "lastSize": 57.47226688 } ] }
```

XMLHttpRequest Compared to the Dynamic Script Tag

| | XmlHttpRequest | Dynamic script Tag | |
|--|----------------|---|---|
| Cross-browser compatible? | No | Yes | Dynamic script tag is used by internet advertisers who use it to pull their ads into a web page |
| Cross-domain browser security enforced? | Yes (*) | No | |
| Can receive HTTP status codes? | Yes | No (fails on any HTTP status other than 200) | The script tag's main advantages are that it is not bound by the web browser's cross-domain security restrictions and that it runs identically on more web browsers than XMLHttpRequest. |
| Supports HTTP GET and POST? | Yes | No (GET only) | |
| Can send/receive HTTP headers? | Yes | No | |
| Can receive XML? | Yes | Yes (but only embedded in a JavaScript statement) | If your web service happens to offer JSON output and a callback function, you can easily access web services from within your JavaScript applications without having to parse the returned data |
| Can receive JSON? | Yes | Yes (but only embedded in a JavaScript statement) | |
| Offers synchronous and asynchronous calls? | Yes | No (asynchronous only) | |

* CORS-compatible browsers allow cross-domain XMLHttpRequest

Arguments against JSON

- JSON doesn't have namespaces
- JSON has no validator
 - Every application is responsible for validating its inputs
- JSON is not extensible
 - But it does not need to be
- JSON is not XML
 - But a JavaScript compiler is a JSON decoder

Features that make JSON well-suited for data transfer

- It is both a human and machine-readable format;
- It has support for unicode, allowing almost any information in any human language to be communicated
- The format is self-documenting in that it describes structure and field names as well as specific values
- The strict syntax and parsing requirements allow the parsing algorithms to remain simple, efficient, and consistent
- JSON has the ability to represent the most general of computer science data structures: records, lists and trees

`eval()` and `JSON.parse()` Security

- The `eval()` function is very fast. However, it can compile and execute any JavaScript program, so there can be security issues
- In general
 - your browser should not trust machines not under your absolute control
 - Your server must validate everything the client tells it
- To help guard the browser from insecure JSON input, use **`JSON.parse()`** instead of `eval` ; e.g., `JSON.parse()` is used this way

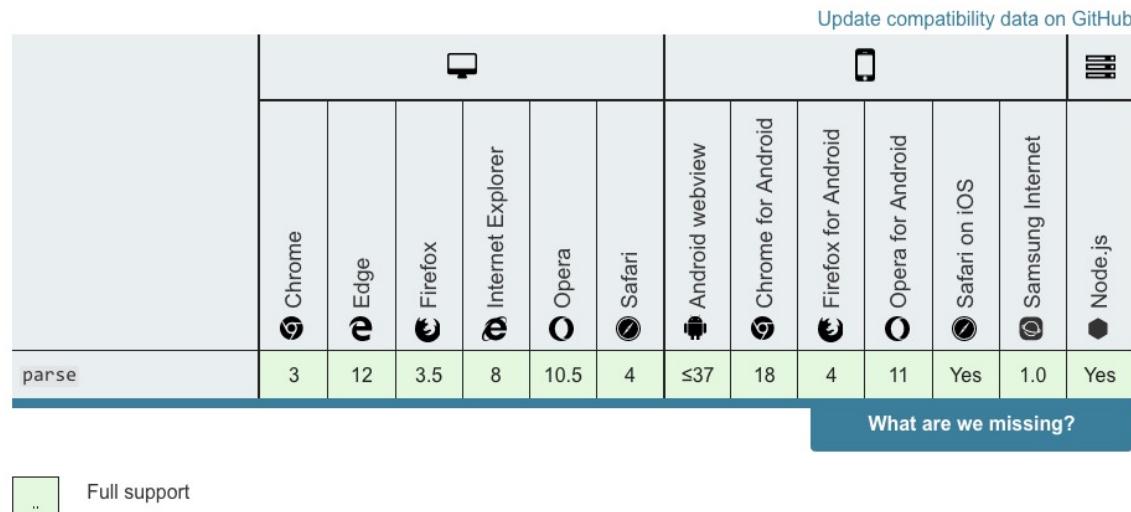
```
var myObject = JSON.parse(JSONtext [, reviver]);
```
- The optional `reviver` parameter is a function that will be called for every key and value at every level of the result. Each value will be replaced by the result of the `reviver` function. This can be used to reform generic objects into instances of pseudoclasses, or to transform date strings into Date objects.

More on JSON.parse()

- JSON.parse() is included in ECMAScript since 5th Ed. and all recent desktop browsers (Chrome, Firefox 3.5+, IE 8+, Opera 10.5+, Safari 4+) and all mobile browsers. JSON.parse() compiles faster than eval(). See:

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON\(parse](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON(parse)

Browser compatibility



More on JSON.parse (cont'd)

- In JavaScript there is a function called JSON.parse. It uses a single call to eval to do the conversion, guarded by a single regexp test to assure that the input is safe.
- The input object is traversed recursively, and various functions are called for each member of the object in post-order (i.e. every object is reviewed after all its members have been reviewed).
- For each member, the following occurs:
 - If reviewer returns a valid value, the member value is replaced with the value returned by reviewer.
 - If reviewer returns what it received, the structure is not modified.
 - If reviewer returns null or undefined, the object member is deleted.
- The reviewer argument is often used to transform JSON representation of ISO date strings into UTC format Date objects.
- Here is the original source code for JSON.parse

```
JSON.parse = function (text) {
  return
    (/^(\s|[,:{}\[\]]|"(\\"\\bfnrtu]|[^\\x00-\\x1f"\\"])*|-?
    ?\d+\\.?\d*)?([eE][+-]?\d+)?|true|false|null)+$/.test(text))
    && eval('(' + text + ')');
}
```
- According to Doug Crockford, “It is ugly, but it is really efficient.”
- For the actual source implementation and explanation of the behavior of the “reference implementation” of JSON.parse see
 - <https://github.com/douglascrockford/JSON-js>
 - (for applications that need to run on obsolete browsers)
- JQuery provides implicit ParseJSON() method and responseJSON property. See
 - <http://api.jquery.com/jquery.parsejson/> and <http://api.jquery.com/jQuery.ajax/>

Douglas Crockford Discusses AJAX

- Go to
- <https://www.youtube.com/watch?v=-C-JoyNuQJs>
- Start video at -27 minutes and run it as long as it is interesting
- He gets to JSON at -13 minutes

JSONP

- JSONP or "*JSON with padding*" is a JSON extension wherein the name of a callback function is specified as an input argument of the call itself.
- It is now used by many Web 2.0 applications such as Dojo Toolkit Applications or Google Toolkit Applications.
- Further extensions of this protocol have been proposed
- Because JSONP makes use of script tags, calls are essentially open to the world. For that reason, JSONP may be inappropriate to carry sensitive data
- JSONP is supported by jQuery. See:

<https://learn.jquery.com/ajax/working-with-jsonp/>

JSONP Example

- Consider the following <script> tag which includes a src attribute referring to a Google spreadsheet

```
<script  
    src="http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic?alt=json-in-script&callback=listTasks"> </script>
```

- the URL in the above script will return the following JSON result:

```
listTasks ({ "version": "1.0", "encoding": "UTF-8", "feed": {  
    "entry": [ {  
        "title": {  
            "type": "text", "$t": "Make google gadget w/spreadsheet example" },  
        "content": { "type": "text", "$t": "Status: Done" } },  
        { "title": {  
            "type": "text", "$t": "Do final project for class" },  
        "content": { "type": "text", "$t": "Status: NotStarted" }  
    ] } } );
```

- notice the src attribute's URL has alt=json-in-script and callback=listTasks. This tells google that it wants not just JSON (data) but JSONP (data passed as a parameter of the listed function listTasks).
- The about output is JSON data, surrounded by the Procedure call "listTasks", with the data as a "parameter".
- Try this in a browser:
<http://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic?alt=json-in-script&callback=listTasks>

What is Actually Returned Today (since 2010)

```
listTasks ({"version":"1.0","encoding":"UTF-8", "feed": {"xmlns": "http://www.w3.org/2005/Atom",  
"xmlns$openSearch": "http://a9.com/-/spec/opensearchrss/1.0/",  
"xmlns$gsx": "http://schemas.google.com/spreadsheets/2006/extended",  
"id": {"$t": "https://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic"},  
"updated": {"$t": "2006-12-05T10:35:42.800Z"}, "category": [{"scheme": "http://schemas.google.com/spreadsheets/2006",  
"term": "http://schemas.google.com/spreadsheets/2006#list"}], "title": {"type": "text", "$t": "Sheet1"},  
"link": [{"rel": "alternate", "type": "text/html",  
"href": "https://spreadsheets.google.com/pub?key\u003do03712292828507838454.2635427448373779250"},  
{"rel": "http://schemas.google.com/g/2005#feed", "type": "application/atom+xml",  
"href": "https://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic"},  
{"rel": "self", "type": "application/atom+xml",  
"href": "https://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic?alt\u003djson  
-in-script"}],  
"author": [{"name": {"$t": "pamela.fox"}, "email": {"$t": "pamela.fox@gmail.com"}}, "openSearch$totalResults": {"$t": "2"},  
"openSearch$startIndex": {"$t": "1"},  
"entry": [{"id": {"$t": "https://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/c  
okwr"},  
"updated": {"$t": "2006-12-05T10:35:42.800Z"}, "category": [{"scheme": "http://schemas.google.com/spreadsheets/2006",  
"term": "http://schemas.google.com/spreadsheets/2006#list"}], "title": {"type": "text",  
"$t": "Make google gadget w/spreadsheet example"}, "content": {"type": "text", "$t": "status: Done"},  
"link": [{"rel": "self", "type": "application/atom+xml",  
"href": "https://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/cokwr"}]},  
{"id": {"$t": "https://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/cpzh4"},  
"updated": {"$t": "2006-12-05T10:35:42.800Z"}, "category": [{"scheme": "http://schemas.google.com/spreadsheets/2006",  
"term": "http://schemas.google.com/spreadsheets/2006#list"}], "title": {"type": "text", "$t": "Do final project for class"},  
"content": {"type": "text", "$t": "status: NotStarted"}, "link": [{"rel": "self", "type": "application/atom+xml",  
"href": "https://spreadsheets.google.com/feeds/list/o03712292828507838454.2635427448373779250/od6/public/basic/cpzh4"}]}]);
```

JSONP Example, Cont'd

- When dynamically executed (e.g. using JavaScript's "eval" on it), it invokes the function `listTasks` :

```
function listTasks(root) {  
    var feed = root.feed;  
    var entries = feed.entry || [];  
    var html = [""];  
    html.push('<ul>');  
    for (var i = 0; i < feed.entry.length; ++i) {  
        var entry = feed.entry[i];  
        var title = entry.title.$t;  
        var content = entry.content.$t;  
        html.push('<li>', title, ' (', content, ') </li>');  
    }  
    html.push('</ul>');  
    document.getElementById("agenda").innerHTML =  
    html.join("");  
}
```

For more discussion of JSONP see
<http://www.west-wind.com/Weblog/posts/107136.aspx>

- The "output" of such a function in the example would be like this:

```
<ul>  
  <li> Make google gadget w/spreadsheet example (Status: Done) </li>  
  <li> Do final project for class (Status: NotStarted) </li>  
<ul>
```

- and it is stored in the "html" variable.
- You then assign such HTML to the "innerHTML" of the element with Id "agenda", which then shows it on the web page.

Python Includes JSON functionality

- Python includes an encoder and a decoder. JSON functions include
 - `json.dump` – serialize object as JSON formatted string
 - `class json.JSONDecoder` - simple JSON decoder
 - `class json.JSONEncoder` – extensible JSON decoder for Python data structures
- See: <https://docs.python.org/3/library/json.html>

JSON	Python
object	<code>dict</code>
array	<code>list</code>
string	<code>str</code>
number (int)	<code>int</code>
number (real)	<code>float</code>
true	<code>True</code>
false	<code>False</code>
null	<code>None</code>

Compact Encoding JSON in Python

```
>>> import json  
>>> json.dumps([1, 2, 3, {'4': 5, '6': 7}], separators=(',', ':'))  
'[1,2,3,{"4":5,"6":7}]'
```

Decoding JSON in Python

```
>>> import json  
>>> json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')  
['foo', {'bar': ['baz', None, 1.0, 2]}]  
>>> json.loads('"\\\"foo\\\"bar"')  
"foo\x08ar"  
>>> from io import StringIO  
>>> io = StringIO('["streaming API"]')  
>>> json.load(io) ['streaming API']  
['streaming API']
```

JQuery JSON Support

- JQuery Ajax APIs provide extensive support for JSON and JSONP
- `jQuery.ajax([settings])` includes the following JSON-related settings:
 - **crossDomain** (default: **false** for same-domain requests, **true** for cross-domain requests) - If you wish to force a crossDomain request (such as JSONP) on the same domain, set the value of crossDomain to true. This allows, for example, server-side redirection to another domain.
 - **dataType** (xml, json, script, or html) - The type of data that you're expecting back from the server.
 - "json": Evaluates the response as JSON and returns a JavaScript object. The JSON data is parsed in a strict manner.
 - If json dataType is specified, the response is parsed using `jQuery.parseJSON` before being passed, as an object, to the success handler. The parsed JSON object is made available through the `responseJSON` property.
 - "jsonp": Loads in a JSON block using JSONP. Adds an extra "?callback=?" to the end of your URL to specify the callback.
 - If jsonp is specified, `$.ajax()` will automatically append a query string parameter of (by default) `callback=?` to the URL.
 - **jsonp** - Override the callback function name in a JSONP request. This value will be used instead of 'callback' in the 'callback=?' part of the query string in the url.
 - See <http://api.jquery.com/jquery.ajax/>

Example 14: A Longer DOM Example in JSON

A screenshot of a web browser window titled "View Aircraft". The URL is "cs-server.usc.edu:45678/examples/dom/aircraft/aircraft_json.html". The page contains the following text:
View the Commercial Aircrafts
Enter boeing.json or airbus.json and click on Submit
airbus.json
Submit Query

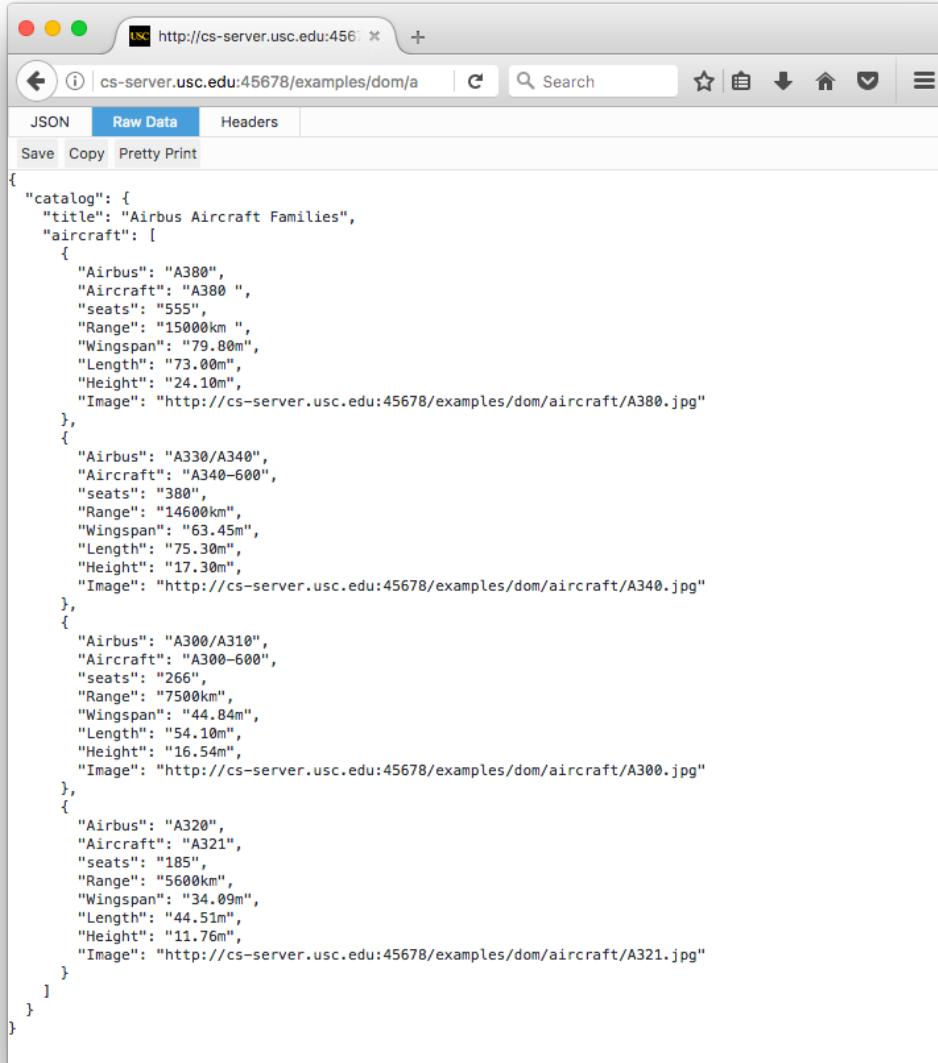
A screenshot of a web browser window titled "XML Parse Result" with the URL "about:blank". The title is "Airbus Aircraft Families". Below the title is a table with the following data:

Family	Aircraft	Seats	Range	Wing Span	Length	Height	Image
A380	A380	555	15000km	79.80m	73.00m	24.10m	
A330/A340	A340-600	380	14600km	63.45m	75.30m	17.30m	
A300/A310	A300-600	266	7500km	44.84m	54.10m	16.54m	
A320	A321	185	5600km	34.09m	44.51m	11.76m	

Given a URL of a JSON file that describes a set of aircraft, re-format the data into an HTML page.

See: https://csci571.com/examples/dom/aircraft/aircraft_json.html

airbus.json



A screenshot of a web browser window displaying the JSON file `airbus.json`. The browser has a tab bar with three tabs: "JSON", "Raw Data", and "Headers". The "JSON" tab is selected. Below the tabs are buttons for "Save", "Copy", and "Pretty Print". The main content area shows the JSON data structure.

```
{
  "catalog": {
    "title": "Airbus Aircraft Families",
    "aircraft": [
      {
        "Airbus": "A380",
        "Aircraft": "A380",
        "seats": "555",
        "Range": "15000km",
        "Wingspan": "79.80m",
        "Length": "73.00m",
        "Height": "24.10m",
        "Image": "http://cs-server.usc.edu:45678/examples/dom/aircraft/A380.jpg"
      },
      {
        "Airbus": "A330/A340",
        "Aircraft": "A340-600",
        "seats": "380",
        "Range": "14600km",
        "Wingspan": "63.45m",
        "Length": "75.30m",
        "Height": "17.30m",
        "Image": "http://cs-server.usc.edu:45678/examples/dom/aircraft/A340.jpg"
      },
      {
        "Airbus": "A300/A310",
        "Aircraft": "A300-600",
        "seats": "266",
        "Range": "7500km",
        "Wingspan": "44.84m",
        "Length": "54.10m",
        "Height": "16.54m",
        "Image": "http://cs-server.usc.edu:45678/examples/dom/aircraft/A300.jpg"
      },
      {
        "Airbus": "A320",
        "Aircraft": "A321",
        "seats": "185",
        "Range": "5600km",
        "Wingspan": "34.09m",
        "Length": "44.51m",
        "Height": "11.76m",
        "Image": "http://cs-server.usc.edu:45678/examples/dom/aircraft/A321.jpg"
      }
    ]
  }
}
```

HTML Code for the Initial Input

```
<h1>View the Commercial Aircrafts </h1>
```

Enter JSON file

```
<form name="myform" method="POST" id="location">
<input type="text" name="URL" maxlength="255" size="100"
       value="airbus.json" />
<br />
<input type="button" name="submit" value="Submit Query"
       onClick="viewJSON(this.form)" />
</form>
```

viewJSON Routine

```
function viewJSON(what) {  
var URL = what.URL.value;  
  
function loadJSON(url) {  
xmlhttp=new XMLHttpRequest();  
xmlhttp.open("GET",url,false); // "synchronous" (deprecated because it freezes the page while waiting for a response) *  
xmlhttp.send();  
jsonObj= JSON.parse(xmlhttp.responseText);  
return jsonObj;  
}  
  
jsonObj = loadJSON(URL);  
jsonObj.onload=generateHTML(jsonObj);  
hWin = window.open("", "Assignment4", "height=800,width=600");  
hWin.document.write(html_text);  
hWin.document.close();  
}
```

- See:

https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Synchronous_and_Asynchronous_Requests

generateHTML Routine

```
function generateHTML(jsonObj)  {
    root=jsonObj.DocumentElement;
    html_text+"<html><head><title>JSON Parse Result</title></head><body>";
    html_text+="<table border='2'>";
    caption=jsonObj.catalog.title;
    html_text+="


```

generateHTML Routine (cont' d)

```
html_text+="</tr>";
// output out the values
for(i=0;i<planes.length;i++) //do for all planes (one per row)
{
    planeNodeList=planes[i]; //get properties of a plane (an object)
    html_text+="<tr>";    //start a new row of the output table
    var aircraft_keys = Object.keys(planeNodeList);
    for(j=0;j<aircraft_keys.length;j++)
    {
        prop = aircraft_keys[j];
        if(aircraft_keys[j]=="Image")
            {//handle images separately
                html_text+=" <img src='"+ planeNodeList[prop] +"' width='"+x+"' height='"+y+"'></td>";             } else {                 html_text+=" "+ planeNodeList[prop] +"</td>";             }     }     html_text+="</tr>"; } html_text+="</tbody>"; html_text+="</table>"; html_text+=" </bo> + <dy> </html>"; } | |
```

Python Tutorial

What is Python?

- Interpreted, Object-oriented, High-level programming language
- Has high-level built in data structures
- Has dynamic typing and dynamic binding
- Supports both procedural and object-oriented paradigm
- Downloads, documentation, community support, news and event at:

<https://www.python.org/>

What is Python (cont'd)

- Useful for scripting
- Does not have a compilation step
- Various built-in functions/modules allow for fast development
- Compatible with many popular databases like PostgreSQL and MySQL

What does Python code look like?

- Simpler than equivalent C, C++, or Java code
- Shorter than C, C++, Java
- Offers greater error checking than C
- Simple, easy to learn syntax
- Allows splitting program into modules which can be reused

Getting Started with Python Interpreter

- The interpreter usually gets installed at `/usr/local/bin/pythonX.Y:`
`/usr/local/bin/python3.7`
- The interpreter can be started using `python3.7` or later versions (3.8 and 3.9 recommended) or simply `python` after putting the interpreter path in Unix shell's search path or the environment variables in Windows.
- Another way to start the interpreter is `python -c command [arg] ...` This executes the statements in 'command' script
- The script name and arguments are turned into a list of strings and assigned to the `argv` variable in the `sys` module

Hello World Example

- sample.py:

```
'''sample.py file to print Hello World'''  
print("Hello World")
```

- Run sample.py as *python sample.py*

```
[(base) Kevins-MBP:Desktop daftary$ python sample.py  
Hello World  
(base) Kevins-MBP:Desktop daftary$ █
```

Comments in Python

- There are two types of comments in Python

```
# Single-line Shell-style comments  
  
''' These are  
Multi-line comments. '''
```

Python Variables

- Python variables can be declared by any name or even alphabets like a, aa, abc, etc.
- **Variables** are case-sensitive (abc != aBc)
- **Global** variables can be used anywhere (declared outside a function)
- **Local** variables restricted to a function or class
- **No** keyword called **static** is present
- Variables assigned values inside a class declaration are **class variables**
- Variables assigned values in class methods are **instance variables**

Python Variables (cont'd)

- Variables are containers for storing data values
- Python has no command for declaring a variable
- A variable is created the moment you first assign a value to it
- Variables are **not statically typed**
- Integers can become floats, then can become strings
- Variables take the type of the current value
- If you want to specify the data type of a variable, this can be done with “casting”
- Variable types include :
 - Boolean, Integer
 - Float, String
 - List, Object
 - NULL, Tuple
 - Dictionary, Set

Python Variables (cont'd)

- Assignment by value

```
a = 10  
b = "foo"  
c = [1, 2, 3, 4] # List  
d = (1,2) # Tuple  
e = {'key': 'value'} # Dictionary
```

Displaying Variables

- To display a variable, use the **print** statement; pass the variable name to the `print` statement, enclosing it in brackets (for python 3.x) or without brackets(for python 2.x):

```
age = 18;  
print(age) #python 3.x  
print age #python 2.x
```

- To display both text strings and variables, pass them to the **print** statement as individual arguments, separated by commas:

```
print("The legal voting age is ", age)
```

Naming Variables

- The following rules and conventions must be followed when naming a variable:
 - Variable names must **begin** with a **letter** or **underscore** (`_`) character
 - Variable names may contain **alphanumeric** characters (uppercase and lowercase letters), **numbers**, or **underscores** (`_`).
 - Variable names cannot contain spaces
 - Variable names are **case sensitive**

Python Constants

- Constants are special variables that hold values that **should not be changed**
- Start with letter or underscore (_) followed by letters, numbers or underscores
- Use them for named items that will not change
- Constant names use all uppercase letters
- Constants have global scope
- The constants module of python can be used for some common constants like PI, GRAVITY etc.
- **Constants are not really part of Python specification but part of community usage**

Python Operators

- Standard **Arithmetic** operators

+ , - , * , / (always returns a float value) , % (modulus) , ** (exponentiation) and // (floor division)

- String **concatenation** with a ‘+’ character

```
car = "SEAT" + " Altea"
```

print(car) would output "SEAT Altea"

- Basic Boolean comparison with “==”
- Using only = will overwrite a variable value (assignment)
- Less than < and greater than >
- <= and >= as above but include equality
- != can be used to check if two variables are not equal

Python Operators (cont'd)

- **Assignment (=) and combined assignment**

```
a = 3;  
a += 5; // sets a to 8;  
b = "Hello ";  
b += "There!"; // sets b to "Hello There!";
```

- **Bitwise operators (&, |, ^, ~, <<, >>)**

`a ^ b` (Xor: Bits that are set in a or b but not both are set.)
`~a` (Not: Bits that are set in a are not set, and vice versa.)

All arithmetic and bitwise operators can be combined with the assignment operator

Note: Python **DOES NOT** support ‘++’ and ‘--’ notation for auto increments and decrement

Python Operators (cont'd)

- **Logical Operators**
 - **and**: returns true if both statements are true (replacement for &&)If $x=3$, then $x<5$ and $x<10$ returns True
 - **or**: Returns **True** if one of the statements is true (replacement for ||)If $x = 4$, then $x<4$ or $x<5$ returns True
 - **not**: Reverse the result, returns False if the result is true
- **Identity Operators**
 - **is**: Returns true if both variables are the same object
 - **is not**: Returns true if both variables are not the same object
- **Membership Operators**
 - **in**: Returns True if a sequence with the specified value is present in the object
 - **not in**: Returns True if a sequence with the specified value is not present in the object

Data Types

- Python is a **dynamically typed** language (similar to JavaScript)
- Python supports the following types:
 - **Boolean**: True or False
 - **Numeric types**
 - **Integer**: Positive or negative whole numbers, complex numbers (eg., $3 + 5j$)
 - **Float**: Any real number
 - **Sequence types**
 - **String**: Sequence-type data type allowing for individual character access
 - **List**: Ordered collection of one or more data items, could be of different types, enclosed in square brackets (eg., [1, 'Hello', 3.41, True])
 - **Tuple**: Ordered collection of one or more data items, could be of different types, enclosed in parentheses (e.g., (1,2,"Hello"))
 - **Dictionary**: Unordered collection of data in *key:value* pairs form, enclosed in curly brackets (e.g., {1:"Professor", 2:"Marco",3:"Papa"})

Data Types Example

```
vat_rate = 0.175                                /* VAT Rate is numeric */
print(vat_rate * 100 + "%")          // throws TypeError
print(str(vat_rate * 100) + "%") // outputs "17.5%"
```

Numeric Data Types

- Python supports two numeric data types:
 - An **integer** is a positive or negative whole number with no decimal places (-250, 2, 100, 10,000) or complex numbers with ‘j’ denoting the imaginary part (2 + 4j)
 - A **floating-point number** is a number that contains decimal places or that is written in scientific notation (-6.16, 3.17, 2.7541)

Boolean Values

- A **Boolean value** is a value of **True** or **False** (true and false, in lower case ,are invalid)
- In Python programming, you can only use True or False Boolean values
- In other programming languages, you can use integers such as 1 = True, 0 = False

Strings in Python

- A **collection of one or more characters**, enclosed in single or double **quotes**
- Can use backslash as escape character
- Concatenate strings using '+'. Repeat using '*'
- Strings can be indexed

```
a = "Hello "
print(a[0]) # prints H
print(a[-1]) # prints o - reverse indexing
```

- Strings can be sliced
- print(a[1:3]) #prints 'el'
- Strings are **immutable**

Lists in Python

- An **ordered collection** of one or more data items, **not** necessarily of **same type**, enclosed by **square [] brackets**
- Lists have multiple methods like `append()`, `insert()`, `remove()`, `sort()`, `count()`, `reverse()`, etc. to manipulate the elements of the list
- The **del** statement allows deletion of elements and even complete lists (converts to empty list) as well as variables (reference error if you try to access the same variable)

```
>>> squares = []
>>> for x in range(10):
...     squares.append(x**2)
...
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Sets in Python

- An **unordered collection** of objects with **no duplicate elements, not necessarily of same type**, separated by commas, and enclosed by **curly {} brackets**
- Used for membership tests, eliminating duplicates, union, intersection, difference and symmetric difference

```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'} >>>
>>> print(basket) # show that duplicates have been removed
{'orange', 'banana', 'pear', 'apple'}
>>> 'orange' in basket # fast membership testing
True
>>> 'crabgrass' in basket
False

>>> # Demonstrate set operations on unique letters from two words
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a # unique letters in a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b # letters in a but not in b
{'r', 'd', 'b'}
>>> a | b # letters in a or b or both
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b # letters in both a and b
{'a', 'c'}
>>> a ^ b # letters in a or b but not both
{'r', 'd', 'b', 'm', 'z', 'l'}
```

Dictionaries in Python

- Dictionaries are like ‘associative arrays’, **unordered sequences of key-value pairs**
- Indexed using **keys** that are of **immutable** types
- General format:

```
dict = { key1:value1, key2:value2, ... keyN:valueN }
```

- `list(dictionaryName)` returns list of keys
- Can be created using either `{}` or `dict()`
- The key and value can be retrieved at the same time using `items()` method

Variable usage

```
foo = 25                      #Numerical variable
bar = "Hello"                   #String variable

foo = (foo * 7)                 #Multiples foo by 7
bar = (bar * 7)                 #VALID expression (bar becomes
                                #HelloHelloHelloHelloHelloHelloHello
bar = (bar + 7)                 #Invalid expression, throws error
```

```
[>>> bar = "Hello"
[>>> print(bar*7)
HelloHelloHelloHelloHelloHelloHello
[>>> print(bar)
Hello
[>>> bar = (bar*7)
[>>> print(bar)
HelloHelloHelloHelloHelloHelloHello
[>>> bar = (bar + 7)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

‘print’ example

```
foo = 25          // Numerical variable  
bar = "Hello"    // String variable  
print(bar)       // Outputs Hello  
print(foo,bar)   // Outputs 25 Hello  
print("5x5=",foo) // Outputs 5x5= 25  
print("5x5=foo") // Outputs 5x5=foo
```

Notice how `print "5x5=foo"` outputs ‘foo’ rather than replacing it with 25

Arithmetict Operations

```
a=15  
b=30  
c=2  
total=a+b  
a_squared = a**c          // 15**2  
print(total)               // total is 45  
print(a_squared)           // 225
```

```
a - b      // subtraction  
a * b      // multiplication  
a / b      // division  
a += 5     // a = a+5       - also works for *= and /=
```

Concatenation

Use a ‘+’ to join strings into one.

```
string1="Hello"  
string2="Python"  
string3= string1 + " " + string2  
print(string3)
```

Output: Hello Python

Escaping Characters

- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.

```
heading = " \"Computer Science\" "
print(heading)
```

Output: "Computer Science"

Python Control Structures

- Control Structures: the structures within a language that allow us to control the flow of execution through a program or script.
- Grouped into **conditional / branching** structures (e.g. if/else) and **repetition** structures (e.g. while loops).
- Example if/elif/else statement: [notice the “:]”]

```
if (foo == 0):  
    print("The variable foo is equal to 0")  
elif ((foo > 0) && (foo <= 5)):  
    print("The variable foo is between 1 and 5")  
else:  
    print("The variable foo is equal to", foo)
```

If ... Else...

```
If (condition):  
    Statements  
Else:  
    Statement
```

No 'Then' in Python !

Example:

```
if(user=="John"):  
    print("Hello John.")  
else:  
    print("You are not John.")
```

While Loops

General format:

```
While (condition) :  
    Statements;
```

Example:

```
count=0  
While(count<3) :  
    print("hello Python. ")  
    count += 1  
    // count = count + 1
```

Output: hello Python. hello Python. hello Python.

For Loops and range()

- **Iterate over a sequence**

- The built-in `range()` function helps iterate over a range of numbers

```
for i in range(5):
```

```
    print(i) # Prints 0,1,2,3 and 4
```

- **Iterate over elements (for each)**

- Used with sequence type data-types like string, lists and tuples.

```
Word = "Hello"
```

```
for letter in word:
```

```
    print(letter) # Prints Hello character by character
```

General format: for condition:

```
    Statements;
```

Date Display

```
import datetime  
  
datedisplay=datetime.datetime.now()  
  
print(datedisplay.strftime("%Y/%-m/%-d"))  
# If the date is April 1st, 2012          Output: 2012/4/1  
# It would display as 2012/4/1  
  
  
  
datedisplay=datetime.datetime.now()  
  
print(datedisplay.strftime("%A, %B, %-d, %Y"))  
# If the date is April 1st, 2012          Output: Wednesday, April 1, 2012  
# Wednesday, April 1, 2012
```

Month, Day & Date Format Symbols

%b	Jan
%B	January
%m	01
%-m (for Linux) %#m (for Windows)	1

Day of Month	%d	01
Day of Month	%-d (for Linux) %#d (for Windows)	1
Day of Week	%A	Monday
Day of Week	%a	Mon

Functions

- Functions MUST be **defined** before they can be **called**
- Function headers are of the format [notice the ':']

```
def functionName(arg_1, arg_2, ..., arg_n):
```

- Note that **no return type** is specified
- Function names are case sensitive

```
(foo(...) != Foo(...) != FoO(...))
```

- Functions can have **default argument values**

```
def functionName(a, b=2, c=[]): # Note: Default argument  
# values are initialized  
# only once
```

Functions example

```
# This is a function
def foo(arg_1, arg_2):
    arg_2 = arg_1 * arg_2
    return arg_2

result_1 = foo(12, 3)      # Store the function
print(result_1)            # Outputs 36
print(foo(12, 3))         # Outputs 36
```

Include Files

- Include “hello.py” within another python file as

```
import hello  
from hello import hello #imports the hello() from  
#hello.py
```

- The file hello.py might look like:

```
def Hello():  
    print("Hello")
```

- In the aforementioned python file, the Hello() function can be called as

```
hello.Hello()
```

- Using '*' allows importing all submodules from a package

```
from packageName import *
```

Classes in Python

- Syntax:

```
class className:  
    statement
```

- To instantiate a class object, use function notation
 - A constructor can be defined as
- ```
def __init__(self):
 statement
```
- The dot ('.') notation can be used to access class variables and methods
  - **Inheritance** can be done as:

```
class DerivedClassName(moduleName.BaseClassName):
 statement
```

# Code Examples

- All following code samples from “The Python Tutorial” at:

<https://docs.python.org/3/tutorial/index.html>

# String Examples

```
>>> # 3 times 'un', followed by 'ium'
>>> 3 * 'un' + 'ium'
'unununium'
```

Strings can be *indexed* (subscripted), with the first character having index 0. There is no separate character type; a character is simply a string of size one:

```
>>> word = 'Python'
>>> word[0] # character in position 0
'P'
>>> word[5] # character in position 5
'n'
```

Indices may also be negative numbers, to start counting from the right:

```
>>> word[-1] # last character
'n'
>>> word[-2] # second-last character
'o'
>>> word[-6]
'P'
```

Note that since  $-0$  is the same as 0, negative indices start from  $-1$ .

# List Examples

All slice operations return a new list containing the requested elements. This means that the following slice returns a **shallow copy** of the list:

```
>>> squares[:]
[1, 4, 9, 16, 25]
```

>>>

Lists also support operations like concatenation:

```
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

>>>

Unlike strings, which are **immutable**, lists are a **mutable** type, i.e. it is possible to change their content:

```
>>> cubes = [1, 8, 27, 65, 125] # something's wrong here
>>> 4 ** 3 # the cube of 4 is 64, not 65!
64
>>> cubes[3] = 64 # replace the wrong value
>>> cubes
[1, 8, 27, 64, 125]
```

>>>

# List Examples (cont'd)

Assignment to slices is also possible, and this can even change the size of the list or clear it entirely:

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g'] >>>
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>> letters
['a', 'b', 'f', 'g']
>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>> letters
[]
```

# List Examples (cont'd)

```
>>> vec = [-4, -2, 0, 2, 4]
>>> # create a new list with the values doubled
>>> [x*2 for x in vec]
[-8, -4, 0, 4, 8]
>>> # filter the list to exclude negative numbers
>>> [x for x in vec if x >= 0]
[0, 2, 4]
>>> # apply a function to all the elements
>>> [abs(x) for x in vec]
[4, 2, 0, 2, 4]
>>> # call a method on each element
>>> freshfruit = [' banana', ' loganberry ', 'passion fruit ']
>>> [weapon.strip() for weapon in freshfruit]
['banana', 'loganberry', 'passion fruit']
>>> # create a list of 2-tuples like (number, square)
>>> [(x, x**2) for x in range(6)]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
>>> # the tuple must be parenthesized, otherwise an error is raised
>>> [x, x**2 for x in range(6)]
 File "<stdin>", line 1, in <module>
 [x, x**2 for x in range(6)]
 ^
SyntaxError: invalid syntax
>>> # flatten a list using a listcomp with two 'for'
>>> vec = [[1,2,3], [4,5,6], [7,8,9]]
>>> [num for elem in vec for num in elem]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# Looping examples

```
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrat
>>> for w in words:
... print(w, len(w))
...
cat 3
window 6
defenestratate 12
```

## Use of range

```
range(5, 10)
5, 6, 7, 8, 9

range(0, 10, 3)
0, 3, 6, 9

range(-10, -100, -30)
-10, -40, -70
```

# Function example

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
 while True:
 ok = input(prompt)
 if ok in ('y', 'ye', 'yes'):
 return True
 if ok in ('n', 'no', 'nop', 'nope'):
 return False
 retries = retries - 1
 if retries < 0:
 raise ValueError('invalid user response')
 print(reminder)
```

# Sets Examples

```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'} >>>
>>> print(basket) # show that duplicates have been removed
{'orange', 'banana', 'pear', 'apple'}
>>> 'orange' in basket # fast membership testing
True
>>> 'crabgrass' in basket
False

>>> # Demonstrate set operations on unique letters from two words
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a # unique letters in a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b # letters in a but not in b
{'r', 'd', 'b'}
>>> a | b # letters in a or b or both
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b # letters in both a and b
{'a', 'c'}
>>> a ^ b # letters in a or b but not both
{'r', 'd', 'b', 'm', 'z', 'l'}
```

# Dictionary Examples

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'jack': 4098, 'sape': 4139, 'guido': 4127}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'jack': 4098, 'guido': 4127, 'irv': 4127}
>>> list(tel)
['jack', 'guido', 'irv']
>>> sorted(tel)
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```

The `dict()` constructor builds dictionaries directly from sequences of key-value pairs:

```
>>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

# Looping Techniques

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.items():
... print(k, v)
...
gallahad the pure
robin the brave
```

When looping through a sequence, the position index and corresponding value can be retrieved at the same time using the `enumerate()` function.

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
... print(i, v)
...
0 tic
1 tac
2 toe
```

To loop over two or more sequences at the same time, the entries can be paired with the `zip()` function.

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
... print('What is your {}? It is {}.'.format(q, a))
...
What is your name? It is lancelot.
What is your quest? It is the holy grail.
What is your favorite color? It is blue.
```

# Handling Exceptions

- Use `try ... except` statement for exception handling
- If no exception occurs, **except** clause will be skipped. Else, the **try** clause is stopped, and the matched exception clause will be executed.

```
while True:
```

```
 try:
```

```
 x = int(input("Please enter a number: "))
```

```
 break
```

```
 except ValueError:
```

```
 print("Oops! That was no valid number. Try
again...")
```

# Import Module

- A module is a file containing Python definitions and statement with the suffix '.py' appended
- Import a module by its name

```
import fibo
```

- Import the methods from a module

```
from fibo import fib, fib2
```

- Import all that a module defines

```
from fibo import *
```

# Flask

- Flask is a lightweight **WSGI (Web Server Gateway Interface)** web application framework
- WSGI is a Python standard defined in PEP 3333  
<https://www.python.org/dev/peps/pep-3333/>
- WSGI specifies a standard interface between web servers and Python web applications or frameworks
- Flask is designed to make getting started quickly and easily, with the ability to scale up to complex applications
- Flask offers suggestions but doesn't enforce any dependencies or project layout
- Documentation at: <https://palletsprojects.com/p/flask/>

# Installation

- Use native virtual environment for Python3

```
$ python3 -m venv venv
```

- Use third party for any version of Python older than 3.4 (includes 2.7)

```
$ virtualenv venv
```

```
$ source venv/bin/activate
```

- Install Flask in venv

```
(venv) $ pip3 install flask
```

# Flask Hello World

hello.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
 return "Hello, World"
```

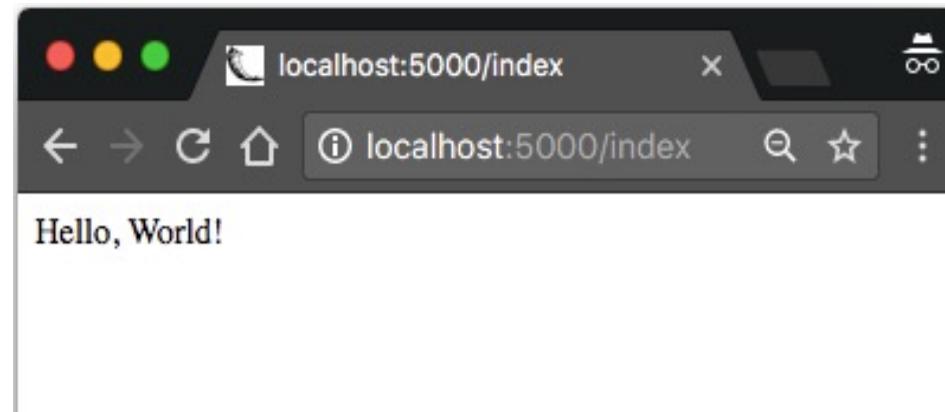
# Flask Hello World (cont'd)

```
(venv) $ export FLASK_APP=hello.py
```

```
(venv) $ flask run
```

```
Serving Flask app "hello_world" *
```

```
Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



# Templates

app/routes.py

```
from app import app

@app.route('/')
@app.route('/index')
def index():
 user = {'username': 'Miguel'}
 return '''
 <html>
 <head> <title>Home Page - Microblog</title> </head>
 <body>
 <h1>Hello, '''+ user['username'] + '''!</h1>
 </body> </html>'''
```



# render\_template

```
app/routes.py: Fake post in view function
from flask import render_template
from app import app

@app.route('/')
@app.route('/index')
def index():
 user = {'username': 'Miguel'}
 posts = [{ 'author': { 'username': 'John' },
 'body': 'Beautiful day in
Portland!', },
 {
 'author': { 'username': 'Susan' },
 'body': 'The Avengers movie was
so cool!' }]
 return render_template('index.html',
 title='Home', user=user,
 posts=posts)
```

# render\_template (con'd)

app/templates/index.html

```
<html> <head>

 {%- if title %}

 <title>{{ title }} - Microblog</title>

 {%- else %}

 <title>Welcome to Microblog</title>

 {%- endif %}

</head>

<body>

 <h1>Hi, {{ user.username }}!</h1>

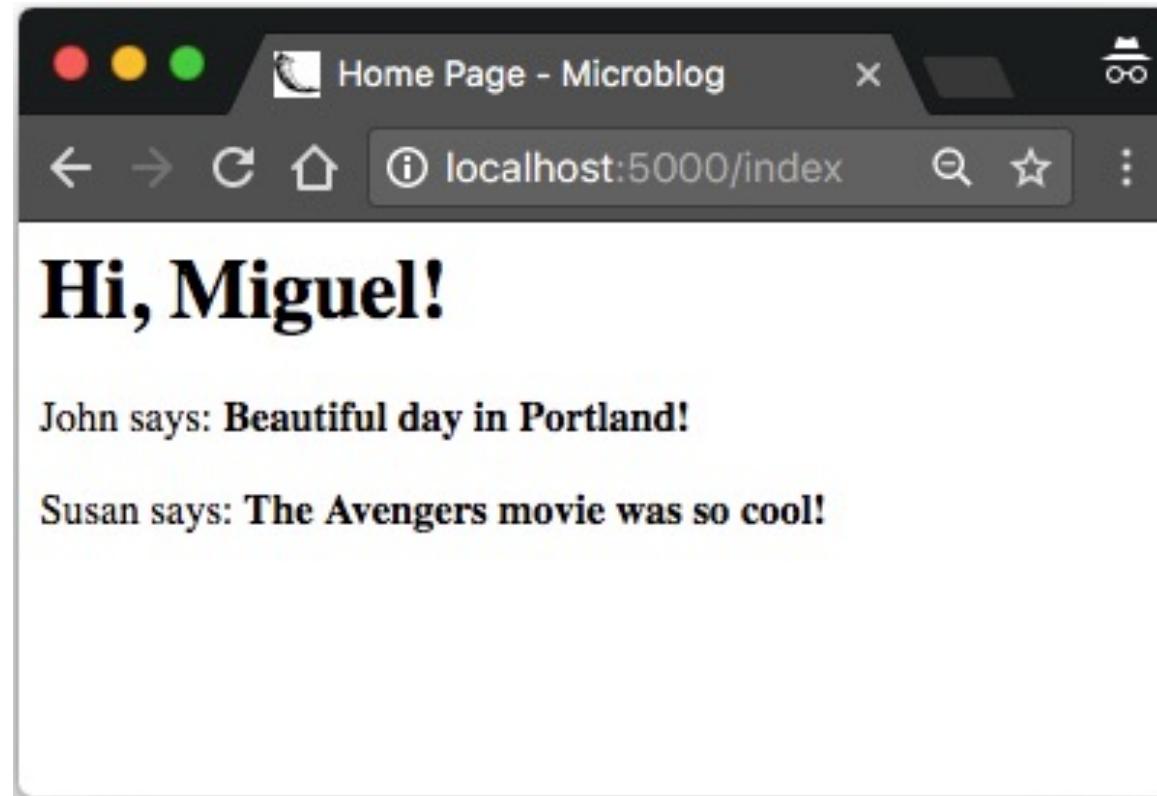
 {%- for post in posts %}

 <div><p>{{ post.author.username }} says:</p>
 {{ post.body }}</div>

 {%- endfor %}

</body> </html>
```

# Web Output



# Templates are old tech

Pre-Ajax coding patterns:

- Python templates + HTML { % ... % }
- PHP + HTML <? ... ?>
- ASP (Active Server Pages) + HTML <% ...%>
- JSP (Java Server Pages) + HTML <% ... %>

Post-Ajax coding patterns: All **RESTful APIs**, returning data only (JSON, XML) and no HTML

# RESTful Service in Flask

## rest.py

```
from flask import Flask, jsonify

app = Flask(__name__)

tasks = [
 { 'id': 1, 'title': u'Buy groceries', 'description':
u'Milk, Cheese, Pizza, Fruit, Tylenol', 'done': False },
 { 'id': 2, 'title': u'Learn Python', 'description':
u'Need to find a good Python tutorial on the web', 'done':
False }]

if __name__ == '__main__':
 app.run(debug=True)
```

# RESTful Service in Flask

```
#retrieve the list of task
@app.route('/todo/api/v1.0/tasks', methods=['GET'])
def get_tasks():
 return jsonify({ 'tasks': tasks})
```

See: <https://flask.palletsprojects.com/en/1.1.x/api/#flask.json.jsonify>

# Result

```
$ curl -i http://localhost:5000/todo/api/v1.0/tasks
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 294
Server: Werkzeug/0.8.3 Python/2.7.3
Date: Mon, 20 May 2013 04:53:53 GMT

{
 "tasks": [
 {
 "description": "Milk, Cheese, Pizza, Fruit, Tylenol",
 "done": false,
 "id": 1,
 "title": "Buy groceries"
 },
 {
 "description": "Need to find a good Python tutorial on the web",
 "done": false,
 "id": 2,
 "title": "Learn Python"
 }
]
}
```

# RESTful Service in Flask (2)

```
from flask import abort

#retrieve a task
@app.route('todo/api/v1.0/tasks/<int:task_id>', methods=['GET'])
def get_task(task_id):
 task = [task for task in tasks if task['id'] == task_id]
 if len(task) == 0:
 abort(404)
 return jsonify({'task': task[0]})
```

# Result

```
$ curl -i http://localhost:5000/todo/api/v1.0/tasks/2
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 151
Server: Werkzeug/0.8.3 Python/2.7.3
Date: Mon, 20 May 2013 05:21:50 GMT

{
 "task": {
 "description": "Need to find a good Python tutorial on the web",
 "done": false,
 "id": 2,
 "title": "Learn Python"
 }
}
```

# Send Static File

- Put index.html into the static folder (same for CSS and JS files)
- Send the static file using send\_static\_file
- Can also use send\_from\_directory

```
app = Flask(__name__)

@app.route('/')
def homepage():
 return app.send_static_file("index.html")
```

# Requests: HTTP for Humans

- Simple HTTP library for Python
- See: <https://requests.readthedocs.io/en/master/>
- Supports Python 2.7 & 3.4–3.10

```
>>> import requests
>>> payload = { 'key1': 'value1', 'key2': 'value2' }
>>> r = requests.get('https://api.github.com/events',
params=payload)
>>> r.json()
[{'repository': {'open_issues': 0, 'url': 'https://github.com/...']}
```

# Django

- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.
- Django was designed to help developers take applications from concept to completion as quickly as possible.
- Django takes security seriously and helps developers avoid many common security mistakes.
- Some of the busiest sites on the Web leverage Django's ability to quickly and flexibly scale.
- Documentation at: <https://www.djangoproject.com/>

# QuickStart

- Installation

```
$ python -m pip install Django
```

- Create a Django project

```
$ django-admin startproject mysite
```

- manage.py:

A command-line utility that lets you interact with this Django project in various ways.

- mysite/settings.py:

Settings/configuration for this Django project.

- mysite/urls.py:

The URL declarations for this Django project; a “table of contents” of your Django-powered site.

```
mysite/
 manage.py
mysite/
 __init__.py
 settings.py
 urls.py
 asgi.py
 wsgi.py
```

# Development Server

```
$ python manage.py runserver
```

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.

February 05, 2020 - 15:50:53
Django version 3.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

# Creating the First App

- Create the polls app

```
$ python manage.py startapp polls
```

- Edit views.py

```
from django.http import HttpResponse
```

```
def index(request):
 return HttpResponse("Hello, world. You're at the
polls index.")
```

```
polls/
 __init__.py
 admin.py
 apps.py
 migrations/
 __init__.py
 models.py
 tests.py
 views.py
```

# Change the URL Config

- polls/urls.py

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
 path('', views.index,
name='index'),
]
```

- mysite/urls.py

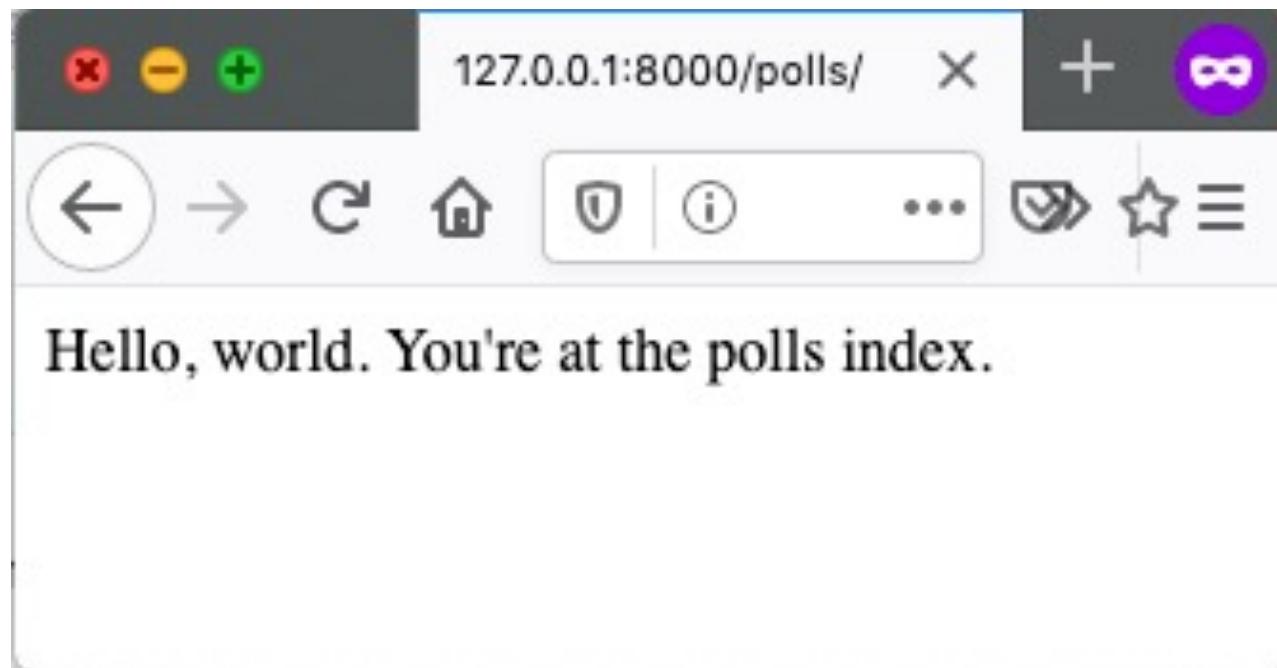
```
from django.contrib import admin
from django.urls import include,
path
```

```
urlpatterns = [
 path('polls/',
include('polls.urls')),
 path('admin/',
admin.site.urls),
]
```

# Result

- Run the server

```
$ python manage.py runserver
```



# Python on Google Cloud

- To quickly deploy Python applications on Google Cloud, see:  
<https://cloud.google.com/python>
- Cloud Code , IDE Integration with IntelliJ, PyCharm and VSCode:  
<https://cloud.google.com/code/>
- Building a Python 3 App on App Engine using Flask:  
<https://cloud.google.com/appengine/docs/standard/python3/building-app>

# **Document Object Model**

# What is DOM

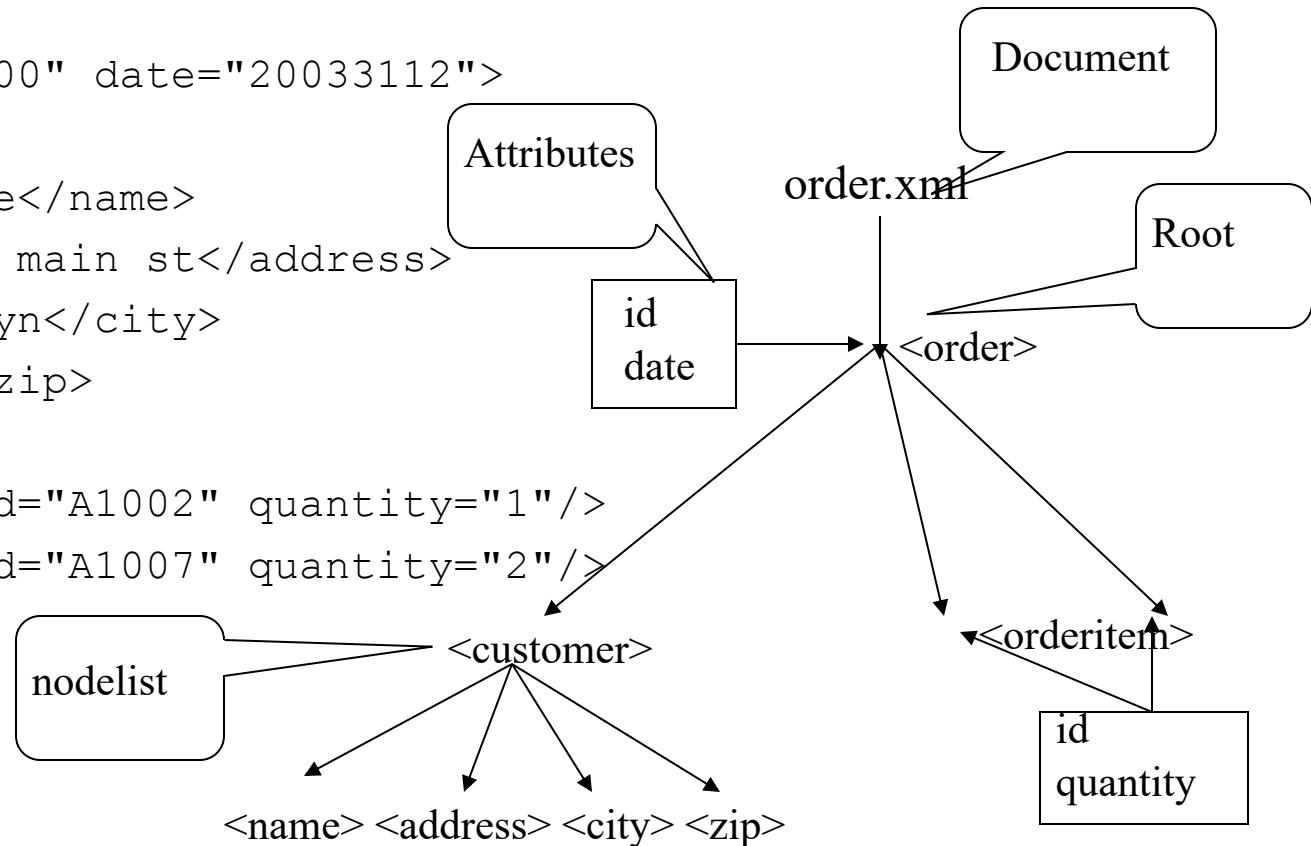
- The Document Object Model (DOM) is a programming interface for XML documents.
  - It defines the way an XML document can be accessed and manipulated
  - this includes HTML documents
- The XML DOM is designed to be used with **any programming language** and any operating system.
- The DOM represents an XML file as a tree
  - The documentElement is the top-level of the tree. This element has one or many childNodes that represent the branches of the tree.

# Version History

- **DOM Level 1** concentrates on HTML and XML document models. It contains functionality for document navigation and manipulation. See:
  - <http://www.w3.org/DOM/>
- **DOM Level 2** adds a stylesheet object model to DOM Level 1, defines functionality for manipulating the style information attached to a document, and defines an event model and provides support for XML namespaces. The DOM Level 2 specification is a set of 6 released W3C Recommendations, see:
  - <https://www.w3.org/DOM/DOMTR#dom2>
- **DOM Level 3** consists of 3 different specifications (Recommendations)
  - DOM Level 3 Core, Load and Save, Validation,  
<http://www.w3.org/TR/DOM-Level-3/>
- **DOM Level 4 (aka DOM4)** consists of 1 specification (Recommendation)
  - W3C DOM4, <http://www.w3.org/TR/domcore/>
    - Consolidates previous specifications, and moves some to HTML5
- See All **DOM Technical Reports** at:
  - <https://www.w3.org/DOM/DOMTR>
- Now DOM specification is **DOM Living Standard (WHATWG)**, see:
  - <https://dom.spec.whatwg.org>

## HTML or XML files viewed as a tree - order.xml

```
<order id="100" date="20033112">
<customer>
<name>S Spade</name>
<address>123 main st</address>
<city>Brooklyn</city>
<zip>10012</zip>
</customer>
<orderitem id="A1002" quantity="1"/>
<orderitem id="A1007" quantity="2"/>
</order>
```



DOM represents documents as a hierarchy of node objects  
Some types of nodes have children

# Some Useful DOM Functions

- **document is the root element**

- **document.getElementById("sample")**

- Returns the one location defined by id=sample, e.g.

```
document.getElementById("sample").style.color="rgb("FF","00","00");
assigns color red to the text
```

- **document.getElementsByTagName("font")**

- returns ALL font elements, e.g.

```
arrayOfDocFonts = document.getElementsByTagName("font");
```

- **innerHTML**

- assigns a new value to text defined by id=counter2

```
document.getElementById("counter2").innerHTML = "Number of clicks = 1";
```

- **style.left, style.color properties**

- one can also assign values to CSS properties, e.g.

```
document.getElementById('counter1').style.left = '500px';
```

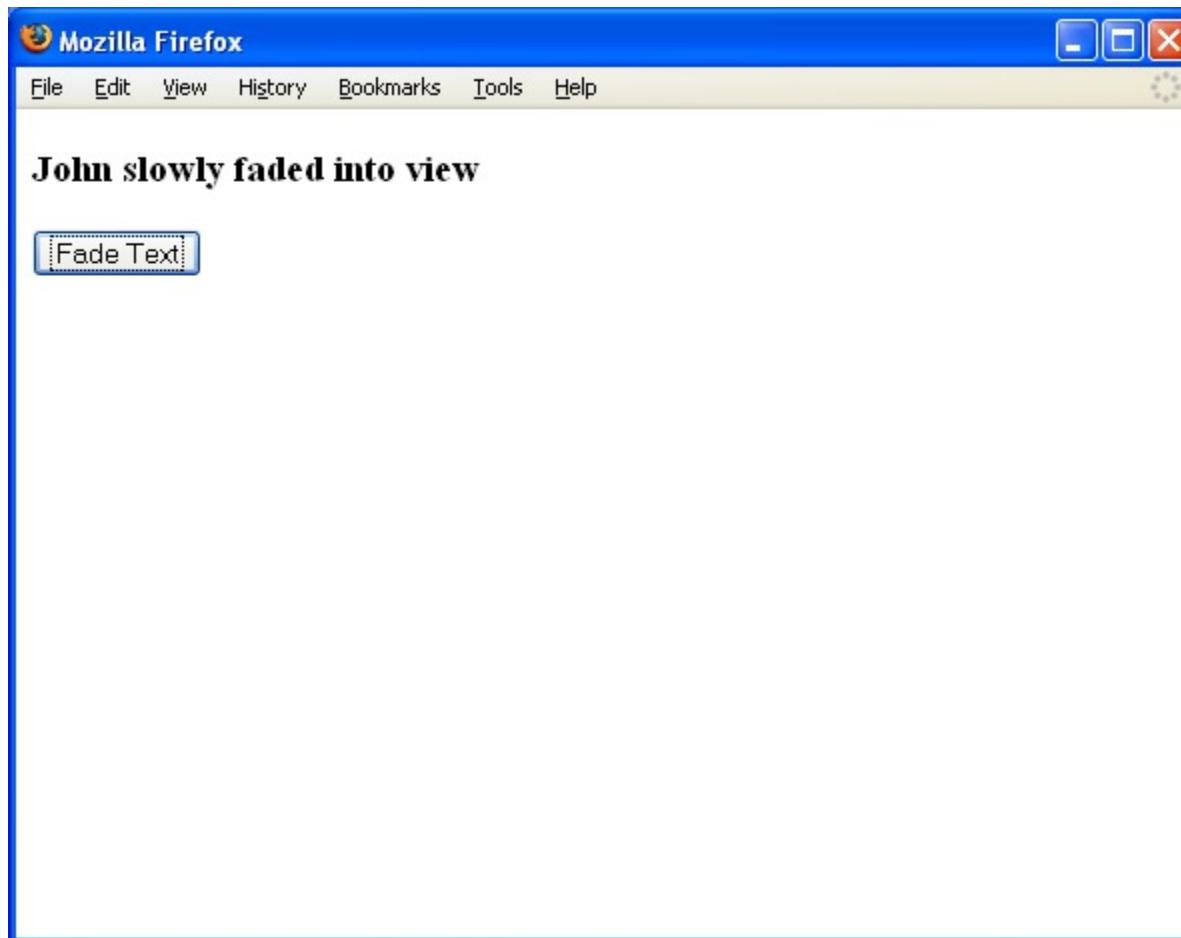
- the following slides have more examples

## **Example 1: Using DOM Functions to Alter a Page - Fading Text**

```
<html><head>
<script language="JavaScript1.2">
hex=255 // Initial color value.
function fadetext(){
if(hex>0) { //If color is not black yet
hex-=11; // increase color darkness
document.getElementById("sample").style.color="rgb("+hex+",
+hex+"," +hex+ ")";
setTimeout ("fadetext()",20); }
else hex=255 //reset hex value }
</script></head><body>
<div id="sample" style="width:100%">
<h3>John slowly faded into view</h3></div>
<button onClick="fadetext()">Fade Text</button>
</body></html>
```

Go to: <http://csci571.com/examples.html#dom>

# Browser Output

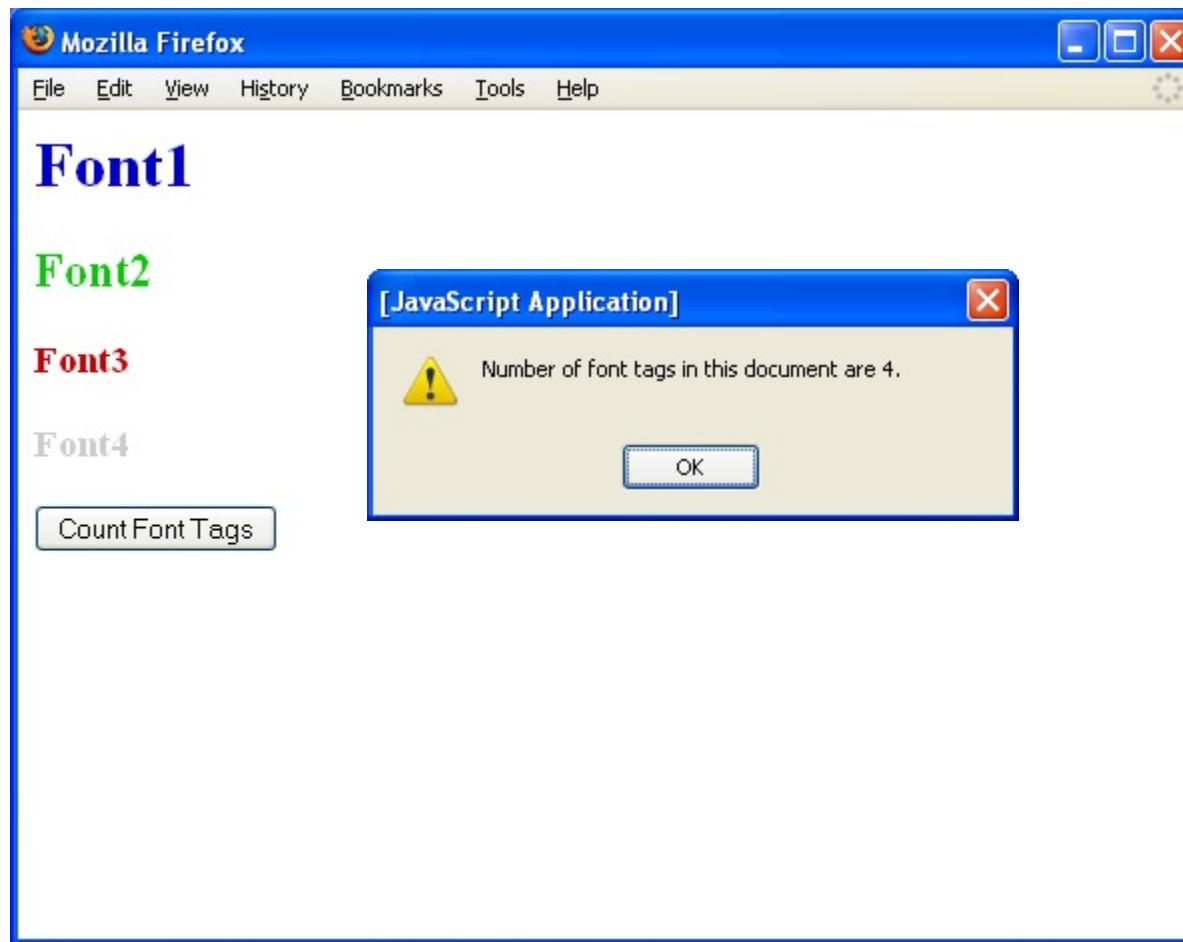


See <http://csci571.com/examples/dom/ex1.html>

## Example 2: Extracting Elements by Tag Name

```
<html><head>
<SCRIPT LANGUAGE="JavaScript">
function handleAllTags()
{
 var arrayOfDocFonts;
 if (document.all || document.getElementById)
 { arrayOfDocFonts = document.getElementsByTagName("font"); }
 else { document.write("Unrecognized Browser Detected"); }
 alert("Number of font tags in this document are " +
arrayOfDocFonts.length + ".");
}
</SCRIPT> </head><body>
<h1>Font1</h1>
<h2>Font2</h2>
<h3>Font3</h3>
<h3>Font4</h4>
<input type=button onclick="handleAllTags()"
 value="Count Font Tags">
</body></html>
```

# Browser Output



# innerHTML Property

- The **innerHTML** property of an element was first introduced as non-standard extension by **Microsoft** in Internet Explorer
- Mozilla- and Gecko-based browsers (Firefox), WebKit as well as IE decided to support it, even though it was not part of the standard
- **innerHTML** is widely used in Ajax-based sites (see *later in the course*)
- Elements that do not have both an opening and closing tag cannot have an **innerHTML** property.
- The **innerHTML** property takes a string that specifies a valid combination of text and elements.
- When the **innerHTML** property is set, the given string completely replaces the existing content of the object. If the string contains HTML tags, the string is parsed and formatted as it is placed into the document
- Example 1: changes the color of the counter:

```
<DIV ID="counter2">Number of clicks = 0</DIV>
```

- This line sets the innerHTML by replacing the entire text as follows:

```
document.getElementById("counter2").innerHTML = "
Number of clicks = " + hits2 + "";
```

- **innerHTML has been added to the HTML5 specification, DOM Parsing and Serialization specification ([sec. 7.1, Attributes](#)):**

<https://www.w3.org/TR/DOM-Parsing/#widl-Element-innerHTML>

<https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML>

## Example 3: Setting innerHTML

- Example: update a counter by clicking a button

```
<DIV ID="counter">Number of clicks = 0</DIV>

 <INPUT TYPE="button"
 VALUE="Increment Counter"
 onclick="updateMessage()">

<SCRIPT LANGUAGE="JavaScript">

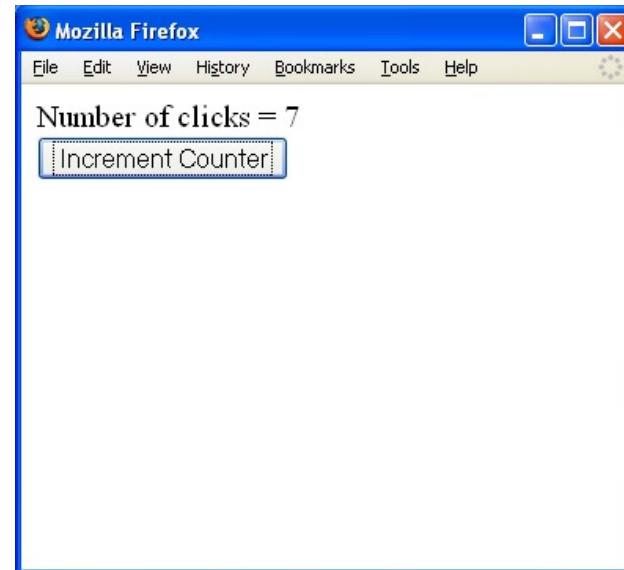
 var hits = 0;

 function updateMessage() {

 hits += 1;

 document.getElementById("counter").innerHTML =
 "Number of clicks = " + hits; }

</SCRIPT>
```



# Final Note on innerHTML

- **Suggested Rule:** If you use innerHTML, don't use the `+=` operator with innerHTML for the following reason:
  - Every time innerHTML is set, the HTML must be parsed, a DOM constructed, and inserted into the document. This takes time.
  - For example, if `elm.innerHTML` has thousands of divs, tables, lists, images, etc, then calling `.innerHTML += ...` is going to cause the parser to re-parse all that stuff over again. This could also break references to already constructed DOM elements and cause other chaos. In reality, all you want to do is append a single new element to the end.
- E.g., it is better to just call `appendChild`:

```
var newElement = document.createElement('div');
newElement.innerHTML = '<p>Hello World!</p>';
elm.appendChild(newElement);
```

This way, the existing contents of `elm` are not parsed again.
- See:  
<https://developer.mozilla.org/en-US/docs/Web/API/Element.innerHTML>
- <script> elements inserted using innerHTML do not execute (HTML5):  
<http://www.w3.org/TR/2008/WD-html5-20080610/dom.html#innerhtml0>

## **Example 4: Moving Objects Horizontally**

- The browser-independent W3C Standard way to set and get an element's position is via the STYLE object's left and top properties
- the W3C DOM Standard defines a "**left**", "**right**", "**top**", "**bottom**" properties of the style object
- **E.g., Moving Objects Horizontally**

```
<INPUT ID="counter1" STYLE="position:relative; left:0px"
 TYPE="button" VALUE="Move Button"
 onclick="document.getElementById('counter1').style.left
 = '500px';">
```

- **E.g., Moving Objects Vertically**

```
<INPUT ID="counter1" STYLE="position:relative; top:0px"
 TYPE="button" VALUE="Move Button"
 onclick="document.getElementById('counter1').style.top =
 '15px';">
```

## **Another Example of Moving Objects on a Web Page**

- The following code segment adds 50 pixels to the button's left property, every time the user clicks the button:

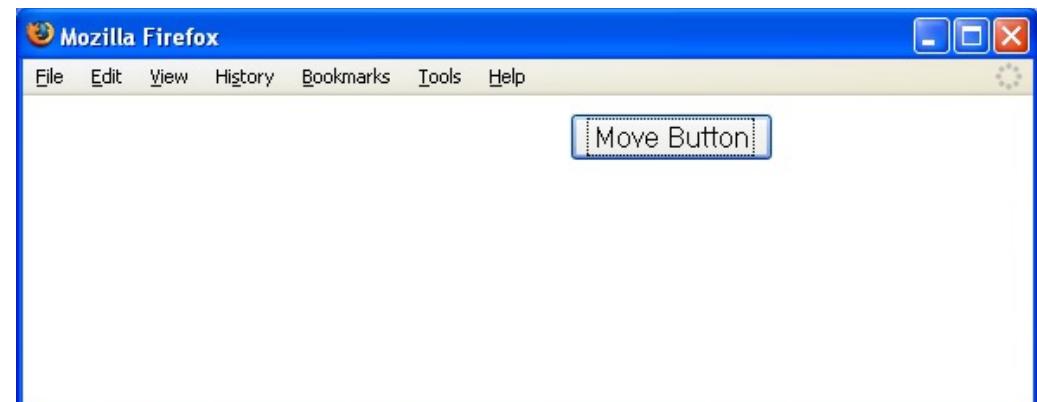
```
<INPUT ID="counter1" STYLE="position:relative; left:0px"
TYPE="button" VALUE="Move Button"
onclick="handleClick()>

<SCRIPT LANGUAGE="JavaScript">

var obj = document.getElementById('counter1');
var xlocation = parseInt(obj.style.left);
alert(xlocation);

function handleClick() { xlocation += 50;
document.getElementById('counter1').style.left =
xlocation + "px"; } </SCRIPT>
```

# Browser Output

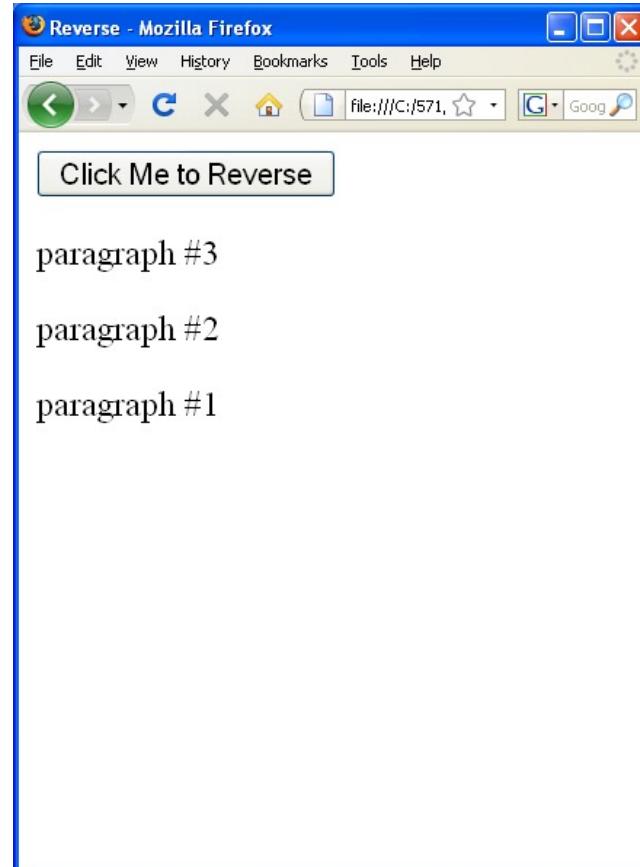
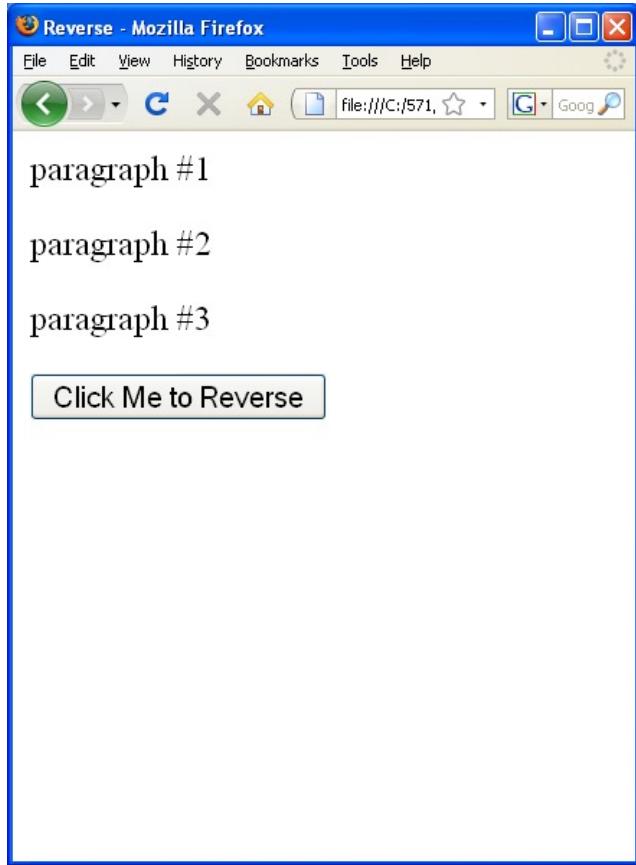


As the button is clicked  
it moves to the right

## Example 5:Reversing the Nodes of a Document

```
<head><title>Reverse</title> <script>
function reverse(n)
{ // Reverse the order of the children of Node n
 var kids = n.childNodes; // Get the list of children
 var numkids = kids.length; // Figure out how many
 children there are
 for(var i = numkids-1; i >= 0; i--) { // Loop backward
 through the children
 var c = n.removeChild(kids[i]); // Remove a child
 n.appendChild(c); // Put it back at its new position
 }
</script> </head> <body> <p>paragraph #1<p>paragraph
#2<p>paragraph #3 <p>
<button onclick="reverse(document.body);" >Click Me to
Reverse</button> </body>
```

# Browser Output



# **XMLHttpRequest Object**

- The XMLHttpRequest object is used to exchange data with a server
- With an XMLHttpRequest object you can:
  - Update a web page without reloading the page
  - Request data from a server after page has loaded
  - Receive data from a server after page has loaded
  - Send data to a server in the background
- All modern browsers (IE7+, Edge, Firefox, Chrome, Safari, etc.) have a built-in XMLHttpRequest object.
- **“Synchronous” XMLHttpRequest is in process of being removed** from web platform (will take many years). Developer Tools will provide warning or error.
- See XMLHttpRequest Editor's Draft at:

<https://dvcs.w3.org/hg/xhr/raw-file/tip/Overview.html>

- See **XMLHttpRequest Living Standard:**

<https://xhr.spec.whatwg.org>

# Loading XML file into the Parser (1)

```
<script type="text/javascript">
var xmlDoc;
function loadXML(url) {
 if (window.XMLHttpRequest)
 { // code for IE7+, Firefox, Chrome, Opera, Safari
 xmlhttp=new XMLHttpRequest();
 }
 else
 { // code for IE6, IE5 [Obsolete]
 xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
 }
 xmlhttp.open("GET",url,false); // 'false' = synchronous request
 xmlhttp.send(); // DEPRECATED
 xmlDoc=xmlhttp.responseXML; // properties of XMLHttpRequest
 return xmlDoc; // (file returned in responseXML
} // or responseText for JSON)
// processing the document goes here
</script>
```

# Loading XML file into the Parser

```
<bookstore>
 <book category="cooking">
 <title lang="en">Everyday Italian</title>
 <author>Giada De Laurentiis</author>
 <year>2005</year>
 <price>30.00</price> </book>
 <book category="children">
 <title lang="en">Harry Potter</title>
 <author>J K. Rowling</author>
 <year>2005</year>
 <price>29.99</price> </book>
 <book category="web">
 <title lang="en">XQuery Kick Start</title>
 <author>James McGovern</author>
 <author>Per Bothner</author>
 <author>Kurt Cagle</author>
 <author>James Linn</author>
 <author>Vaidyanathan Nagarajan</author>
 <year>2003</year>
 <price>49.99</price> </book>
 <book category="web" cover="paperback">
 <title lang="en">Learning XML</title>
 <author>Erik T. Ray</author>
 <year>2003</year>
 <price>39.95</price> </book>
</bookstore>
```

books.xml

## Loading XML file into the Parser (2)

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>

<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
 if (this.readyState == 4 && this.status == 200) {
 myFunction(this);
 }
};
xhttp.open("GET", "books.xml", true); // asynchronous
xhttp.send();

function myFunction(xml) {
 var xmlDoc = xml.responseXML;
 document.getElementById("demo").innerHTML =
 xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
}
</script>
</body>
</html>
```

This example reads "books.xml" into xmlDoc and retrieves the text value of the first <title> element in books.xml.

# **Firefox and I.E. / Edge Represent DOM Structures Differently**

- As a result, the node count for each is different
  - You can check this by printing out  
`document.write(x.length);`
- In Mozilla FF:
  - all whitespace in the text content of the original document are represented in the DOM
  - this does not include whitespace within tags
  - some text nodes will contain only whitespace
  - some text nodes will have whitespace at the beginning or end
- See the article "Whitespace in the DOM" at :  
[https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Whitespace\\_in\\_the\\_DOM](https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Whitespace_in_the_DOM)
- The solutions for handling these distinctions is to check the node type

# Node Types

Node Type	Named Constant
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE

## Another DOM Example

### A simple XML file for a book store

```
- <bookstore>
 - <book category="cooking">
 <title lang="en">Everyday Italian</title>
 <author>Giada De Laurentiis</author>
 <year>2005</year>
 <price>30.00</price>
 </book>
 + <book category="children"></book>
 + <book category="web"></book>
 + <book category="web" cover="paperback"></book>
</bookstore>
```

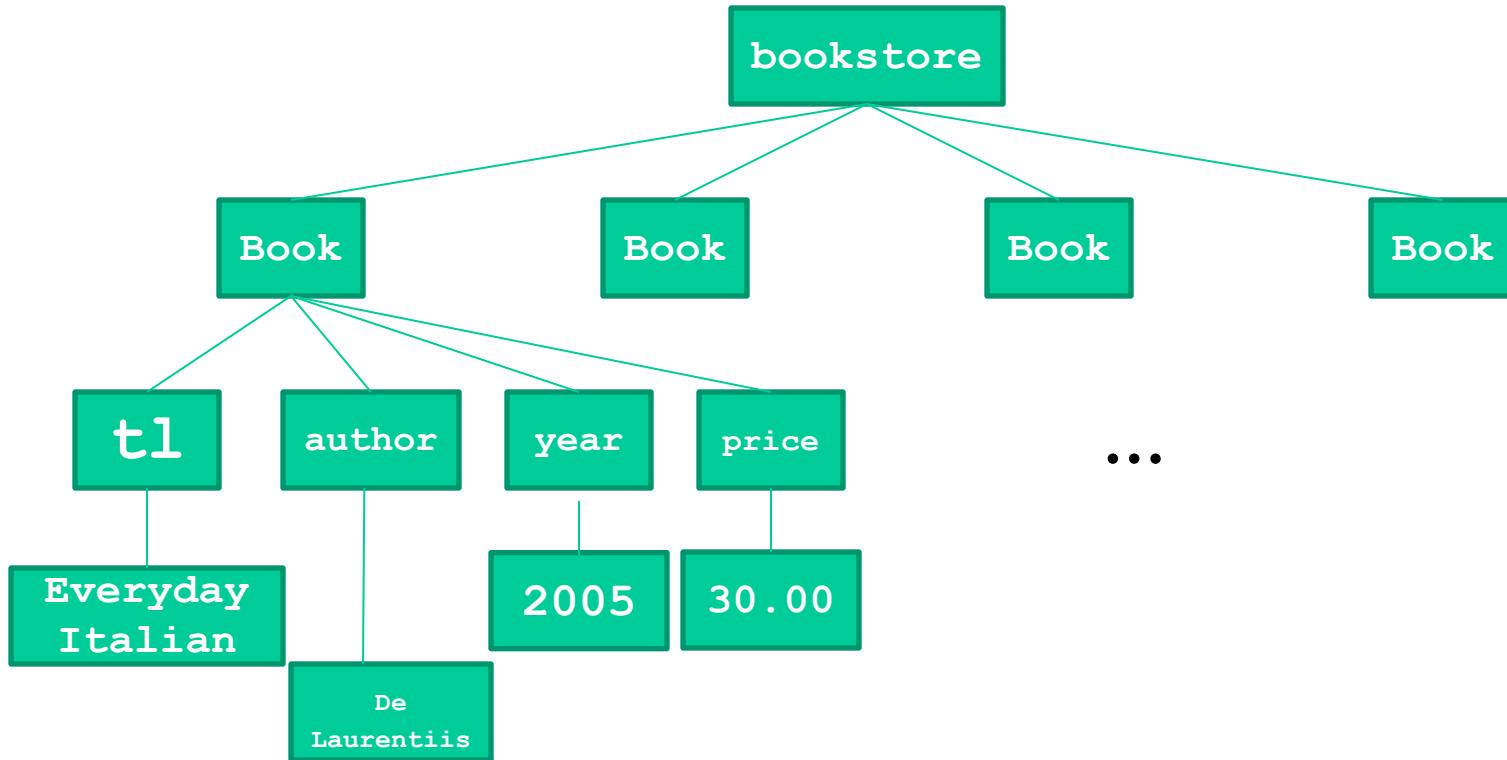
# Some Node Types in an XML File

## A Sample XML File

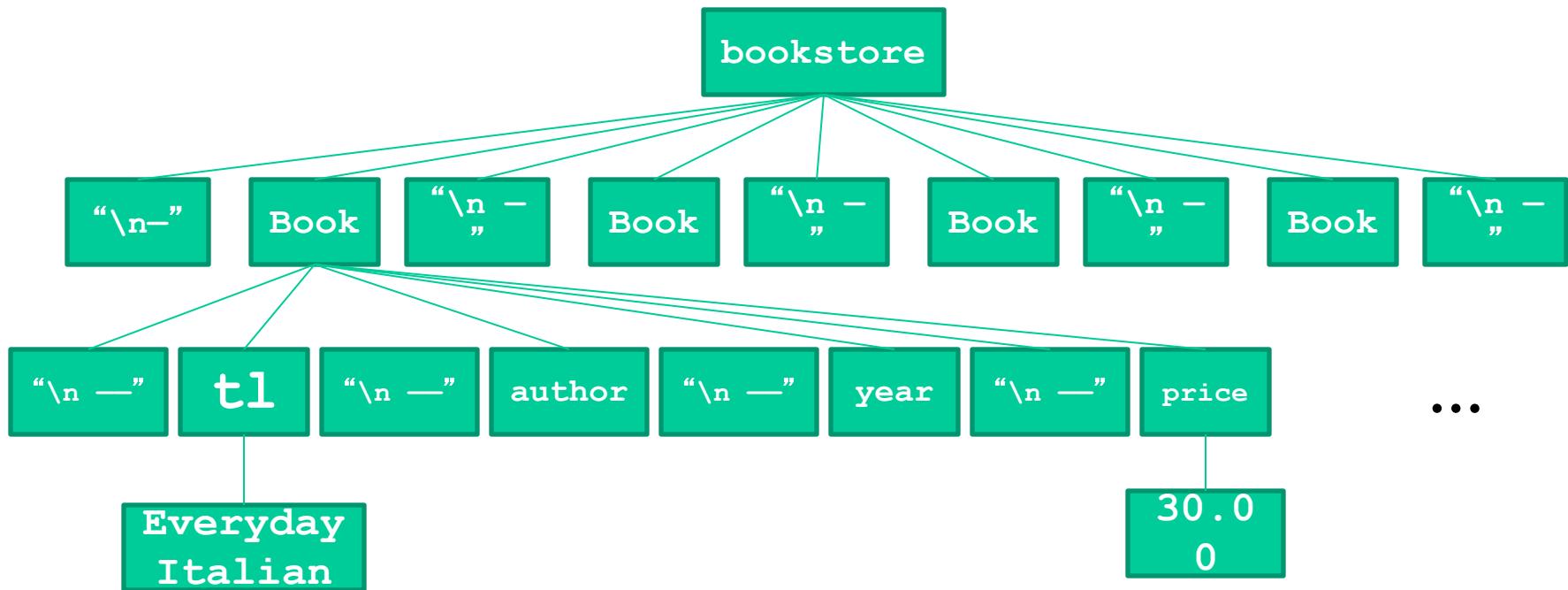
```
- <bookstore>
 <book category="cooking">
 <title lang="en">Everyday Italian</title>
 <author>Giada De Laurentiis</author>
 <year>2005</year>
 <price>30.00</price>
 </book>
+ <book>
+ <book>
</bookstore>
```

- Some possible node types
- ELEMENT\_NODE (type 1)
  - bookstore, book, title, author, year, price
- TEXT\_NODE (type 3)
  - “/n” nodes
  - “Everyday Italian”, “30.00”, ...
- Hint:
  - element nodes have children
  - text nodes are leaves
- x[i].nodeType == 1
  - tests for element nodes.
  - text nodes (like “\n”) are ignored

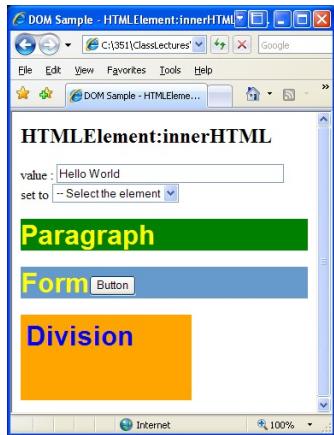
# An XML Tree in Internet Explorer



# The Same XML tree in FireFox



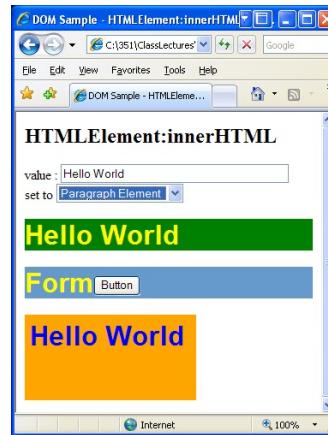
# Example 6: DOM and Three InnerHTML Examples



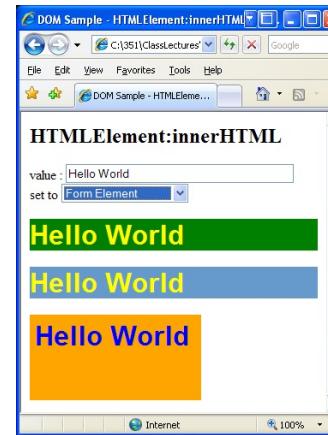
Initial page



Select division



Select paragraph



Select form

id definitions  
**t1 (orange)**  
**t2 (green)**  
**t3 (yellow)**

**setInnerHTML function**  
**defined here**

**Handle selection**

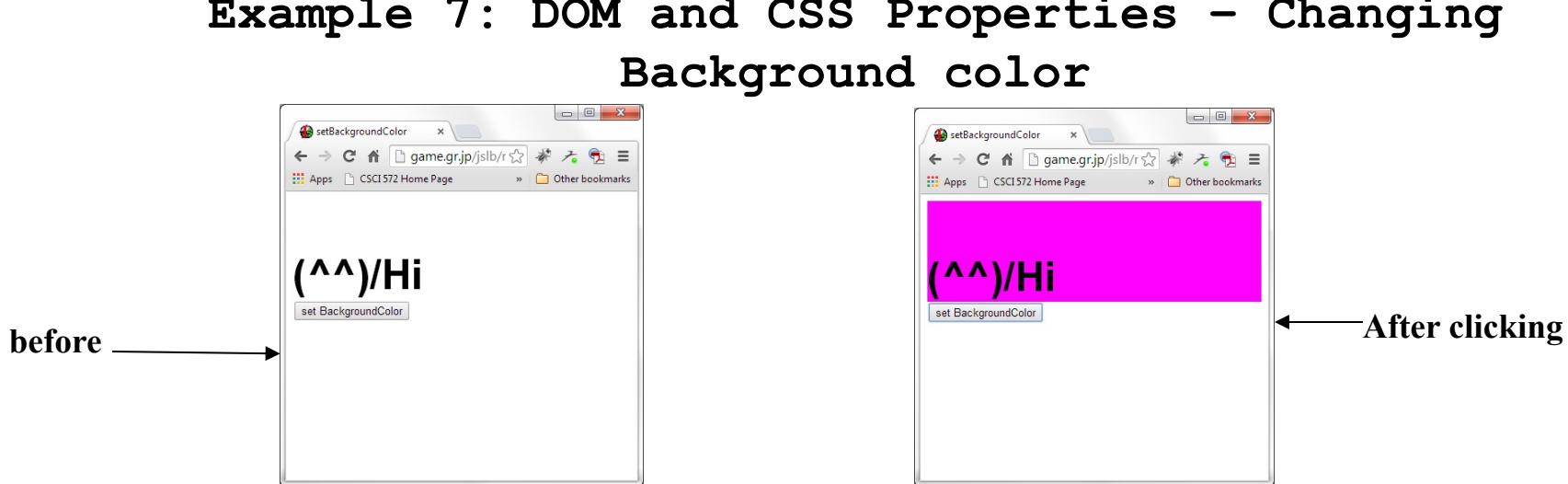
```

<DOM Sample - HTMLElement:innerHTML.htm - Notepad>
File Edit Format View Help
<HTML><HEAD><TITLE>DOM Sample - HTMLElement:innerHTML</TITLE>
<STYLE type=text/css>.smp {
 PADDING-RIGHT: 0.2em; PADDING-LEFT: 0.2em; PADDING-BOTTOM: 0.2em; WIDTH: 200px;
 PADDING-TOP: 0.2em; POSITION: absolute; HEIGHT: 100px}
#t1 {FONT-WEIGHT: 700; FONT-SIZE: 2em; COLOR: blue;
 FONT-FAMILY: sans-serif; BACKGROUND-COLOR: orange}
#t2 {FONT-WEIGHT: 700; FONT-SIZE: 2em; LEFT: 120px;
 COLOR: yellow; FONT-FAMILY: sans-serif; TOP: 200px; BACKGROUND-COLOR: green}
#t3 {FONT-WEIGHT: 700; FONT-SIZE: 2em; COLOR: yellow;
 FONT-FAMILY: sans-serif; BACKGROUND-COLOR: #6699cc}</STYLE>
<SCRIPT language=JavaScript type=text/javascript><!--
function notSupported() { alert('your browser is not supported.')} ;
function setInnerHTML(nm,value){
 if(nm == '') return;
 var element=document.getElementById?document.getElementById(nm):(document.all?document.all(nm):null)
 if(element){
 if(element.innerHTML){
 element.innerHTML=value;
 }
 else notSupported();
 }
 else notSupported();
} // --></SCRIPT></HEAD><BODY>
<H2>HTMLElement:innerHTML</H2>
<FORM>value : <INPUT size=40 value="Hello World" name=t>
set to <SELECT id=sel
onchange=setInnerHTML(this.options[this.selectedIndex].value,form.t.value);
name=sel> <OPTION value="" selected>-- Select the element<OPTION
value=t1>Division Element<OPTION value=t2>Paragraph Element<OPTION
value=t3>Form Element</OPTION></SELECT> </FORM>
<P id=t2>Paragraph</P>
<FORM id=t3 name=t3><INPUT type=button value=Button></FORM>
<DIV class=smp id=t1>Division</DIV>
<TABLE height=100 width=250>
 <TBODY><TR><TD></TD></TR></TBODY></TABLE></BODY></HTML>

```

3 select  
options:  
division,  
paragraph  
form

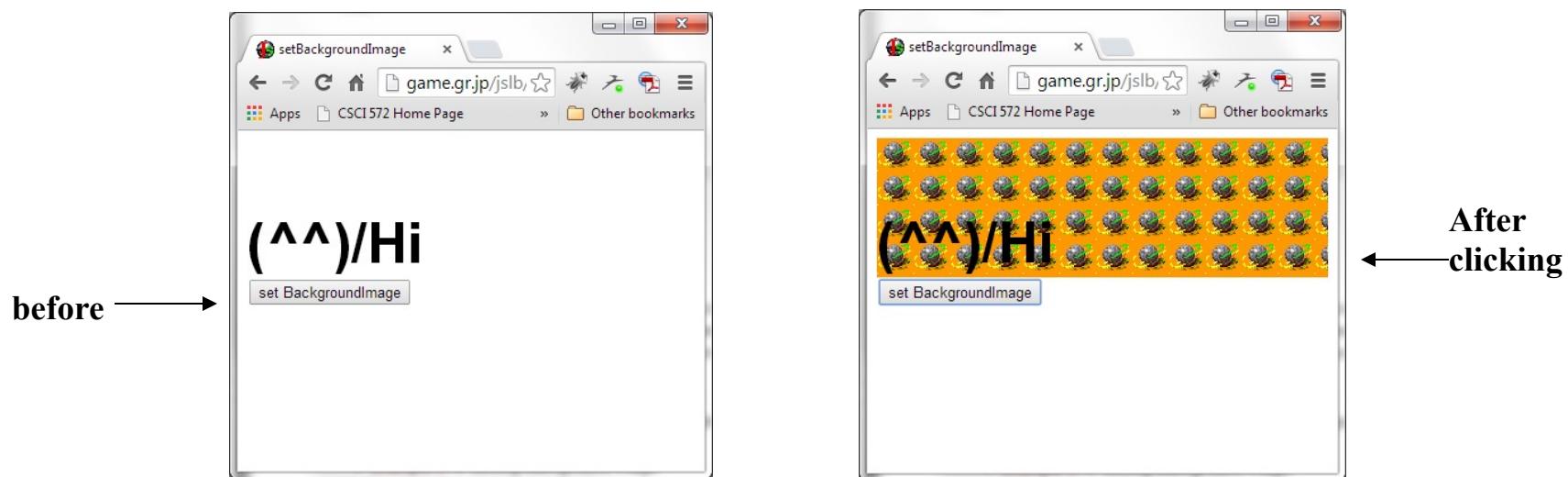
## Example 7: DOM and CSS Properties - Changing Background color



```
<HTML><HEAD><TITLE>setBackgroundColor</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function setBackgroundColor(id,bgcolor){
 document.getElementById(id).style.backgroundColor = bgcolor ;
}
//-->
</SCRIPT></HEAD><BODY>
<DIV ID="test" STYLE="font:900 50px Arial">
(^^) / Hi</DIV>
<FORM>
<INPUT TYPE="button"
 VALUE="set BackgroundColor"
onClick="if (document.getElementById) setBackgroundColor ('test', 'magenta ') "
</FORM>
</BODY></HTML>
```

See all examples at: <http://csci571.com/examples.html#dom>

## Example 8: DOM and CSS Properties - Changing Background Image

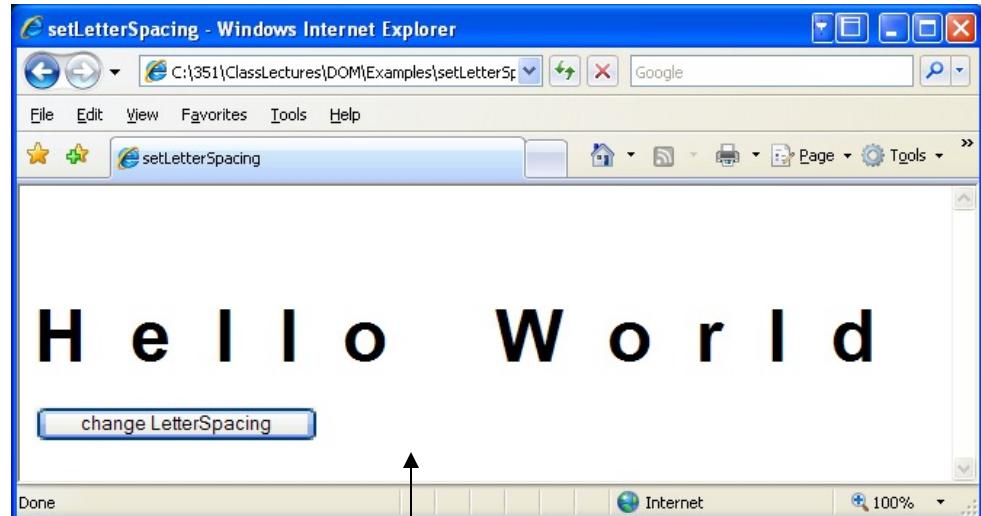


```
<HTML><HEAD><TITLE>setBackgroundColor</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function setBackgroundColor(id,image) {
 document.getElementById(id).style.backgroundImage = 'url('+image+')' ;
}
//-->
</SCRIPT>
</HEAD>
<BODY><DIV ID="test" STYLE="font:900 50px Arial">
(^^) /Hi</DIV>
<FORM>
<INPUT TYPE="button"
 VALUE="set BackgroundImage"
 onClick="if(document.getElementById)setBackgroundColor('test','tamas.gif')"
</FORM></BODY></HTML>
```

## Example 10: DOM and CSS Properties – Changing Letter Spacing



before



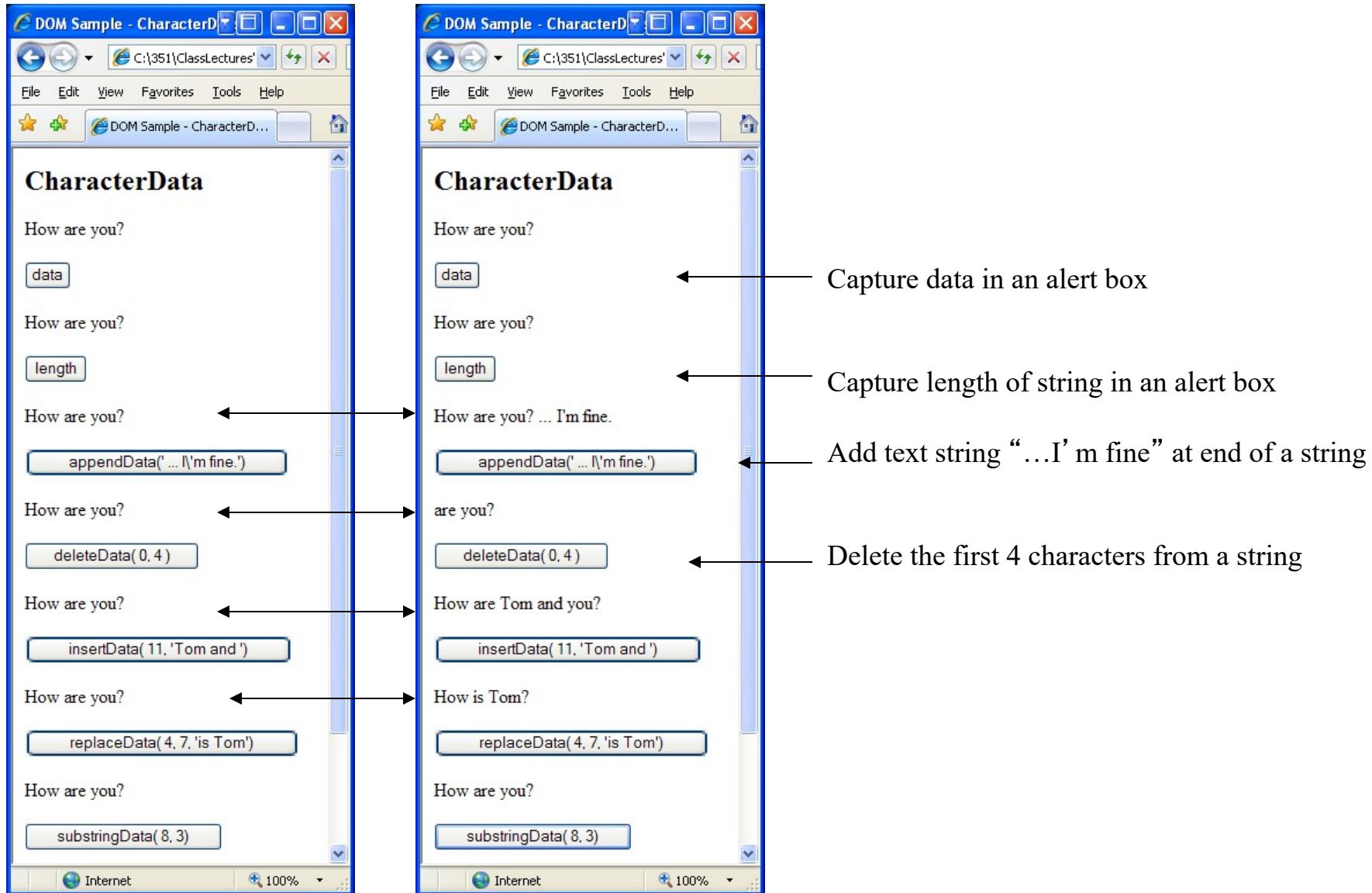
After clicking

A screenshot of a Microsoft Notepad window titled "setLetterSpacing.htm - Notepad". The code is as follows:

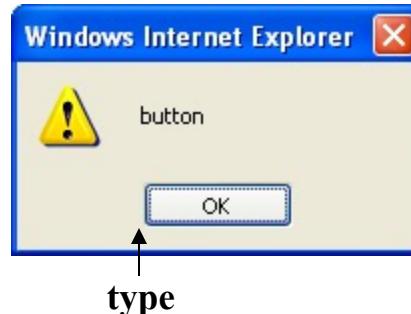
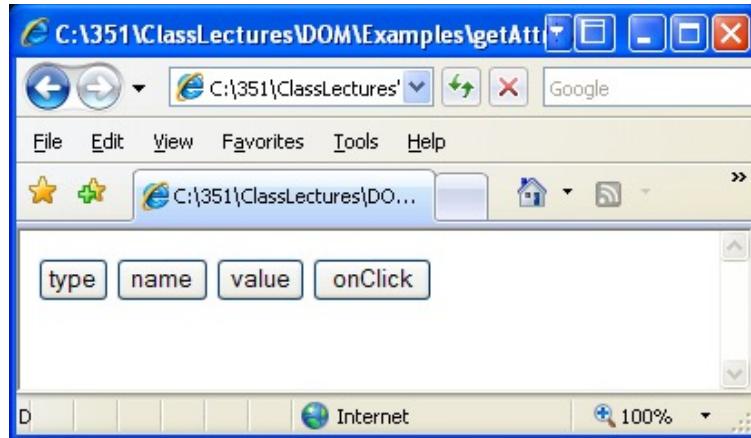
```
<HTML><HEAD><TITLE>setLetterSpacing</TITLE>
<SCRIPT type=text/javascript>
<!--
function setLetterSpacing(id,space){
 document.getElementById(id).style.letterSpacing = space ;
}
//-->
</SCRIPT></HEAD><BODY>
<DIV id=test style="FONT: 900 50px Arial">
Hello world</DIV>
<FORM><INPUT onclick="if(document.getElementById) setLetterSpacing('test','30px')"
type=button value="change LetterSpacing">
</FORM></BODY></HTML>
```

<http://csci571.com/examples/dom/setLetterSpacing.htm>

## Example 11: DOM and Manipulating Character Data



## Example 12: DOM and Retrieving Attributes



A screenshot of Notepad showing the HTML code for the example. The file is named "getAttribute.htm". The code defines a form with four input buttons, each with a different name and an onclick event that alerts the value of its attribute.

```
<HTML><HEAD><TITLE></TITLE></HEAD><BODY>
<FORM name=f0>
<INPUT onclick="alert(this.getAttribute('type'))" type=button value=type name=e0>
<INPUT onclick="alert(this.getAttribute('name'))" type=button value=name name=e1>
<INPUT onclick="alert(this.getAttribute('value'))" type=button value=value name=e2>
<INPUT onclick="alert(this.getAttribute('onClick'))" type=button value=onClick name=e3>
</FORM></BODY></HTML>
```

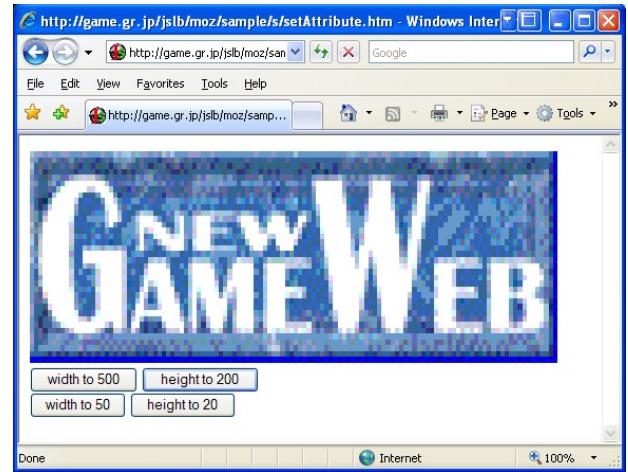
## Example 13: DOM and Setting Attributes



initial



Change Width to 500



Change Height to 200

```
setAttribute[1] - Notepad
File Edit Format View Help
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
 function getObj(id){
 return document.getElementById(id);
 }
//--> </SCRIPT></HEAD><BODY>
<IMG ID="imgTest"
 SRC="http://game.gr.jp/Gamelweb/NGwtools/images/logo01.gif">

<INPUT TYPE=button
 VALUE="width to 500"
 onClick="getObj('imgTest').setAttribute('width', 500)">
<INPUT TYPE=button
 VALUE="height to 200"
 onClick="getObj('imgTest').setAttribute('height', 200)">

<INPUT TYPE=button
 VALUE="width to 50"
 onClick="getObj('imgTest').setAttribute('width', 50)">
<INPUT TYPE=button
 VALUE="height to 20"
 onClick="getObj('imgTest').setAttribute('height', 20)">
</BODY></HTML>
```

# Nodes a DOM Can Contain

- An Example

```
<sentence> The &projectName; <! [CDATA[<i>project</i>]]>
 is <?editor: red><bold>important</bold><?editor: normal>.
</sentence>
 - contains an entity ref., CDATA section, processing
 instructions (<?...?>)
```

- Its DOM structure looks like this:

- + ELEMENT: sentence
  - + TEXT: The
  - + ENTITY REF: projectName
    - + COMMENT: The latest name we're using
    - + TEXT: Eagle
  - + CDATA: <i>project</i>
  - + TEXT: is
  - + PI: editor: red
  - + ELEMENT: bold
    - + TEXT: important
  - + PI: editor: normal

# Summary of XML/HTML node types and children

- *Document* -- Element (maximum of one), ProcessingInstruction, Comment, DocumentType (maximum of one)
- *DocumentFragment* -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- *DocumentType* -- no children
- *EntityReference* -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- *Element* -- Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
- *Attr* -- Text, EntityReference
- *ProcessingInstruction* -- no children
- *Comment* -- no children
- *Text* -- no children
- *CDATASection* -- no children
- *Entity* -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- *Notation* -- no children

## Example 14: A Longer DOM Example

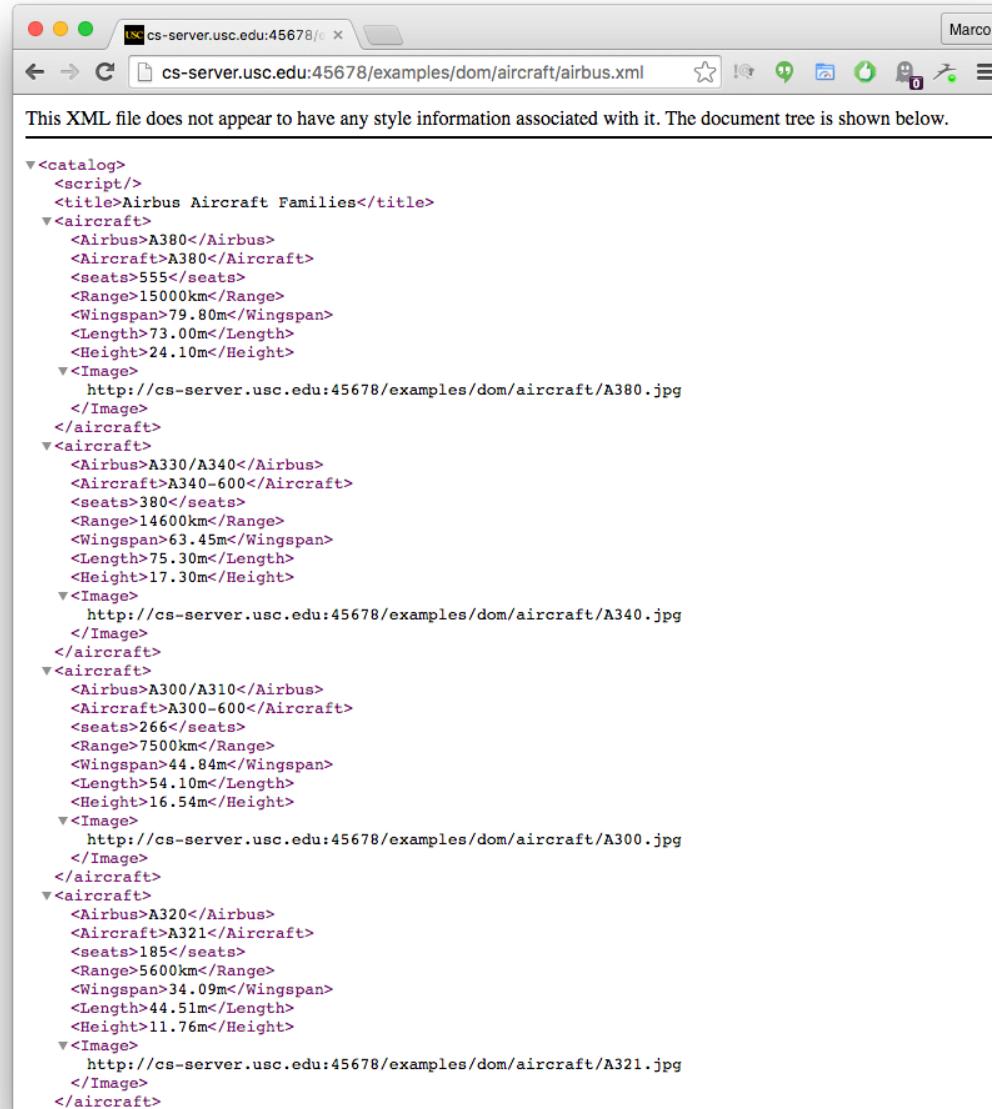
The diagram illustrates the process of reformatting XML data into an HTML page. It consists of two parts:

- Left Window (View Aircraft):** A screenshot of a web browser showing a form to "View the Commercial Aircrafts". The form includes a text input field with "airbus.xml" and a "Submit Query" button.
- Right Window (XML Parse Result):** A screenshot of a browser showing the "about:blank" page with the title "Airbus Aircraft Families". Below the title is a table displaying aircraft data, with four sample images of Airbus planes (A380, A340-600, A300-600, and A321) shown in the last column.

A single arrow originates from the bottom-left corner of the left window and points towards the top-right corner of the right window, indicating the flow of data from the XML source to the generated HTML output.

Given a URL of an XML file that describes a set of aircraft, re-format the data into an HTML page

# airbus.xml



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<catalog>
 <script/>
 <title>Airbus Aircraft Families</title>
 <aircraft>
 <Airbus>A380</Airbus>
 <Aircraft>A380</Aircraft>
 <seats>555</seats>
 <Range>15000km</Range>
 <Wingspan>79.80m</Wingspan>
 <Length>73.00m</Length>
 <Height>24.10m</Height>
 <Image>
 http://cs-server.usc.edu:45678/examples/dom/aircraft/A380.jpg
 </Image>
 </aircraft>
 <aircraft>
 <Airbus>A330/A340</Airbus>
 <Aircraft>A340-600</Aircraft>
 <seats>380</seats>
 <Range>14600km</Range>
 <Wingspan>63.45m</Wingspan>
 <Length>75.30m</Length>
 <Height>17.30m</Height>
 <Image>
 http://cs-server.usc.edu:45678/examples/dom/aircraft/A340.jpg
 </Image>
 </aircraft>
 <aircraft>
 <Airbus>A300/A310</Airbus>
 <Aircraft>A300-600</Aircraft>
 <seats>266</seats>
 <Range>7500km</Range>
 <Wingspan>44.84m</Wingspan>
 <Length>54.10m</Length>
 <Height>16.54m</Height>
 <Image>
 http://cs-server.usc.edu:45678/examples/dom/aircraft/A300.jpg
 </Image>
 </aircraft>
 <aircraft>
 <Airbus>A320</Airbus>
 <Aircraft>A321</Aircraft>
 <seats>185</seats>
 <Range>5600km</Range>
 <Wingspan>34.09m</Wingspan>
 <Length>44.51m</Length>
 <Height>11.76m</Height>
 <Image>
 http://cs-server.usc.edu:45678/examples/dom/aircraft/A321.jpg
 </Image>
 </aircraft>
```

## **HTML Code for the Initial Input**

```
<h1>View the Commercial Aircrafts </h1>
Enter XML file
<form name="myform" method="POST" id="location">
<input type="text" name="URL" maxlength="255"
 size="100" value="airbus.xml" />

<input type="button" name="submit" value="Submit
 Query" onClick="viewXML(this.form)" />
</form>
```

## viewXML Routine

```
function viewXML(what)
{var URL = what.URL.value;
 function loadXML(url) {
 if (window.XMLHttpRequest)
 {// code for IE7+, Firefox, Chrome, Opera, Safari
 xmlhttp=new XMLHttpRequest(); }
 else {// code for IE6, IE5
 xmlhttp=new ActiveXObject("Microsoft.XMLHTTP"); }
 xmlhttp.open("GET",url,false);
 xmlhttp.send();
 xmlDoc=xmlhttp.responseXML;
 return xmlDoc; }
 xmlDoc = loadXML(URL);
if (window.ActiveXObject) //if IE, simply execute script (due to async prop).
{ if (xmlDoc.parseError.errorCode != 0) {
 var myErr = xmlDoc.parseError;
 generateError(xmlDoc);
 hWin = window.open("", "Error", "height=300,width=340");
 hWin.document.write(html_text);
} else { generateHTML(xmlDoc);
 hWin = window.open("", "Assignment4", "height=800,width=600");
 hWin.document.write(html_text); }
} else //else if FF, execute script once XML object has loaded
{ xmlDoc.onload=generateHTML(xmlDoc);
 hWin = window.open("", "Assignment4", "height=800,width=600");
 hWin.document.write(html_text); }
hWin.document.close(); Copyright © 1999 - 2022 Ellis Horowitz
```

## generateXML Routine

```
function generateHTML(xmlDoc)
{ ELEMENT_NODE = 1; // MS parser doesn't define Node.ELEMENT_NODE
 root=xmlDoc.documentElement;
 html_text("<html><head><title>XML Parse Result</title></head><body>");
 html_text+="<table border='2'>";
 caption+xmlDoc.getElementsByTagName("title").item(0).firstChild.nodeValue;
 html_text+="


```

## generateXML Routine (cont'd)

```
if(header=="Wingspan") header="Wing Span";
if(header=="height") header="Height";
 html_text+=" " +header+" |"; } }

html_text+="</tr>";
// output out the values
for(i=0;i<planes.length;i++) //do for all planes
{ planeNodeList=planes.item(i).childNodes; //get properties of a plane
 html_text+="|"; //start a new row of the output table
 for(j=0;j<planeNodeList.length;j++)
 { if(planeNodeList.item(j).nodeType==ELEMENT_NODE)
 {
 if(planeNodeList.item(j).nodeName=="Image")
 //handle images separately
 html_text+=" </td>"; } else { html_text+=" " +planeNodeList.item(j).firstChild.nodeValue+" |"; } html_text+="</tr>"; } html_text+=""; html_text+="</table>"; html_text+=" |

```

## Example 15: Another DOM Example

### A simple XML file for a bookstore

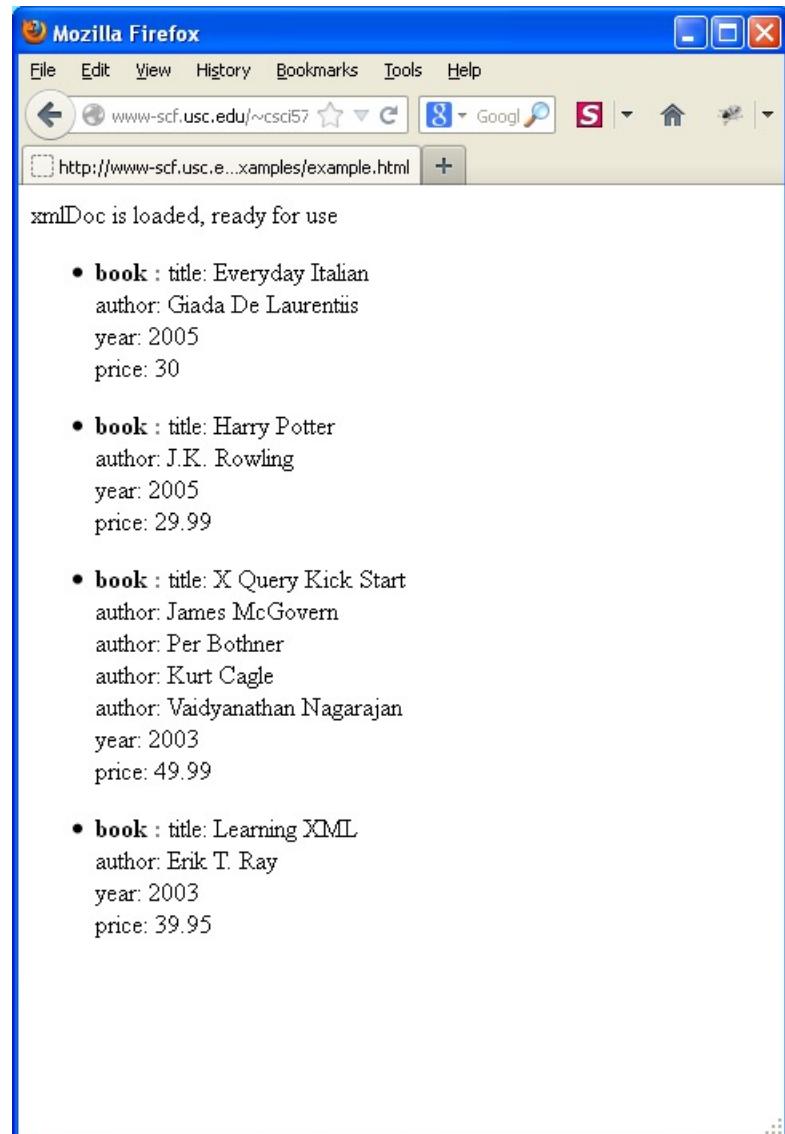
```
- <bookstore>
 - <book category="cooking">
 <title lang="en">Everyday Italian</title>
 <author>Giada De Laurentiis</author>
 <year>2005</year>
 <price>30.00</price>
 </book>
 + <book category="children"></book>
 + <book category="web"></book>
 + <book category="web" cover="paperback"></book>
</bookstore>
```

## Example Cont'd - function `xmlparse` traverses and outputs the books

```
function displayString(out) {
 var output = document.getElementById("output");
 output.innerHTML = out; }

function xmlparse() {
 var html = "";
 xmlDoc = loadXML("bookstore.xml");
 html += ("xmlDoc is loaded, ready for use
");
 var bookstore = xmlDoc.documentElement;
 for (i=0;i< bookstore.childNodes.length ;i++)
 { var book = bookstore.childNodes[i];
 if (book.nodeType==1)
 { html += ('');
 html += ('' +bookstore.childNodes[i].nodeName+ ' : ');
 y = book.childNodes;
 for (j=0;j<y.length;j++)
 { if (y[j].nodeType==1)
 { html += y[j].nodeName + ": "; //-> title, author etc
 html += y[j].childNodes[0].nodeValue; //-> text values
 html += ("
"); } }
 html += ('');
 } }
 displayString(html); }
</script></head><body><h2>This is the domtest web page</h2>
<input type="button" name="submit" value="Submit Query" onClick="xmlparse()" />
<noscript><div id="output"></div></body></html>
```

Before and After →



An alternate solution that makes use of “bookstore.children” instead of Childnodes can be found at

<http://csci571.com/examples/dom/example2.html> (Example 17)

## **Additional Reads**

DOM Examples at `w3schools.com`:

`https://www.w3schools.com/jsref/dom_obj_document.asp`

(hint: try all examples)

## **Lecture**

# **Forms and Common Gateway Interface Mechanism**

## **Forms**

- Used to create a set of pages that contain fields in which the viewer can select and supply information
  - Introduced very early in HTML 2.0
  - Allows WWW users to perform data entry
  - Permit direct interaction with customers for inquiries, registration, sales of products, and services
  - To create a capability requires two steps:
    - Use HTML form elements to create the pages that contain the form
    - Write a server-side script to process form data; this program must be placed so the WWW server can execute it

# The Original Set of User Interface Elements

<INPUT>

Text

File

**Browse**

Checkbox



Radio button



Submit

Reset

Password

<TEXTAREA>

A large rectangular input field with a double-headed vertical arrow indicating scrollability.

<SELECT>

A rectangular dropdown menu showing the option "Red" followed by a small square selection box.A vertical rectangular dropdown menu listing the options "Red", "Green", and "Blue" vertically, with a double-headed vertical arrow indicating scrollability.

# FORM Element and Some Attributes

- **Syntax**    `<FORM>...</FORM>`
- **Attribute Specifications**
  - ACTION=*URI* (form handler)
  - METHOD=[ *get* | *post* ] (HTTP method for submitting form)
    - GET is the default; form contents are appended to the URL
    - POST form contents to be sent as payload
  - ENCTYPE=*ContentType* (content type to submit form as)
    - **Defaults** to application/x-www-urlencoded which returns name/value pairs, separated by &, spaces replaced by + and reserved characters (like #) replaced by %HH, H a hex digit
  - ACCEPT-CHARSET=*Charsets* (supported character encodings)
  - TARGET=*FrameTarget* (frame to render form result in, in HTML4)  
**(a browsing context name or keyword, in HTML5, such as \_self, \_blank, \_parent, \_top, iframename)**
  - ONSUBMIT=*Script* (form was submitted)
  - ONRESET=*Script* (form was reset)
  - AUTOCOMPLETE (**HTML5 ONLY**) values completed by browser  
<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form>

## <INPUT> Tag

- Used inside <FORM> tag to specify a data-entry object
- Has attributes, here are a few
  - TYPE: User input type (default is TEXT)
  - NAME: Name of data entry object whose value the user will supply
  - VALUE: Required for radio and checkboxes
  - CHECKED: For radio buttons and checkboxes
  - SIZE: Specific to each type of field
  - MAXLENGTH: Limit on accepted characters
  - SRC: Image file used as a graphical submit button when TYPE=IMAGE
  - DISABLED unavailable in this context
  - READONLY for text and passwords
- **HTML5 adds several new attributes for validation**
- See [http://www.w3schools.com/tags/tag\\_input.asp](http://www.w3schools.com/tags/tag_input.asp)

## <INPUT> Tag Attributes

- *<input> tag also supports the **Global Attributes**, that can be used with all HTML elements*
  - CLASS: Specifies one or more class names for an element (refers to a class in a style sheet)
  - HIDDEN: Specifies that an element is not yet, or is no longer, relevant
  - And many more. See:  
[https://www.w3schools.com/tags/ref\\_standardattributes.asp](https://www.w3schools.com/tags/ref_standardattributes.asp)
- The <input> tag also supports the **Event Attributes** in HTML.
  - Window event attributes: onload, onunload, etc.
  - Form event attributes: onchange, onfocus, etc.
  - Keyboard / mouse events: onkeydown, onclick, etc.
  - Drag, clipboard, media events: onpause, onplay, etc.
  - And many more. See:  
[https://www.w3schools.com/tags/ref\\_eventattributes.asp](https://www.w3schools.com/tags/ref_eventattributes.asp)

## <INPUT> Element, Type Options (cont'd)

- TYPE: [CHECKBOX | FILE | HIDDEN | IMAGE | PASSWORD |  
                  RADIO | RESET | SUBMIT | TEXT]

[HTML5 adds 13 new input types. See later slides]

- **CHECKBOX**: A single value, on/off; each generates name/value pair

```
<INPUT TYPE=CHECKBOX CHECKED NAME="MARRIED"
VALUE="yes">
```

- **FILE**: Users attach a file to the form contents; a text field holds the file name, and a button permits browsing

```
<INPUT TYPE=FILE NAME="fname">
```

- **HIDDEN**: The field is not rendered, so servers can maintain state information

```
<INPUT TYPE=HIDDEN NAME="BANKACCT" VALUE="A057-23-
789">
```

## <INPUT> Element, Type Options (cont'd)

- **RESET**: Defines a button that users click to reset fields to their initial state

```
<INPUT TYPE=RESET VALUE="CLEAR">
```

- **SUBMIT**: Defines a button that users click to submit the form's contents to the server

```
<INPUT TYPE=SUBMIT VALUE="submit data">
```

- **TEXT**: An input field of a single line where users can enter data

```
<INPUT TYPE=TEXT SIZE=20 NAME="lastname"
VALUE="type your last name">
```

## <INPUT> Element, Type Options (cont'd)

- **IMAGE**: Used for graphical submit buttons

```
<INPUT TYPE=IMAGE SRC="banner.gif" VALUE="gohome">
```

- **PASSWORD**: Just like TYPE=TEXT, but the input is echoed with \*

```
<INPUT TYPE=PASSWORD SIZE=10 NAME="pw">
```

- **RADIO**: Used for attributes that take a single value from a set of alternatives; all buttons have same name and explicit value

```
<INPUT TYPE=RADIO NAME="AGE" VALUE="0-20">
```

```
<INPUT TYPE=RADIO NAME="AGE" VALUE="21-50">
```

```
<INPUT TYPE=RADIO NAME="AGE" VALUE="51-100"
```

```
CHECKED>
```

## <INPUT> Element, Type Options

- **TYPE:** [COLOR | DATE | DATETIME | DATETIME-LOCAL | EMAIL | MONTH | NUMBER | RANGE | SEARCH | TEL | TIME | URL | WEEK]
  - **COLOR:** Used for input fields that should contain a color  
Select color: <INPUT TYPE="COLOR" name="favcolor">
  - **DATE:** Allows the user to select a date  
Birthday: <INPUT TYPE="DATE" NAME="bday">
  - **DATETIME:** Allows the user to select a date and time (with time zone)  
Birthday: <INPUT TYPE="DATETIME" NAME="BDAYTIME">
  - **EMAIL:** Allows the user to enter an e-mail address  
E-Mail: <INPUT TYPE="EMAIL" NAME="email">
  - **MONTH:** Allows the user to select month/year  
Birthday (M/Y) : <INPUT TYPE="MONTH" NAME="bdaymonth">

## <INPUT> Element, Type Options (cont'd)

- **NUMBER**: Used to enter a numeric value

Quantity (1-5) : <INPUT TYPE="NUMBER"  
name="quantity" min="1" max="5">

- **RANGE**: Used to enter a value from a range of numbers

<INPUT TYPE="RANGE" NAME="points">

- **SEARCH**: Used for search fields (behaves like regular TEXT)

Search Google: <INPUT TYPE="SEARCH"  
NAME="GOOGLESEARCH">

- **TEL**: Allows the user to enter a telephone num.

Telephone: <INPUT TYPE="TEL" NAME="ustel"

- **TIME**: Allows the user to select a time

Time: <INPUT TYPE="TIME" NAME="ustime"

## <INPUT> Element, Type Options (cont'd)

- **URL**: Used to enter a URL address

Add Homepage: <INPUT TYPE="URL" name="homepage">

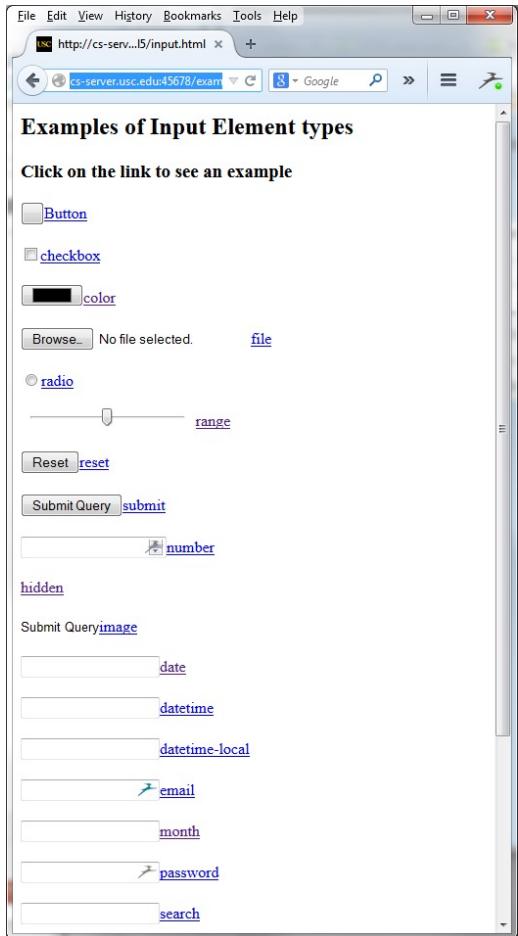
- **WEEK**: Used to select a week and year

Select a week: <INPUT TYPE="WEEK" NAME="week\_YR">

- **DATETIME-LOCAL**: Used select date and time (no time zone)

Birthday: <INPUT TYPE="DATETIME-LOCAL"  
NAME="bday">

# <INPUT> Element, Type Options (cont'd)



Examples of all of the <input> element types, including the most recent in HTML5 provided by w3schools

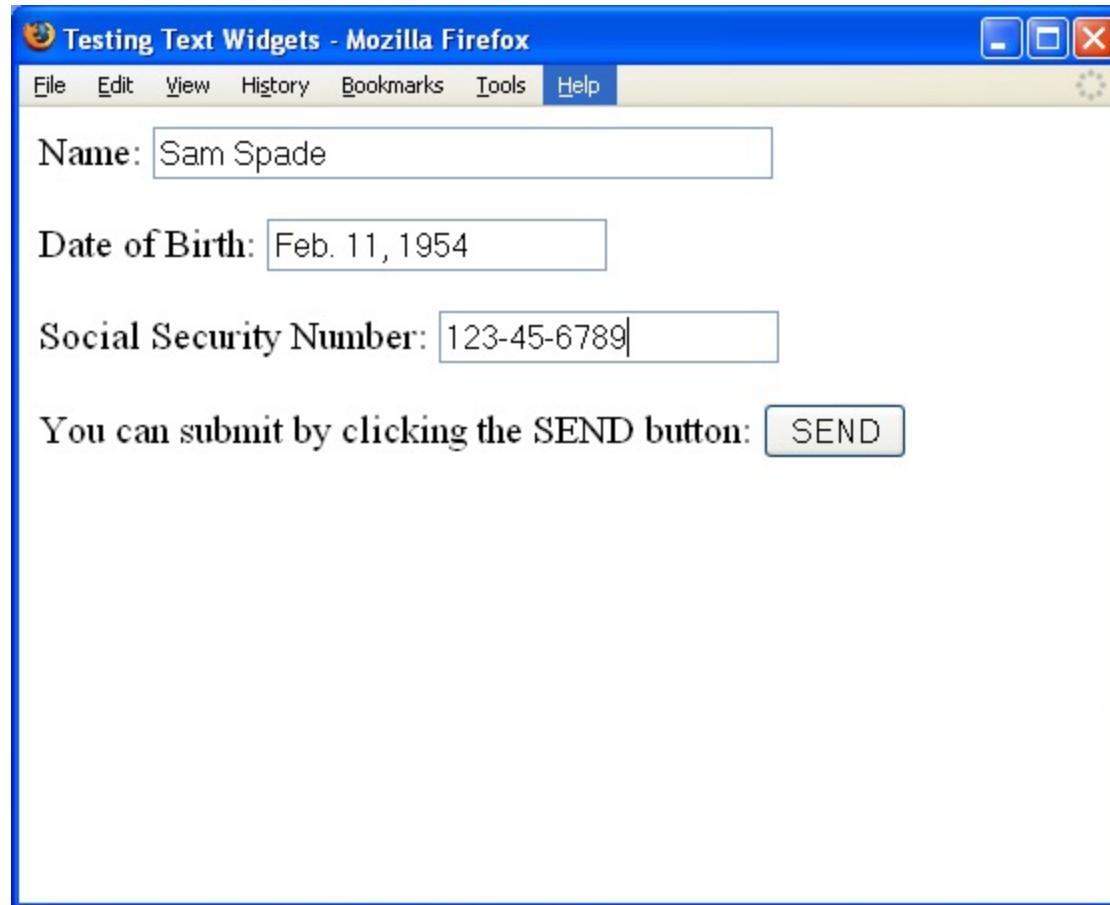
<http://csci571.com/examples/html5/input.html>

## Example of <FORM> With Text Widgets

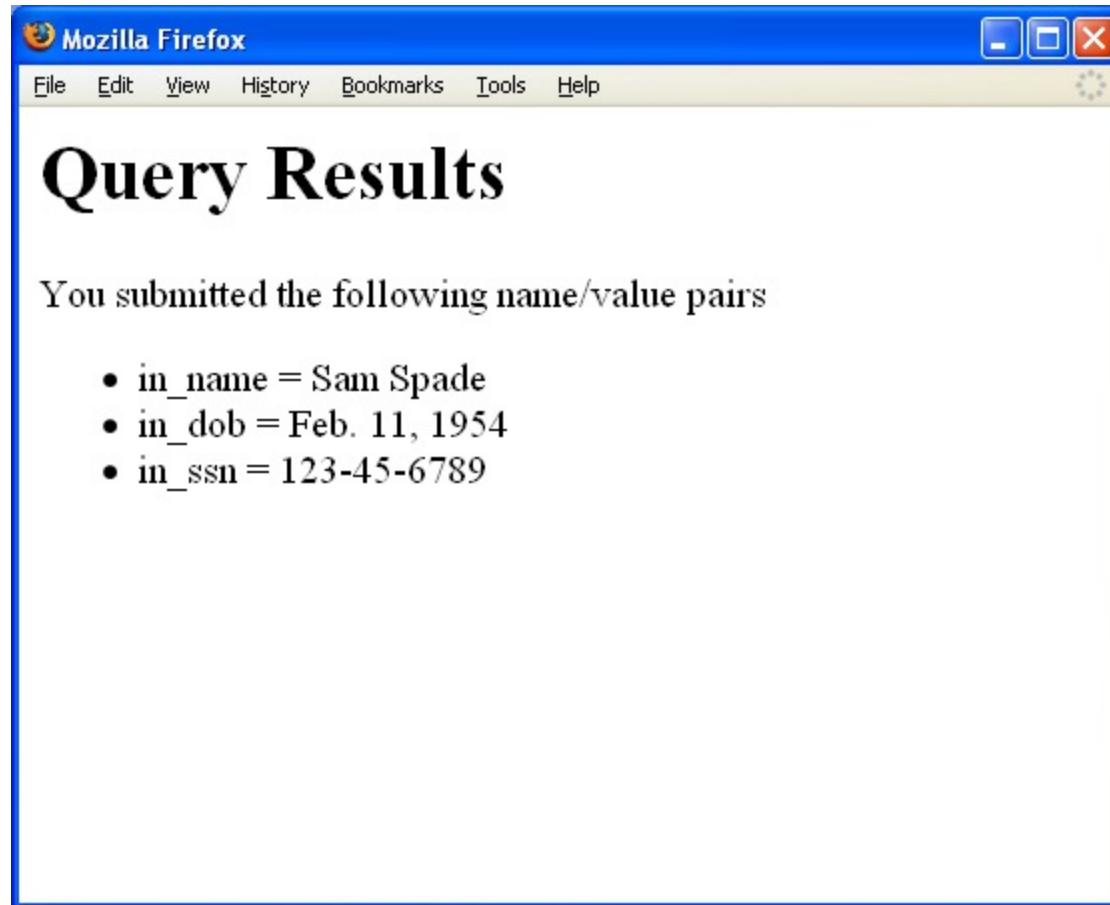
```
<HTML><HEAD><TITLE>Testing Text Widgets</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST"
ACTION="/cgi-bin/post-query">
Name: <INPUT NAME="in_name" TYPE="text" SIZE=40><P>
Date of Birth: <INPUT TYPE="text" NAME="in_dob"><P>
Social Security Number: <INPUT TYPE="text"
NAME="in_ssn"><P>
You can submit by clicking the SEND button:
<INPUT TYPE="submit" VALUE="SEND">
</FORM></BODY></HTML>
```

*Note:post-query is a standard Apache CGI program distributed by web servers and used to check that form elements are being properly sent to the server*

# Browser Output of Text Widgets Example



# Query Results for Text Widget Example



## **Example of <FORM> With Checkboxes**

```
<HTML><HEAD><TITLE>Testing Checkboxes</TITLE></HEAD>
<BODY>
<FORM METHOD="POST" ACTION="/cgi-bin/post-query">
Fill in facts about yourself:<P>
<INPUT TYPE="checkbox" NAME="house" VALUE="yes">own a
house

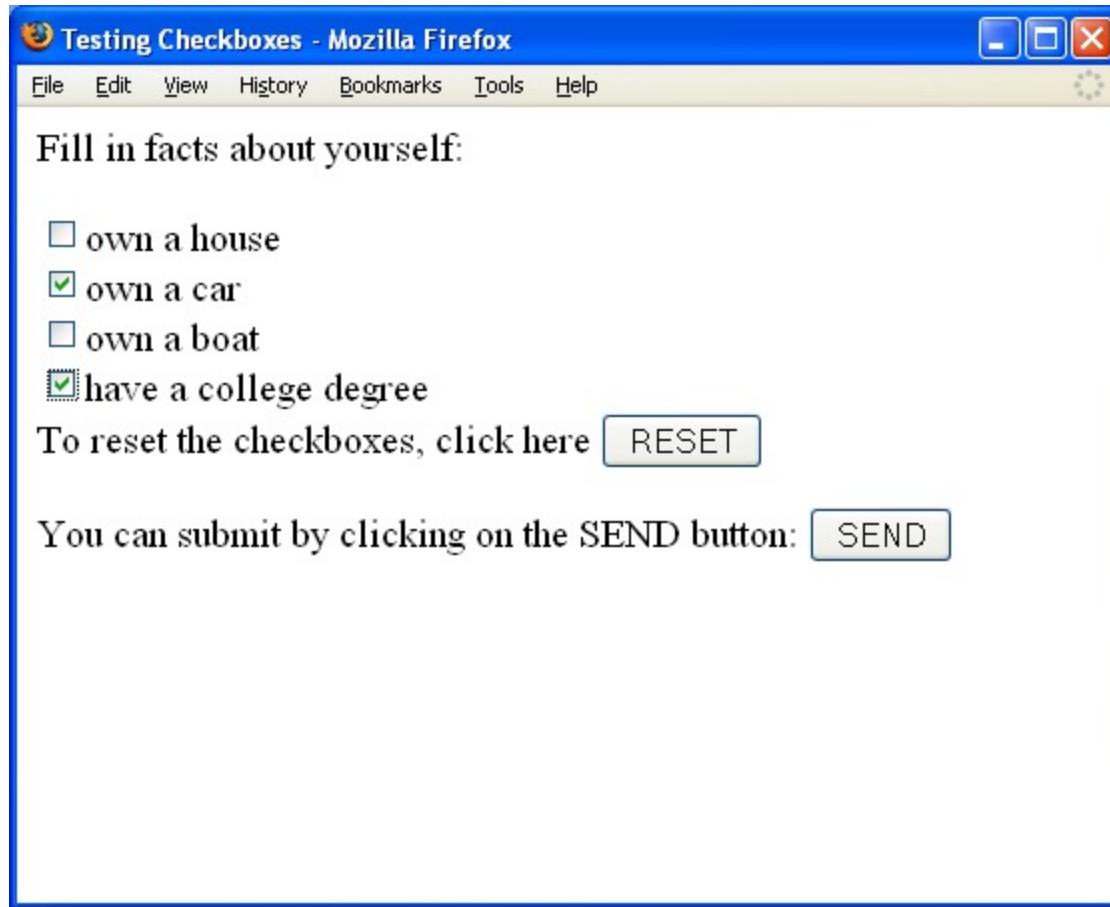
<INPUT TYPE="checkbox" NAME="car" VALUE="yes">own a
car

<INPUT TYPE="checkbox" NAME="boat" VALUE="yes">own a
boat

<INPUT TYPE="checkbox" NAME="degree" VALUE="yes">have a
college degree

To reset the checkboxes, click here
<INPUT TYPE=reset VALUE="RESET"><P>
You can submit by clicking on the SEND button:
<INPUT TYPE=submit VALUE="SEND"><P>
</FORM></BODY></HTML>
```

# Browser Output of Checkbox Example



The screenshot shows a Mozilla Firefox browser window with a blue title bar and a white content area. The title bar says "Testing Checkboxes - Mozilla Firefox". The menu bar includes "File", "Edit", "View", "History", "Bookmarks", "Tools", and "Help". The content area contains the following text and form elements:

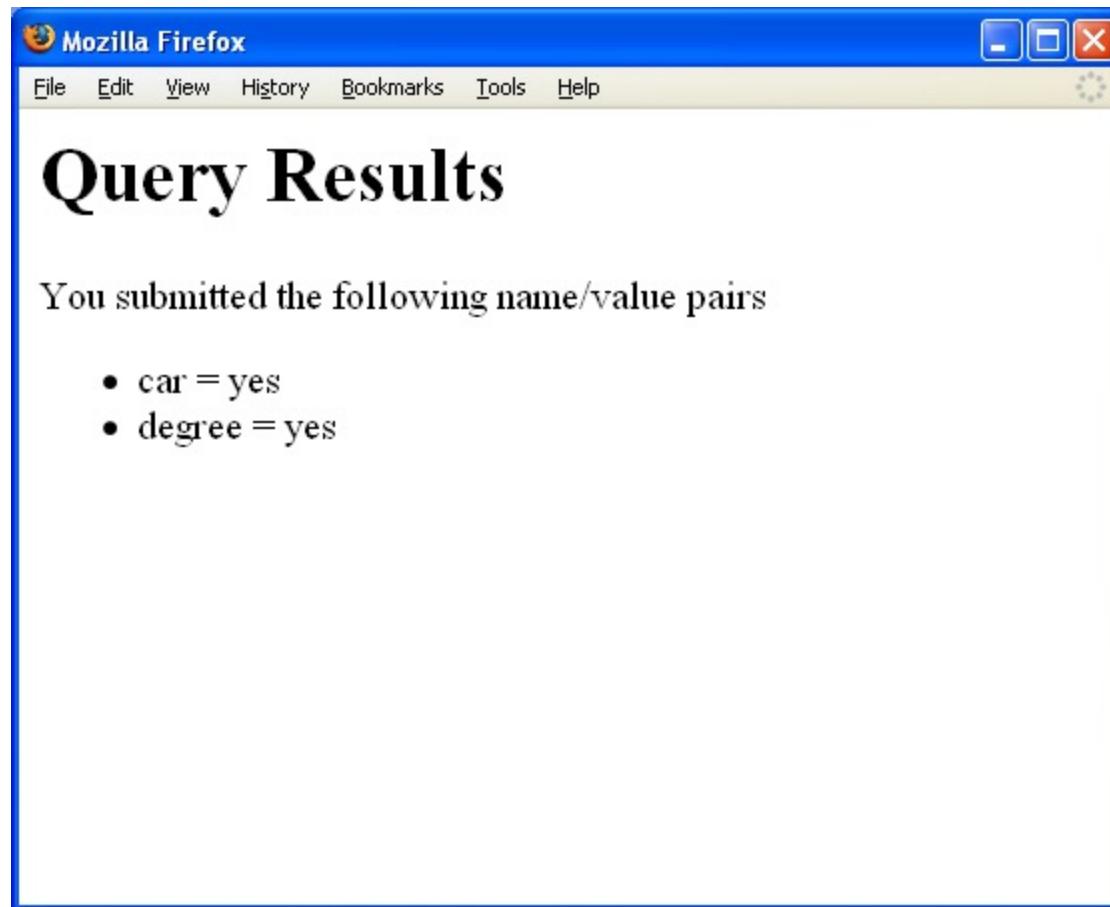
Fill in facts about yourself:

own a house  
 own a car  
 own a boat  
 have a college degree

To reset the checkboxes, click here

You can submit by clicking on the SEND button:

# Query Results of Checkbox Example



## Example of <FORM> With Radio Buttons

```
<HTML><HEAD><TITLE>Testing Radio Buttons</TITLE></HEAD>
<BODY><FORM METHOD="POST" ACTION="/cgi-bin/post-query">
How would you like to pay? Choose one of the following:<P>
<INPUT TYPE="radio" Name="paymethod" VALUE="billme"
CHECKED>Bill me

<INPUT TYPE="radio" Name="paymethod"
VALUE="check">Check

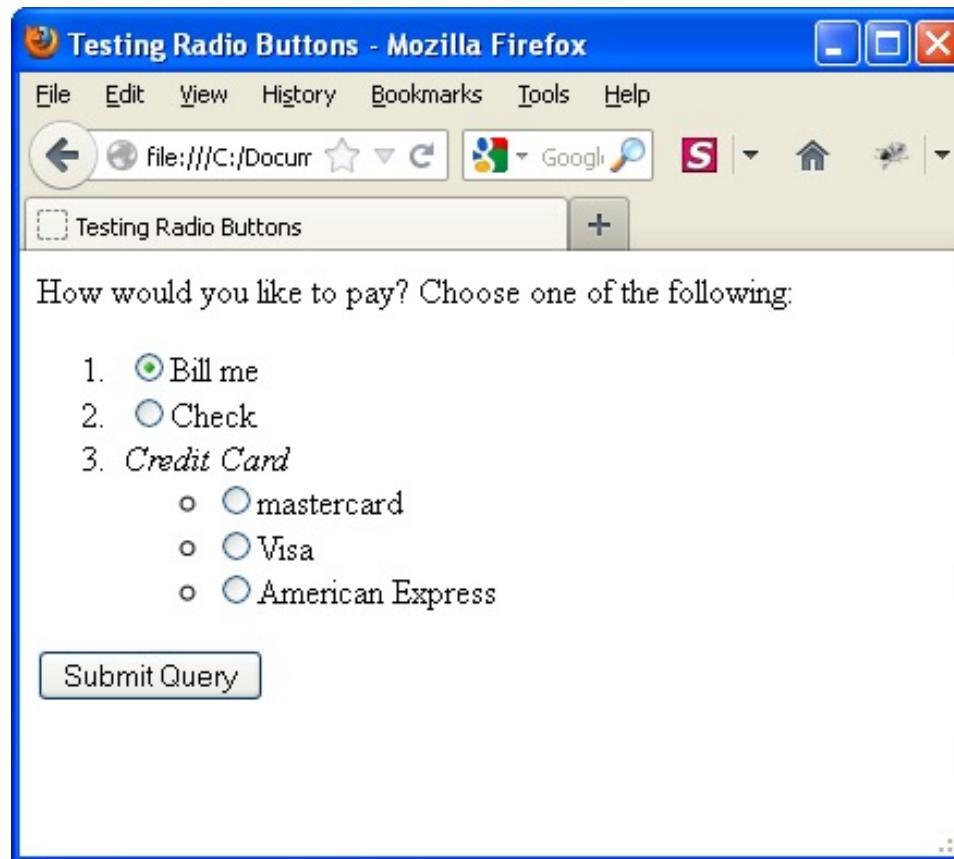
<I> Credit Card </I>
<INPUT TYPE="radio" Name="paymethod"
VALUE="mastercard">mastercard

<INPUT TYPE="radio" Name="paymethod"
VALUE="visa">Visa

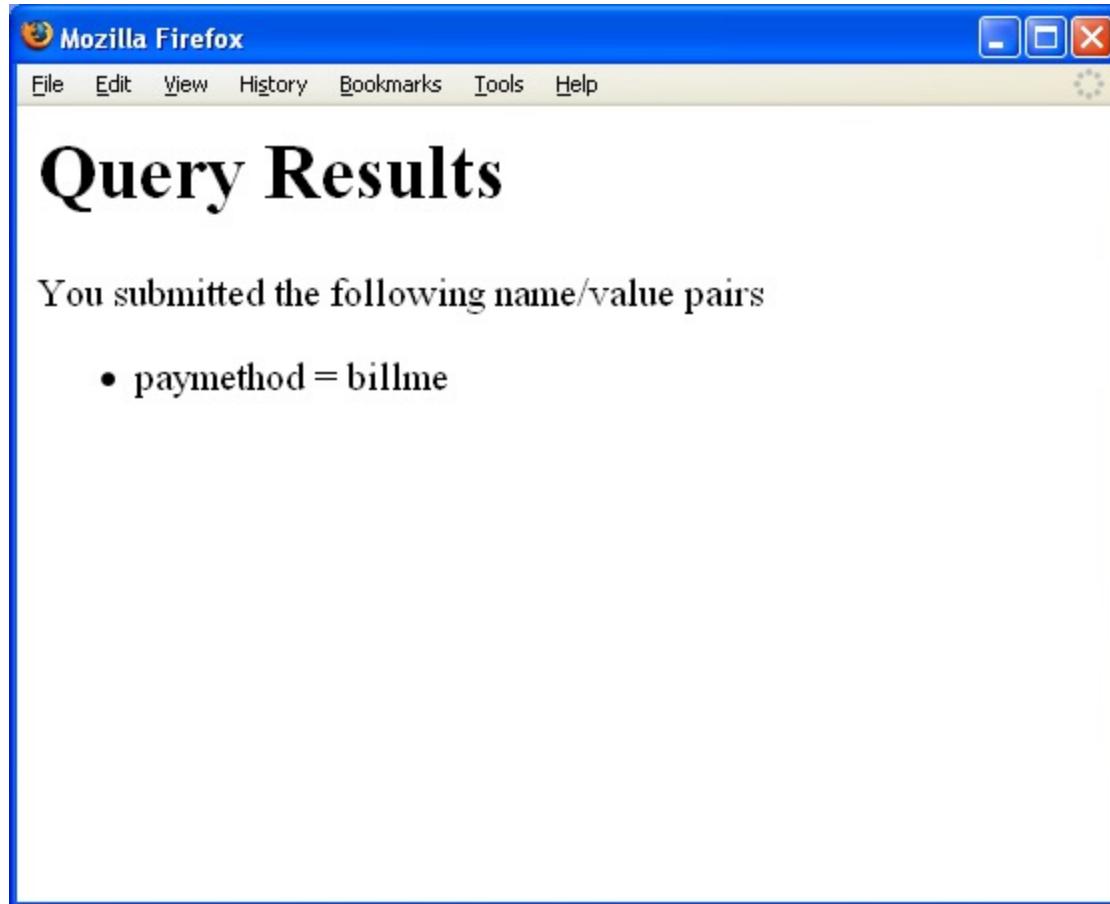
<INPUT TYPE="radio" Name="paymethod"
VALUE="amer">American Express

<INPUT TYPE="submit" VALUE="Submit Query">
</FORM></BODY></HTML>
```

# Browser Output of Radio Buttons



# Query Results for Radio Buttons Example



## **<TEXTAREA> Tag**

- specifies a large rectangular text-entry object with multi-line input and scroll bars
- Attributes:

NAME=name specifies a name for the data entry object to be sent to the server-side script

COLS=num

- Width (in characters) of a text-entry region on the screen
- If user types more than COLS characters, field is scrolled

ROWS=num

- Height (in characters) of a text-entry region on the screen
- If user types more than ROWS lines, field is scrolled

## Example of Multiline Input Areas

```
<HTML><HEAD><TITLE>Form Example with Multiple
Multiline Inputs</TITLE></HEAD> <body>
<form method="POST" action="/cgi-bin/postquery">
<TEXTAREA NAME="largearea" ROWS=10 COLS=30>This is
10 x 30</TEXTAREA>

Here is a 10 x 30 text area.

<TEXTAREA NAME="smallarea" ROWS=2 COLS=20>This is 2
x 20</TEXTAREA>

Here is a 2 x 20 text area.

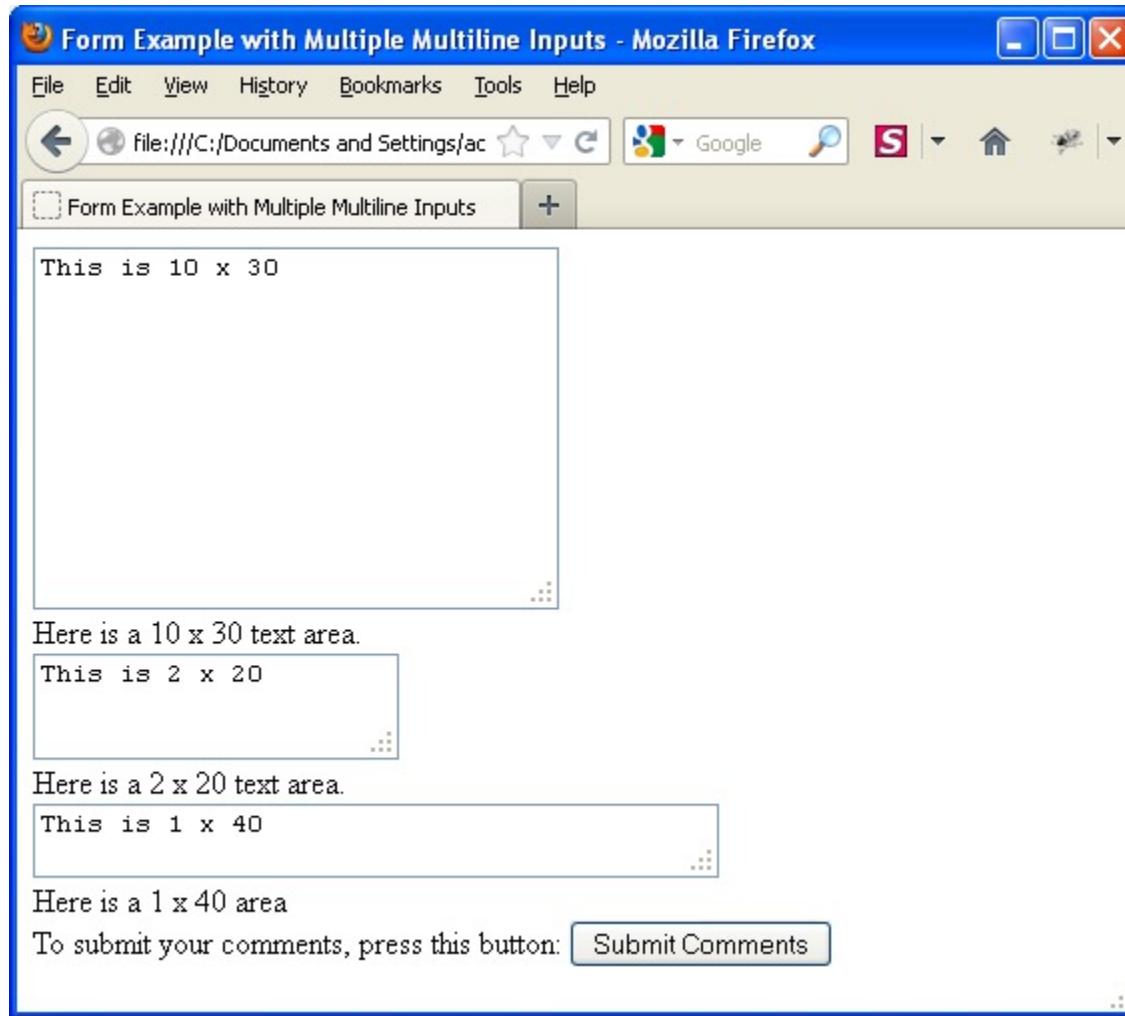
<TEXTAREA NAME="narrowarea" ROWS=1 COLS=40>This is 1
x 40</TEXTAREA>

Here is a 1 x 40 area

To submit your comments, press this button:
<INPUT TYPE="submit" VALUE="Submit Comments">

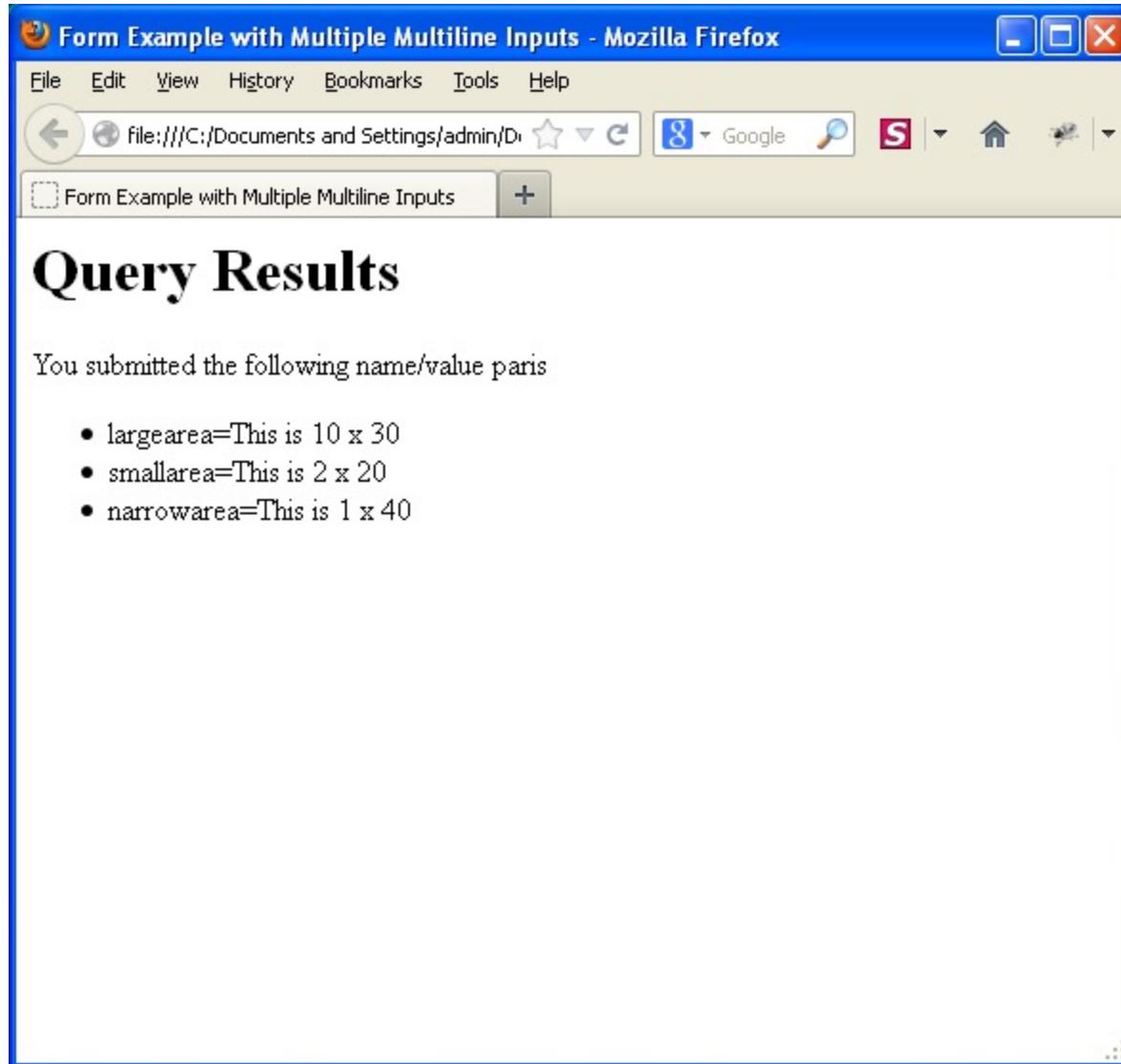
</FORM></BODY></HTML>
```

# Browser Output of Multiline Input Areas



Initial Screen

# Query Results of Textarea Example



## <SELECT> Tag

- Used inside the <FORM> element to specify a selection list object (a list of items or a pop-down menu that the user can select from)
- Attributes:
  - NAME=name
    - Specifies a name for the data entry object to be passed to the server-side script
  - SIZE=num
    - Number of lines of the list to display at a time
    - If SIZE is 1 or unspecified, browser will display as a drop-down list box
    - If SIZE is greater than 1, browser will display as a scrollable list with only SIZE options visible at a time

## <SELECT> Tag Attributes

- MULTIPLE
  - Specifies that multiple list items may be selected (whereas normally only 1 item can be selected)
  - All selected values are sent to server-side script as separate name/value pairs
- HTML5 adds more attributes:
  - **AUTOFOCUS**: drop-down list should automatically get focus
  - **FORM**: defines one or more forms the select fields belongs to
  - **REQUIRED**: user is required to select a value before submitting the form

## <OPTION> Tag

- Used inside the <SELECT> tag to specify the start of a new menu item in the selection list
- Syntax as follows:  
`<OPTION attributes> Text`
- Attributes:
  - SELECTED
    - Menu item is pre-selected in the list
  - VALUE="text"
    - Text specifies the value to be sent to the script if the option is selected
    - By default, the text following the OPTION element is sent
  - DISABLED
    - Specifies a “grayed”, non-selectable item
  - **HTML5 adds the REQUIRED attribute**

## Example of <SELECT>, <OPTION> Tags

```
<HTML><HEAD><TITLE>Forms Example with
Options</TITLE></HEAD><BODY>
<FORM METHOD="POST" ACTION="/cgi-bin/post-query">
Which School would you like to apply to?

<SELECT NAME="school" SIZE=5>
 <OPTION> Letters&Science</OPTION>
 <OPTION SELECTED> Engineering</OPTION>
 <OPTION> Business</OPTION>
<OPTION>Law</OPTION><OPTION> Medicine</OPTION></SELECT>

What semester do you wish to start?

<SELECT NAME="semester">
 <OPTION SELECTED> Fall</OPTION>
 <OPTION> Spring</OPTION>
 <OPTION>Summer</OPTION></SELECT>

To submit your choices, press this button: <INPUT
TYPE="submit" VALUE="Submit Choices">.

To reset the form, press this button: <INPUT TYPE="reset"
VALUE="Reset">.
</FORM></BODY></HTML>
```

# Browser Output of <SELECT>, <OPTION> Example

Forms Example with Options - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Which School would you like to apply to?

Letters&Science

Engineering

Business

Law

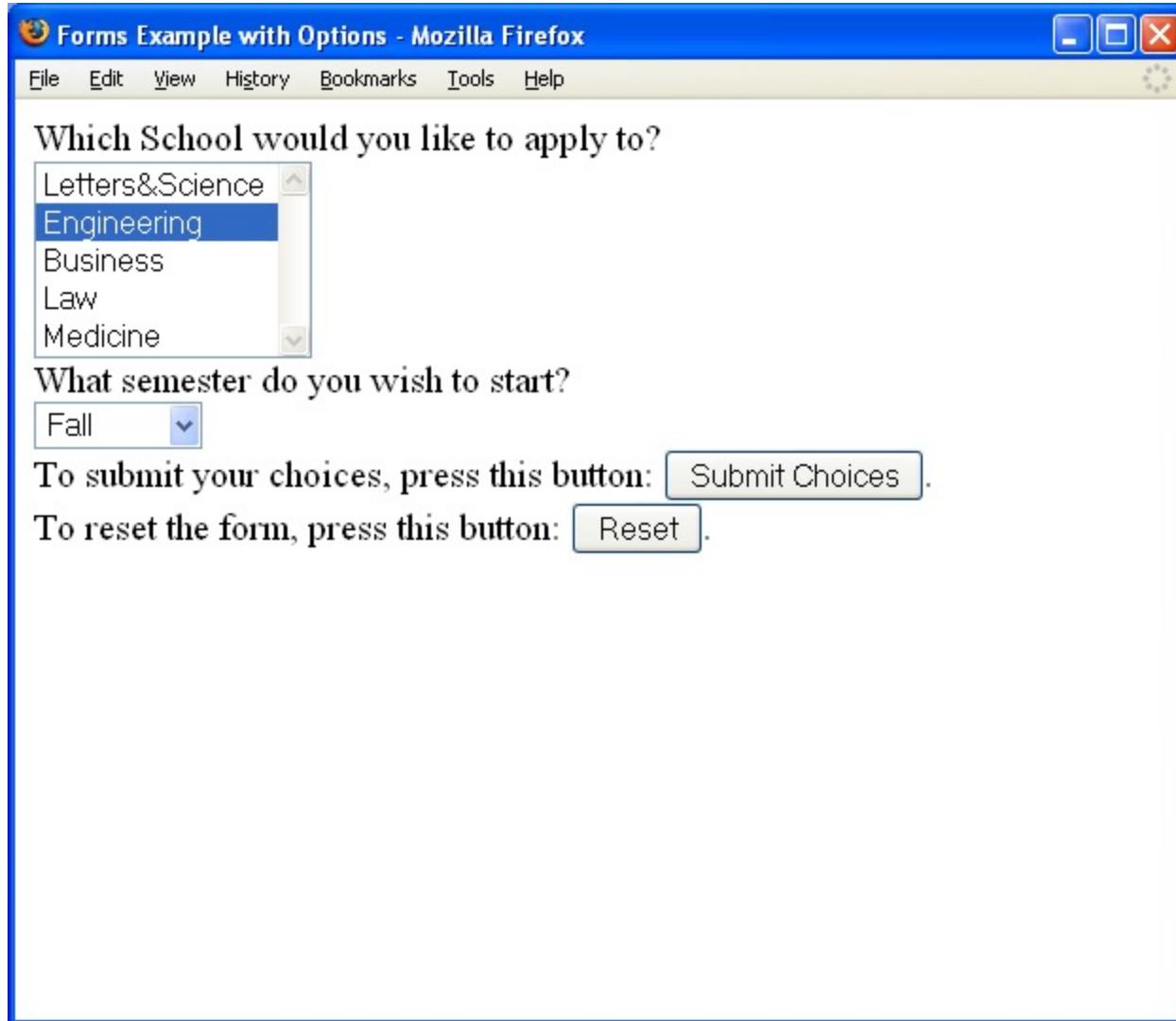
Medicine

What semester do you wish to start?

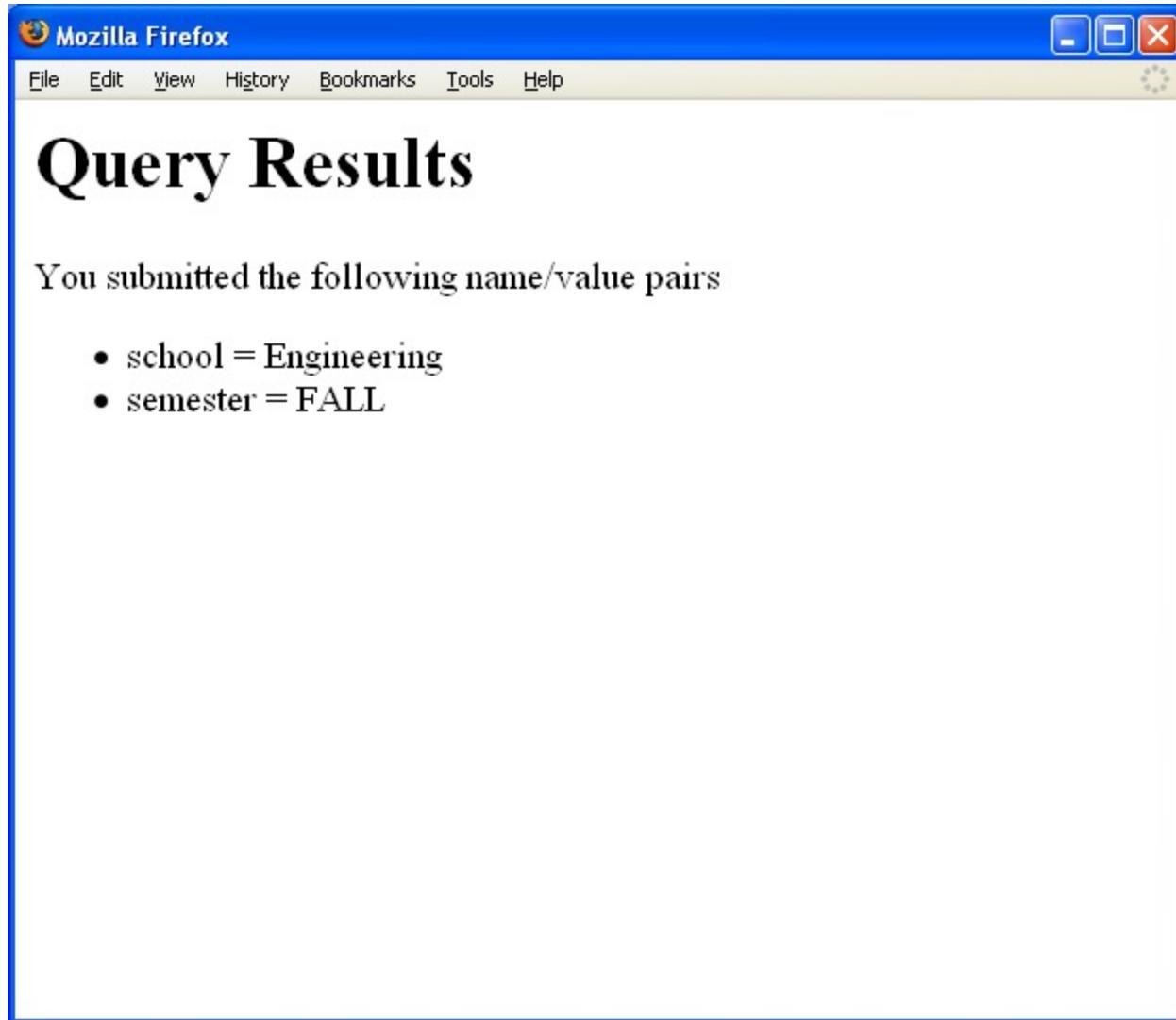
Fall

To submit your choices, press this button: .

To reset the form, press this button: .



# Query Results for <SELECT> Example



## **FIELDSET** – Form Control Group

- The **FIELDSET** element defines a *form control group*.
  - By grouping related form controls, authors can divide a form into smaller, more manageable parts, improving the usability problem that can strike when confronting users with too many form controls.
  - The grouping provided by **FIELDSET** also helps the accessibility of forms to those using aural browsers by allowing these users to more easily orient themselves when filling in a large form.
- The content of a **FIELDSET** element must begin with a **LEGEND** to provide a caption for the group of controls. Following the **LEGEND**, **FIELDSET** may contain any HTML element, including another **FIELDSET**.

# Browser Output

## 3 Fieldsets Grouping form elements

FieldSet Example - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///C:/Documents and Settings/admin/Desktop/test.html Google S

Bookmarks FieldSet Example

Contact Information

Name: Richard Stark

E-mail Address: stark@cs.com

123 Main Street  
Los Angeles, CA  
90069

Mailing Address:

Ordering Information

Please select the product(s) that you wish to order:

[HTML 3.2 Reference](#)  
 [HTML 4.0 Reference](#)  
 [Cascading Style Sheets Guide](#)

Credit Card Information

Visa  MasterCard

Number: 1234 5678 9123 4567

Expiry: 05/2015

Submit order Clear order form

*Run through Tab order:*

Name

E-mail

Mailing Address

HTML 3.2

HTML 4.0

.

.

Etc

To test ACCESSKEY in Chrome use

ALT + ACCESSKEY (I, O, C)

To test ACCESSKEY in Firefox use

ALT + SHIFT + ACCESSKEY

<http://csci571.com/examples/html5/fieldsettest.html>

# Fieldset Example (see next slide)

```
<FORM METHOD=post ACTION="/cgi-bin/order.cgi">

<FIELDSET> <LEGEND ACCESSKEY=I>Contact Information</LEGEND>
<TABLE> <TR> <TD> <LABEL FOR=name ACCESSKEY=N>Name:</LABEL> </TD>
 <TD> <INPUT TYPE=text NAME=name ID=name> </TD> </TR>
<TR> <TD> <LABEL FOR=email ACCESSKEY=E>E-mail Address:</LABEL> </TD>
<TD> <INPUT TYPE=text NAME=email ID=email> </TD> </TR>
<TR> <TD> <LABEL FOR=addr ACCESSKEY=A>Mailing Address:</LABEL> </TD>
<TD> <TEXTAREA NAME=address ID=addr ROWS=4 COLS=40></TEXTAREA> </TD> </TR> </TABLE> </FIELDSET>

<FIELDSET> <LEGEND ACCESSKEY=O>Ordering Information</LEGEND>
<P>Please select the product(s) that you wish to order:</P>
<P> <LABEL ACCESSKEY=3>
<INPUT TYPE=checkbox NAME=products VALUE="HTML 3.2 Reference">
HTML 3.2 Reference </LABEL>
 <LABEL ACCESSKEY=4> <INPUT TYPE=checkbox
NAME=products VALUE="HTML 4.0 Reference">
HTML 4.0 Reference </LABEL>
 <LABEL ACCESSKEY=S> <INPUT TYPE=checkbox
NAME=products VALUE="CSS Guide"> Cascading Style Sheets Guide </LABEL> </P>
</FIELDSET>

<FIELDSET> <LEGEND ACCESSKEY=C>Credit Card Information</LEGEND> <P> <LABEL ACCESSKEY=V> <INPUT TYPE=radio
NAME=card VALUE=visa> Visa </LABEL> <LABEL ACCESSKEY=M>
<INPUT TYPE=radio NAME=card VALUE=mc> MasterCard </LABEL>

<LABEL ACCESSKEY=u> Number: <INPUT TYPE=text NAME=number> </LABEL>

<LABEL ACCESSKEY=E> Expiry: <INPUT TYPE=text NAME=expiry> </LABEL> </P> </FIELDSET>

<P> <INPUT TYPE=submit VALUE="Submit order"> <INPUT TYPE=reset VALUE="Clear order form">
</P> </FORM>
```

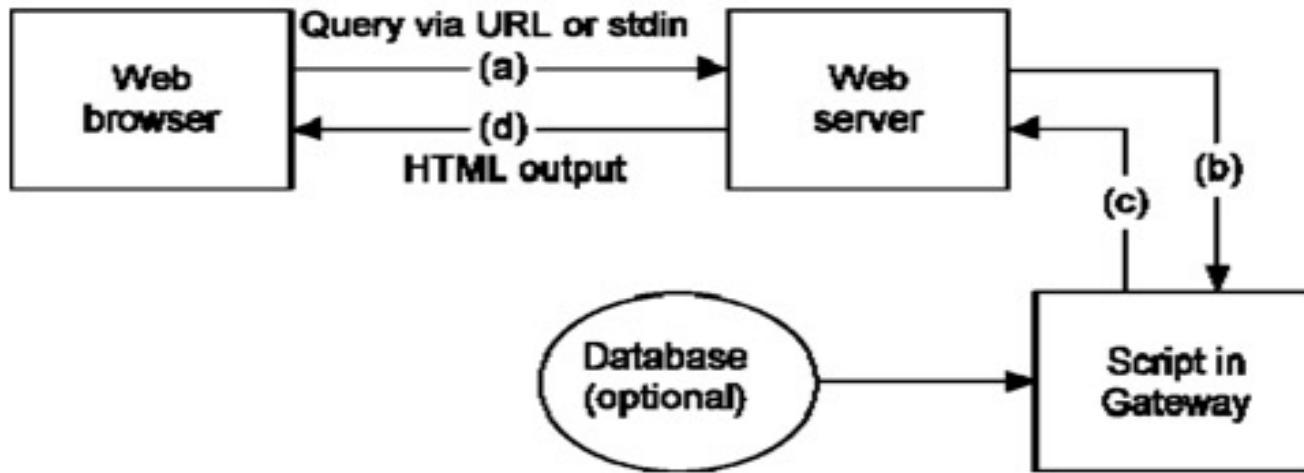
3 *fieldsets* with legends  
*ACCESSKEY* specifies a single character for giving focus

## Purpose of the CGI

- Common Gateway Interface (CGI) is a mechanism by which programs, called *scripts*, can be used to create dynamic Web documents
  - Scripts are placed in a server directory often named cgi-bin
  - Scripts can deliver information that is not directly readable by clients
  - Scripts dynamically convert data from a non-Web source (e.g., DBMS) into a Web-compatible document
- Current version of CGI is 1.1
- The reason for the term “common gateway” is these programs act as gateways between the WWW and any other type of data or service
- See <http://www.w3.org/CGI/>

# Basic Operation

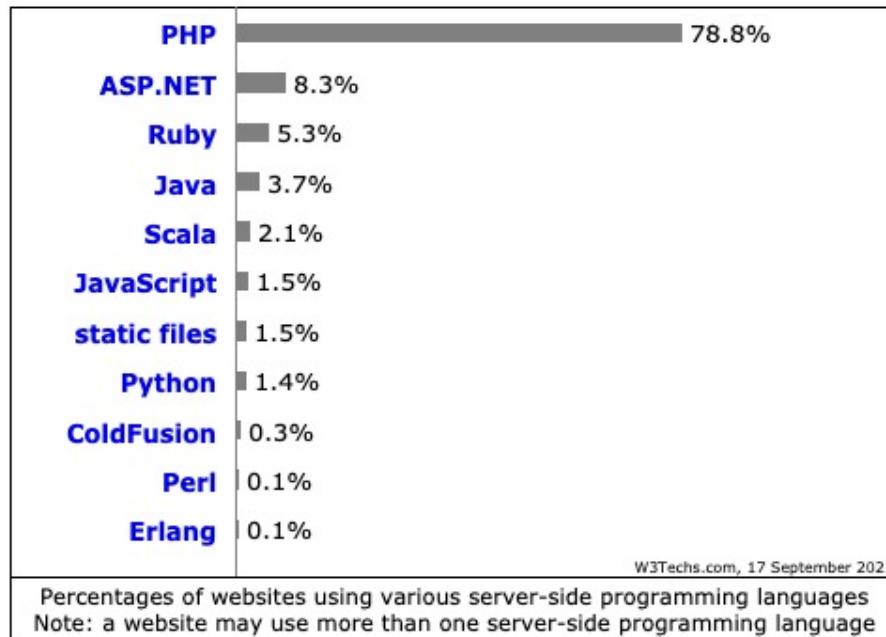
- An executable program that can be run without being directly invoked by users



The browser issues a query, (a), which is sent to the server; the server interprets it and invokes the proper CGI script, passing it the input data, (b); output from the script is returned, (c), via the server, to the browser, (d); output may be HTML, but it may instead be a URL, which is fetched by the server

# Languages to Write Gateway Programs

- Any language that can produce an executable file
- Some typical ones are:
  - Traditional compiled languages such as C/C++
  - Or interpreted languages such as:
    - PHP, JavaScript, Python or Java
- Interpreted languages are often preferred as they are
  - Easy to write and portable, and speed is usually not a factor



[http://w3techs.com/technologies/overview/programming\\_language/all](http://w3techs.com/technologies/overview/programming_language/all)

# **Anchors Are Used to Invoke CGI Scripts**

- A hypertext reference can refer to:

- A remote file

```

```

- An executable script in the cgi-bin directory

```

```

- An executable script with arguments

```

```

- URLs produced by the query “bicycle tours”:

http://search.yahoo.com/bin/search?p=bicycle+tours

http://search.msn.com/results.asp?RS=CHECKED&FORM=M  
SNH&v=1&q=bicycle+tours&zip=90211

# **CGI Script Environment Variables**

- Environment variables
  - are a set of pre-defined dynamic values that can affect a running program
  - they are generally part of the operating environment in which a program runs;
  - UNIX (its variants) and Windows all use these as a means of passing information about the environment of a process
  - CGI environment variables are created by the web server and set immediately before the web server executes a gateway script
  - the CGI script can retrieve the values and use the data they send
  - CGI environment variables are defined in  
<https://datatracker.ietf.org/doc/html/rfc3875>

# CGI Environment Variables

- Can be classified into two major categories:
  - 1. Non-request specific
  - 2. Request specific
- **Non-request-specific** environment variables are the same for all requests:
  - SERVER\_SOFTWARE, the name and version of the information server software answering the request  
e.g., SERVER\_SOFTWARE = Apache/1.3.15
  - SERVER\_NAME, server's hostname, DNS alias, or IP address, e.g., SERVER\_NAME = nunki.usc.edu
  - GATEWAY\_INTERFACE, the revision of the CGI specification with which this server complies
  - SERVER\_PROTOCOL, the name and revision of the information protocol with which this request came in  
e.g., SERVER\_PROTOCOL = HTTP/1.0
  - SERVER\_PORT, the port number to which the request was sent  
e.g., SERVER\_PORT = 8088

# CGI Environment Variables (cont'd)

- **Request-specific** environment variables
  - These variables are set depending on each request
    - REQUEST\_METHOD, the method with which the request was made; e.g., (GET, POST)
    - PATH\_INFO, the extra path information as given by the client; e.g.,  
given `http://nunki.usc.edu:8080/cgi-bin/test.cgi/extra/path`  
then `PATH_INFO = /extra/path`
    - PATH\_TRANSLATED, the PATH\_INFO path translated into an absolute document path on the local system  
`PATH_TRANSLATED = /auto/home-scf-03/csci571/WebServer/apache_1.2.5/htdocs/extra/path`
    - SCRIPT\_NAME, the path and name of the script being accessed as referenced in the URL  
`SCRIPT_NAME = /cgi-bin/test.cgi`
    - QUERY\_STRING, the information that follows the ? in the URL that referenced this script

## **CGI Environment Variables (cont'd)**

- `REMOTE_HOST`, Internet domain name of the host making the request
- `REMOTE_ADDR`, the IP address of the remote host making the request
- `AUTH_TYPE`, the authentication method required to authenticate a user who wants access
- `REMOTE_USER`, username that server and script have authenticated
- `REMOTE_IDENT`, the remote username retrieved by the server using inetd identification (RFC 1413)
- `CONTENT_TYPE`, for queries that have attached information, such as POST method, this is the MIME content type of the data
- `CONTENT_LENGTH`, the length of the content as given by the client

## CGI Environment Variables (cont'd)

- Also, every item of information in an HTTP request header is stored in an environment variable
  - Capitalize the name in the request header field
  - Convert dashes to underscores
  - Add the prefix HTTP\_
- For example:
  - HTTP\_USER\_AGENT contains the request header User\_Agent field data

e.g., HTTP\_USER\_AGENT = Mozilla/4.7 [en]C-DIAL (WinNT; U)
  - HTTP\_ACCEPT contains the request header Accept field, of the form type/subtype
  - HTTP\_REFERER contains the URL of the document that generated this request

# **CGI Script Output**

- The script sends its output to stdout; the server adds appropriate headers and returns this output to the client
- Output from a script to the server could be:
  - A document generated by a script
  - The type of document could be: HTML, plain text, image, video or audio clip, and many other types
  - Instructions to the server for retrieving the desired output elsewhere
  - an error indicator

## **Server Directives**

- The output of scripts begins with a small header consisting of text lines containing server directives
  - This must be followed by a blank line
- Any headers that are not server directives are sent directly back to the client
- Server directives are used by CGI scripts to inform the server about the type of output
- The current CGI specification defines three (3) server directives:
  - Content-type
  - Location
  - Status

## Server Directives (cont'd)

- 1. **Content-type: type/subtype**

- The MIME type of the document being returned
  - For example,

content-type: text/html (HTML document)

content-type: text/plain (plain-text document)

- 2. **Location**

- Alerts the server that the script is returning a reference to a document, not an actual document
  - If the argument is a URL, the server will issue a redirect to the client; for example,

location: http://www.ncsa.uiuc.edu/

- If the argument is a path, the document specified will be retrieved by the server, starting at the document root; for example,

location: /path/doc.txt

## **Server Directives (cont'd)**

- 3. **Status**
  - This is used to give the server an HTTP/1.1 status line to send to the client
  - The format is nnn xxxx:
    - nnn is the three-digit status code
    - xxxx is the informative message
- E.g., 403 Forbidden

# Things to Check Before Running CGI Scripts

- The following need to be readable and executable by the server
  - CGI scripts
  - Other programs that the scripts call
  - The directory in which the scripts reside
- In UNIX, check the read/write/execute permissions of the files and directories
- In Windows, check the web server settings of the script directories

# show\_vars.php

- PHP is a language with built-in ability to access environment variables
- show\_vars.php is a program that prints environment variables
- The code is available at:  
[http://csci571.com/examples/php/show\\_vars.php](http://csci571.com/examples/php/show_vars.php)
- Below is some sample output

```
Show $_SERVER, $_POST, $_GET Variables
www-scf.usc.edu/~csci571/Special/php_ex/show_vars.php
Show $_SERVER, $_POST, $_GET...
```

\$_SERVER	\$_POST	\$_GET
DOCUMENT_ROOT: /var/local/www/htdocs	empty	empty
HTTP_ACCEPT: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8		
HTTP_ACCEPT_ENCODING: gzip, deflate		
HTTP_ACCEPT_LANGUAGE: en-US,en;q=0.5		
HTTP_CACHE_CONTROL: max-stale=0		
HTTP_CONNECTION: Keep-Alive		
HTTP_COOKIE: soc_textsize=small		
HTTP_DNT: 1		
HTTP_HOST: www-scf.usc.edu		
HTTP_PRAGMA: no-cache		
HTTP_USER_AGENT: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:18.0) Gecko/20100101 Firefox/18.0		
HTTP_X_BLUECOAT_VIA: 88e360ea4520a546		
PATH: /sbin:/user/bin:/user/bin:/user/sbin		
REMOTE_ADDR: 159.83.168.252		
REMOTE_PORT: 19753		
SCRIPT_FILENAME: /home/scf-22/csci571/public_html/Special/php_ex/show_vars.php		
SCRIPT_URI: http://www-scf.usc.edu/~csci571/Special/php_ex/show_vars.php		
SCRIPT_URL: /csci571/Special/php_ex/show_vars.php		
SERVER_ADDR: 68.181.201.23		
SERVER_ADMIN: webadmin@usc.edu		
SERVER_NAME: www-scf.usc.edu		
SERVER_PORT: 80		
SERVER_SIGNATURE:		
Apache/1.3.31 Server at www-scf.usc.edu		
Port 80		
SERVER_SOFTWARE: Apache/1.3.31 (Unix)		
PHP/4.3.9 mod_layout/3.0.3		
UNIQUE_ID: URFtKslyRcADdeA6c		
GATEWAY_INTERFACE: CGI/1.1		
SERVER_PROTOCOL: HTTP/1.1		
REQUEST_METHOD: GET		
QUERY_STRING:		
REQUEST_URI: /~csci571/Special/php_ex/show_vars.php		

```
Show $_SERVER, $_POST, $_GET Variables
www-scf.usc.edu/~csci571/Special/php_ex/show_vars.php
Show $_SERVER, $_POST, $_GET...
```

REQUEST_URI: /~csci571/Special/php_ex/show_vars.php	SCRIPT_NAME: /~csci571/Special/php_ex/show_vars.php	PATH_TRANSLATED: /home/scf-22/csci571/public_html/Special/php_ex/show_vars.php	PHP_SELF: /~csci571/Special/php_ex/show_vars.php	argv:
				empty
				argc: 0

• The \$\_SERVER array is initialized by the server, and contains special parameters such as headers, server version.

• The \$\_POST array is set when a form of method=POST has an action to this page and is submitted.

- A form with method = POST

First Name:

Last Name:

• The \$\_GET array is set when a form of method=GET has an action to this page and is submitted, or a URL of this page has a query.

- An example of a URL with a GET query
- A form with method = GET

First Name:

Last Name:

### Handling Arrays

PHP also accepts multidimensional arrays as form input. To create an array, add brackets to end of the name variable. (e.g. time[]). For example, if there is an input named time[], it would be inputted into the time[] array in PHP.

Time 1:

Time 2:

Time 3:

### Code

## **show\_vars.php – output tabs & arrays**

```
<!doctype html><html>
<head><title>Show $_SERVER, $_POST, $_GET Variables</title></head>
<body>
<?php
 function print_tabs($tabs) {
 for($i = 0; $i < $tabs; $i++) {
 echo " &nbsp&nbsp&nbsp&nbsp";
 }
 }

 function print_array($arr, $tabs = 0) {
 if(!empty($arr)) {
 foreach($arr as $k=> $v) {
 print_tabs($tabs);
 echo "" . $k . ": " . $v .
"
";
 if(is_array($v)) {
 print_array($v, $tabs+1);
 }
 }
 } else {
 echo "empty
";
 }
 }
?>
```

## **show\_vars.php - \$\_SERVER, \$\_POST, \$\_GET**

```
<table>
 <tr><th width="33%">$_SERVER</th><th
width="34%">$_POST</th><th
width="33%">$_GET</th></tr>
 <tr>
<td valign="top">
<?php print_array($_SERVER); ?>
</td>
<td valign="top">
<?php print_array($_POST); ?>
</td>
<td valign="top">
<?php print_array($_GET); ?>
</td>
 </tr>
</table>
```

## **show\_vars.php – POST**

```

$_SERVER array is initialized by the server, and contains
special parameters such as headers, server version.
The $_POST array is set when a form of method=POST has an
action to this page and is submitted.

A form with method = POST
 <div>
 <form method="POST" action="">
 <p>
 <label for="fname">First Name</label>
 <input type="text" value="" name="fname"></p>
 <p>
 <label for="lname">Last Name</label>
 <input type="text" value="" name="lname"></p>
 <p>
 <input type="submit" value="Submit" name="submit"></p>
 </form></div>
```

## Demo session

- Firefox developer Tools
  - Built-in, no download required
  - Similar tools as Firebug, but sometimes named differently
  - Provides additional tools not available in Firebug (now retired)
  - Invoke at “burger” -> More Tools -> Web Developer Tools

# **Lecture**

## **The Web Server**

# Outline

- Available Web Servers
- Server Features
  - Document Root
  - Authentication
  - Proxy Servers
  - Caching
  - CGI Scripting
  - Application Program Interface
- Configuring a Server
- Analyzing a Server's Performance
- Server Log Files

# What Does the WWW Server Do?

- Enables browser requests
- Mainly provides
  - Support for retrieving hypertext documents
  - Manages access to the Web site
  - Provides several mechanisms for executing server-side scripts
    - Common Gateway Interface (CGI)
    - Application Program Interface (API)
    - Direct Module Interfaces (SAPI)
  - provides log files and usage statistics

# **What to Look for in a Web Server**

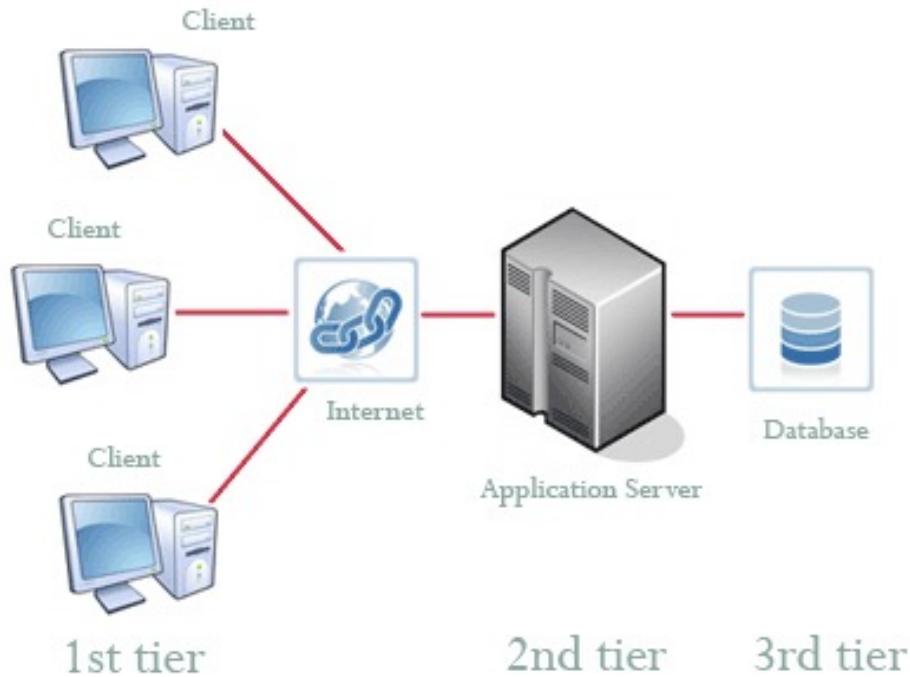
- Main features
  - platform they run on;
  - complete support for HTTP 1.1 / HTTP 2
  - Multithreading, load balancing
- Security features
  - ability to provide IP address restriction
  - ability to provide domain name restriction
  - Support for secure transactions: SSL
  - Ability to act as a proxy server

# How Servers Handle Multiple Requests

- For each request, a complete copy of the server is made and executed
  - Initially a parent process waits for requests; for each new request a child process is spawned
- Or a single server program handles many requests simultaneously (multithreaded)
  - the server must keep track of all requests and switch between them as needed
  - writing multithreaded programs is easier in a language that supports multiple threads, e.g., Java, C#, Swift, Kotlin, ...

# Application Web Server

- An **application server** is software that typically interfaces one or more databases to convey processed data to and from a user interface such as a web browser
- It performs **business logic**



- An application server acts as a set of components accessible through an API
- For web applications, these components are usually performed in the same machine where the web server is running, and their main job is to support the construction of dynamic pages.
- For example, Apache Tomcat is a popular, light-weight application container, but not a full application server as it doesn't provide the services specified in the J2EE specification. The web modules include servlets. Business logic resides in Enterprise JavaBeans (EJB).

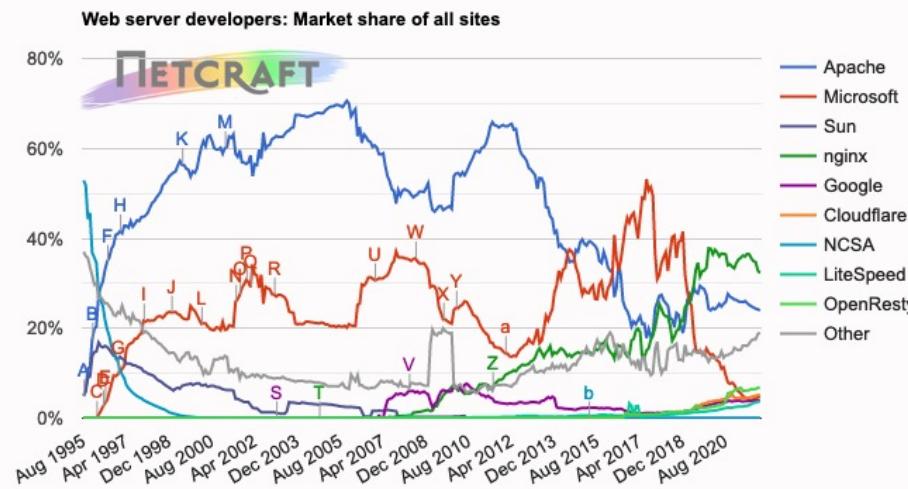
# Some Available Web & Application Servers

Publisher	Product	Platform	URL
<b>Web Servers</b>			
<i>apache</i>	<b>apache</b> (2.4)	Win32/UNIX	<a href="http://httpd.apache.org/">http://httpd.apache.org/</a>
<i>Microsoft</i>	<b>IIS</b> 7-10.0	Win32	<a href="http://www.iis.net">http://www.iis.net</a>
<i>NGINX</i>	<b>nginx</b> (open source)	Linux, BSD, Win32	<a href="http://nginx.org">http://nginx.org</a>
<i>NGINX (F5)</i>	<b>NGINX Plus</b>	Ubuntu, AWS GCP, Azure	<a href="http://www.nginx.com">http://www.nginx.com</a>
<b>Application Servers</b>			
<i>Oracle</i>	<b>GlassFish</b> 4.1	Solaris/Win32/Linux <a href="https://javaee.github.io/glassfish/">https://javaee.github.io/glassfish/</a> Reference Implementation)	(now Open-Source Java EE
IBM	<b>WebSphere</b>	Win32/UNIX/Linux <a href="https://www.ibm.com/cloud/websphere-application-server">https://www.ibm.com/cloud/websphere-application-server</a> <a href="https://www.ibm.com/us-en/marketplace/java-ee-runtime">https://www.ibm.com/us-en/marketplace/java-ee-runtime</a>	
<i>Oracle</i>	<b>WebLogic</b> Server	Win32/UNIX/Linux <a href="https://www.oracle.com/java/weblogic/">https://www.oracle.com/java/weblogic/</a>	

For a comparison of web servers see <http://www.serverwatch.com/>, click on Server Comparison Tool

# Web Server Usage

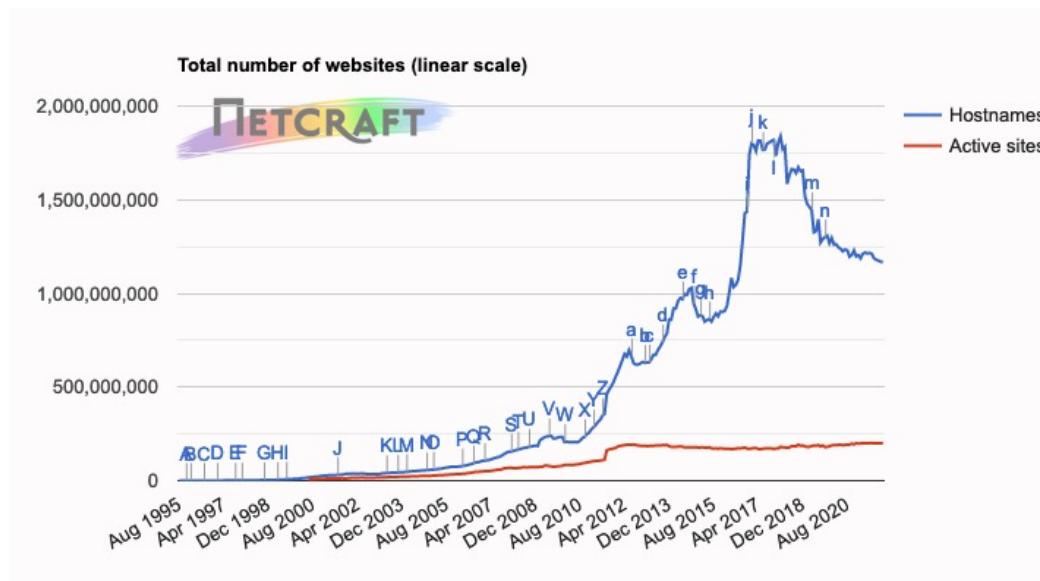
- Netcraft has identified more than 1,167,715,133 sites (1/2022)
  - Apache no longer dominates Internet servers
  - NGINX on top at 32%, Apache at #2 at 24%, OpenResty at 7%
- Statistics on intranets are difficult to determine
  - <http://news.netcraft.com/archives/category/web-server-survey/>



Developer	December 2021	Percent	January 2022	Percent	Change
nginx	384,347,394	32.88%	377,019,054	32.29%	-0.60
Apache	283,409,491	24.25%	279,709,815	23.95%	-0.29
OpenResty	78,902,138	6.75%	80,238,470	6.87%	0.12
Cloudflare	59,904,450	5.13%	60,881,028	5.21%	0.09

# Web Server Survey Takeaways

- From Netcraft **January 2022** and previous surveys we can make these observations
  - Nginx and Apache** have each achieved their widespread popularity in the server market due to their general availability, but many large companies in the web industry choose to roll out their own solutions. **Microsoft disappeared**.
  - Google's custom server software ("GSE")** operates on over 37 million hostnames and 2.0 million domains. Google open-sourced the **Google Servlet Engine**, at the end of 2008, but the software has not received any official public updates in 11 years.
  - Taobao**, China's largest online marketplace and part of Alibaba Group, developed their own **fork of nginx**. The fork, known as **Tengine**, was released back to the community as open source, leading to wider adoption. Tengine now runs on 56.8 million sites, making it the fourth most popular web server by this metric, despite it having only 1.2 million domains.
  - OpenResty**, based on nginx 1.21.4, a scalable Web Platform by Extending NGINX with Lua.



# **Web Server Features**

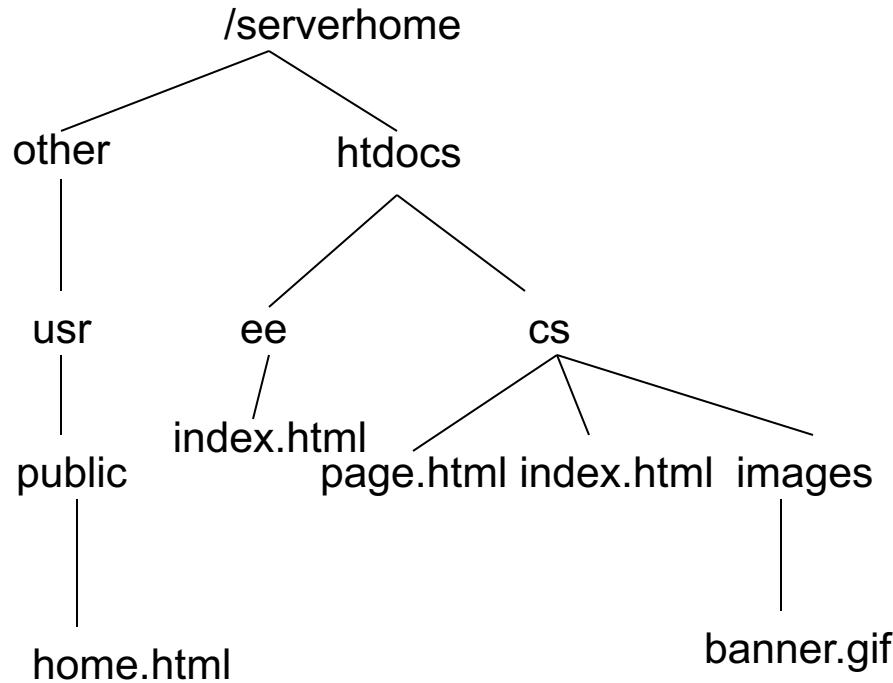
- Document Root
- Authentication
- Proxy Servers
- Caching
- CGI Scripting
- Application Program Interface

## **Web Server Features - Document Root**

- The Web server has access to a tree of files that it can deliver
  - the files are created by content providers
  - files may contain text, images, sound, video, other programs, etc.
- the document tree is organized by the web site administrator
- The root of the document tree is given to the web server when it starts

# Managing the Document Tree

http://domain/cs is mapped to /serverhome/htdocs/cs/index.html



The document root is /serverhome/htdocs

index.html

```
Click here

Display Image

Click Here


```



*The last link is an error*

# Web Server Features (1)

## Virtually Hosted Document Roots

- Hosting multiple web sites by the same web server
- It uses the host name or IP address to distinguish the document roots, e.g.

GET /index.html HTTP/1.0

**Host:** www.hardware.com

might return documents from /htdocs/hardware, while

GET /index.html HTTP/1.0

**Host:** www.antiques.com

might return documents from /htdocs/antiques

- This might be configured in, say apache by writing

<VirtualHost www.hardware.com>

<VirtualHost www.antiques.com>

ServerName www.hardware.com

ServerName www.antiques.com

DocumentRoot /htdocs/hardware

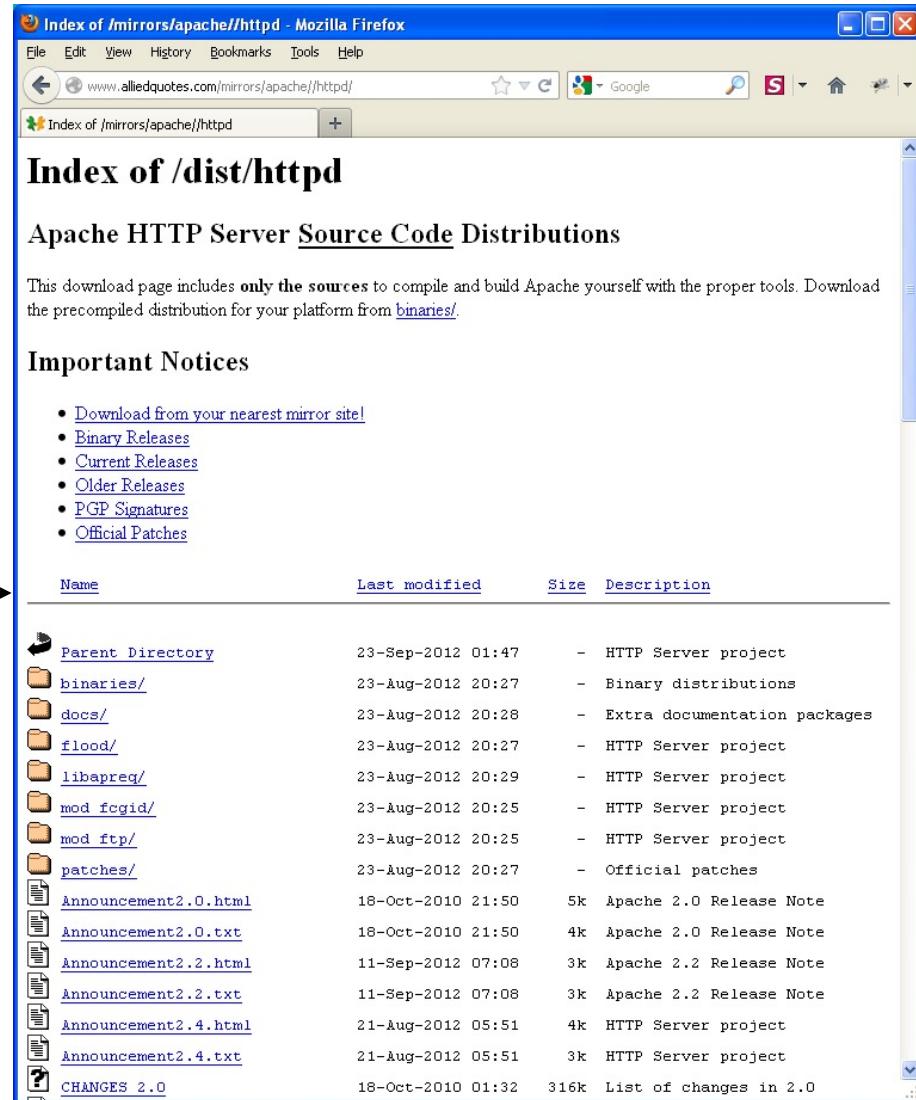
DocumentRoot /htdocs/antiques

</VirtualHost>

</VirtualHost>

# Web Server Features (2) - Directory Listing

- When the URL path resolves to a directory, not to a file, a web server can be configured to return more than just a default file; to the right you see the server returning a list of files in the \_\_\_\_\_ directory including special icons for files and folders
- Important Note:**  
Turn off this feature! Remove "Indexes" from **Options Indexes** in httpd.conf



## Web Server Features (3)

### Basic User Authentication

- Basic authentication is supported by all HTTP servers
  - The server administrator creates secure directories accessible via password files maintained by the server
  - Client requests a resource from a secure directory; e.g., **GET /secure/test.html HTTP/1.0**
  - Server responds with authentication request to the client; e.g., **HTTP/1.0 401 Unauthorized**
  - Browser requests username and password, scrambles them, and retries the request

**GET /secure/test.html HTTP/1.0**

**Authorization: Basic 0<V%L:EB.G-E8W) )\$J**

- Server decodes name and password and checks it against its password file

# **Web Server Features (4)**

## **Creating Server-Side Applications**

- Web Servers offer several mechanisms
  - Application Programming Interface (API)
  - Common Gateway Interface (CGI)
  - J2EE / .NET interfaces
  - Chrome V8 JavaScript Engine (Google)
    - Node.js
  - Direct Module Interfaces (PHP)
    - Apache offers a PHP module
- Microsoft IIS server-side supports:
  - CGI applications
  - API applications compiled as DLL
  - Active Server Pages (written in VBScript)
  - ASP.NET applications (written in C++, VB.NET, C#, or HTML/CSS/JavaScript)

# Configuring a Server

- No matter which operating system or server, you will need to define the
  1. location of the server (server root)
  2. location of documents to deliver (document root)
  3. location of CGI scripts or server-side components to execute
- You may also wish to define
  - Access restrictions
  - Fancy indexing
  - Sys admin e-mail
  - Other features

# Port Numbers

- A typical computer will have thousands of ports
- System ports: 0 - 1023, are reserved for certain functions, typically assigned by IANA (Internet Assigned Number Authority), <http://www.iana.org/>
- Well-known TCP / UDP ports:

[http://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers#Well-known\\_ports](http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers#Well-known_ports)

- programs running on these ports must be started under root

- TCP/UDP ports used by Apple products:

<http://support.apple.com/kb/HT6175>

- Dynamic ports 49152 - 65535, ports that are freely available (check OS X possible conflicts)

# Some Well-Known Port Numbers

<b>Port</b>	<b>Protocol</b>	<b>Purpose</b>
20	ftp	ftp data connection
21	ftp	ftp control connection
23	telnet	telnet session
25	SMTP	simple mail transfer protocol
53	DNS	domain name service
70	gopher	gopher protocol
79	finger	finger protocol
80	http	hypertext transfer protocol
88	kerberos	kerberos authentication protocol
110	pop3	post office protocol
113	ident	remote identity service
119	nntp	network news transfer protocol
143	IMAP	email access
161	snmp	simple network management proto
162	snmp	snmp traps
194	irc	internet relay chat
280	http mgmt	http management
389	ldap	lightweight directory access pr

## Standard Port Numbers (cont'd)

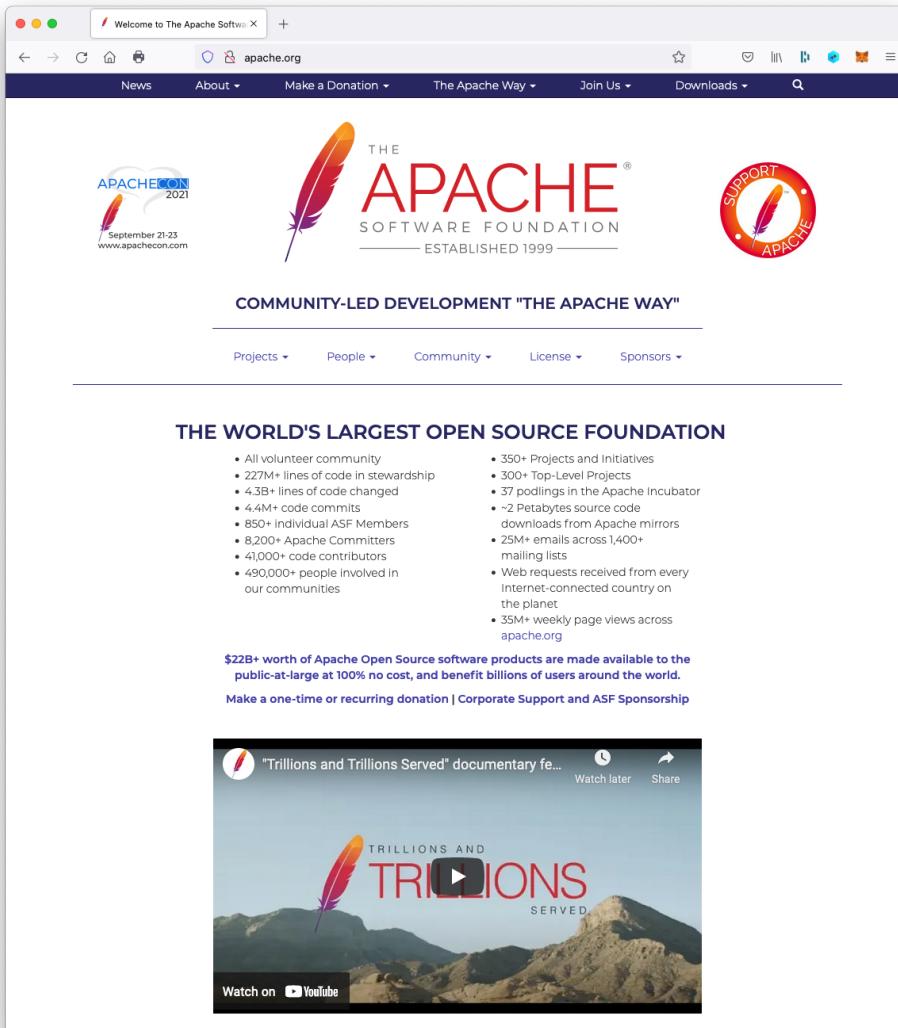
<b>Port</b>	<b>Protocol</b>	<b>Purpose</b>
427	srvloc server	location protocol
443	https	secure http (SSL)
465	smt�ps	secure email (SSL)
535	iiop	internet inter-orb protocol
551	cybercash	secure money transaction
563	snews	secure news (SSL)
614	ssl shell	ssl shell
636	ldaps	secure ldap (SSL)
989	ftps	secure ftp data connection
990	ftps	secure ftp control connection
992	telnets secure	telnet (SSL)
993	imaps	secure imap (SSL)
994	ircs	secure irc (SSL)
995	pop3s	secure pop3 (ssl)
1080	socks	socks protocol, circuit level

# **The Apache Web Server**

# **Apache Web Server**

- Apache is based on code and ideas developed at NCSA in httpd version 1.3 (early 1995)
- The name “Apache” comes from the fact that it is A set of **PATCHES** to **NCSA’s httpd**
- Apache is free and is downloadable from <http://httpd.apache.org>
- Apache included in every version of macOS:  
<https://wpbeaches.com/install-apache-mysql-php-on-macos-11-big-sur-and-earlier-macos/>  
<https://discussions.apple.com/docs/DOC-250004361>
- Using Apache in Windows 10/11  
<https://www.thewindowsclub.com/install-apache-on-windows-10>
- **Apache HTTP Server version 2.4.52 is current.**  
Version **2.4.43** or newer is required in order to operate a **TLS 1.3** web server with **OpenSSL 1.1.1**.

# Apache Home Page



Available for many platforms:  
Win64 (Win7-10), Win32, Solaris, Linux, macOS, etc.

# Apache httpd server

- The Apache httpd server
  - is a powerful, flexible, HTTP 1.1 & HTTP/2 compliant web server
  - implements the latest protocols, including **HTTP/2**
  - is highly configurable and extensible with third-party modules
  - can be customized by writing 'modules' using the Apache module API
  - provides full source code and comes with an unrestrictive license
  - runs on Windows, Netware, Mac OS X, Linux, and most versions of Unix, as well as several other operating systems
  - Latest release 2.4.46

# Configuring Apache

- server functionality is available through modules which are either built-into or loaded into the server
- server instructions
  - apache reads its run-time configuration instructions from text files
- no GUI available (visual start/stop available on Windows)
- 182 configuration directives in the basic package
- On UNIX, Apache starts several processes to implement the server and handle requests
- On Windows there are only two processes, a parent and a child that handles all requests. Within the child all requests are handled by separate “threads”
- See <http://httpd.apache.org/docs/2.4/configuring.html>

# How does Apache Work?

- Apache runs under a multitasking operating system, e.g., UNIX, macOS, Windows
- the binary is called httpd
- Key directories include:
  - conf contains configuration files
  - htdocs contains html files
  - logs contains logging data
  - cgi-bin contains executable scripts
  - icons set of \*.gif files used e.g. in directory listings
  - src contains actual source code if you download the source distribution
- Apache idles, listening to an IP address and port

## **Configuring Apache - conf Directory**

- In the conf directory there is a file provided with model configuration parameters
  - httpd.conf
- This file must be edited
- Partial contents of the mime.types file

text/html

html htm

text/plain

txt

text/richtext

rtx

video/mpeg

mpeg mpg mpe

video/quicktime

qt mov

video/x-msvideo

avi

x-world/x-vrml

wrl vrml

## Apache Settings - httpd.conf

- Directives are keywords followed by their value, e.g.

Listen 9637

ServerAdmin horowitz@usc.edu

ServerRoot /home/scf-03/csci571/WebServer/apache\_2.2

ErrorLog logs/error\_log

DocumentRoot "/home/scf-17/csci571c/apache2/htdocs"

ScriptAlias /cgi-bin/ "/home/scf-17/csci571c/apache2/cgi-bin/"

- the above are ones that *must* be set

## **Some Apache Settings - httpd.conf**

```
Timeout: The number of seconds before receives
and sends time out

Timeout 300

KeepAlive: Whether or not to allow persistent
connections (more than one request per
connection). Set to "Off" to deactivate.

KeepAlive On

MaxKeepAliveRequests: The maximum number of
requests to allow during a persistent connection.
Set to 0 to allow an unlimited amount. They
recommend you leave this number high, for maximum
performance.

MaxKeepAliveRequests 100

KeepAliveTimeout: Number of seconds to wait for
the next request

KeepAliveTimeout 15
```

## **Apache Settings – http-mpm.conf**

```
Server-pool size regulation. Apache dynamically
adapts to the load it sees --- it tries to
maintain enough server processes to handle the
current load, plus a few spare servers to
handle transient load spikes. It does this by
periodically checking how many servers are
waiting for a request. If there are fewer than
MinSpareServers, it creates a new spare. If
there are more than MaxSpareServers, some of
the spares die off.
```

```
MinSpareServers 2
```

```
MaxSpareServers 2
```

```
StartServers 2 #number of servers to start
```

```
MaxClients 2
```

## **Apache Settings - httpd.conf**

```
DocumentRoot: The directory out of which you will
serve your documents.

DocumentRoot /home/cscixxx/WebServer/apache/htdocs

UserDir: The name of the directory which is
appended onto a user's home directory if a ~user
request is received.

UserDir public_html

DirectoryIndex: Name of the file or files to use
as a pre-written HTML

DirectoryIndex index.html

FancyIndexing is whether you want fancy directory
indexing or standard

IndexOptions FancyIndexing
```

## **Apache Settings - Scripting**

```
ScriptAlias: This controls which directories contain server scripts. Format: ScriptAlias fakename realname
ScriptAlias /cgi-bin/
/home/cscixxx/WebServer/apache/cgi-bin/
If you want to use server side includes, or CGI outside ScriptAliased directories, uncomment the following lines.
AddType allows you to tweak mime.types without actually editing it, or to make certain files to be certain types. Format: AddType type/subtype ext1
To use CGI scripts:
#AddHandler cgi-script .cgi
To use server-parsed HTML files
#AddType text/html .shtml
#AddHandler server-parsed .shtml
```

# Authentication in Apache

- There are two methods for controlling access to directories
  1. the file access.conf in the conf/ directory can be used
    - placing access directives in a <Directory> block is the preferred method
  2. per-directory access rules can be set by a file placed in a specific directory
    - the name of the file is set by the directive AccessFileName
    - .htaccess is the default name
    - however, using .htaccess slows down the server;
    - NOTE: initially .htaccess is turned OFF
- When the server attempts to retrieve a document, it looks for an access control file in the directory or the parent directory.
- The file it looks for is set by the directive AccessFileName

## Some Sectioning Directives

- The server configuration files use sectioning directives to define special access rights within a section or sub-directory
- Example: <Directory> specifies directory to which directives apply
- Options controls which server features are available
  - None
  - All
  - FollowSymLinks follow symbolic links
  - ExecCGI execution of CGI scripts allowed
  - Includes server-side includes are enabled
  - Indexes will automatically return a list of files in the directory

## **Limit Sectioning Directive**

- <Limit> controls which clients can access a directory; directives within LIMIT include:
  - order in which deny and allow are evaluated
  - deny from host1, host2, ...
  - allow from host1, host2, ...
  - require named-users or group-users or AuthUserFile
  - referer allows access only from this directory
  - satisfy all or any

## Using LIMIT

- Host Filtering is used to limit document trees to certain machines
- Example 1: to limit access to the cscixxx public\_html documents to USC only

```
<Directory /usr/~cscixxx/public_html/>
<Limit GET>
order deny,allow
deny from all
allow from .usc.edu
</Limit>
</Directory>
```

- If someone tries to access documents in this directory from outside of usc they get a 403 Forbidden message

## Using LIMIT (cont'd)

- Example 2: to limit documents so USC people CANNOT access them

```
<Directory /usr/~cscixxx/public_html/>
<Limit GET>
order allow,deny
allow from all
deny from from .usc.edu
</Limit>
</Directory>
```

## Using LIMIT (cont'd)

- Example 3: a directive used to limit access to only USC and ISI domains

```
<limit GET>
order deny, allow
deny from all
allow from 128.125
allow from 128.9
</Limit>
```

## Using LIMIT (cont'd)

- Example 4: Suppose you want to restrict files in a directory called secure/ to the user named student1 and password XXXYYYY
- Step 1. create a file called .htaccess in directory secure/ that contains these 7 lines:

```
AuthUserFile /otherdir/.htpasswd must be a full
UNIX pathname
```

```
AuthGroupFile /dev/null there is no group file
```

```
AuthName ByPassword any name is OK here
```

```
AuthType Basic Basic or Digest is OK
```

```
<Limit GET>
```

```
require user student1
```

```
</Limit>
```

## Using LIMIT (cont'd)

- Step 2. Next create the password file /otherdir/.htpasswd using the htpasswd program distributed with apache.
- Type

```
htpasswd -c /otherdir/.htpasswd student1
and enter the password twice.
```

- You are done.
- Step 3. To add other names and passwords run htpasswd without the -c option.

## Using LIMIT (cont'd)

- To generalize this example to include several students,
- Step 4. create a group file called /otherdir/.htgroup which looks like:

my-users: student1 student2 student3 student4

Alter the .htaccess file to look like

AuthUserFile /otherdir/.htpasswd

AuthGroupFile /otherdir/.htgroup      Now it points  
to the group file

AuthName ByPassword

AuthType Basic

<Limit GET>

require group my-users

</Limit>

## Contents of .htaccess

- The .htaccess file contains pointers to the users file, the group file, and defines the name and type of authorization

nunki.usc.edu(10) : more .htaccess

```
AuthUserFile /home-scf-03/cscixxx/pwd/.htpasswd
```

```
AuthGroupFile /home-scf-03/cscixxx/pwd/.htgroup
```

```
AuthName OnlineCourse
```

```
AuthType Basic
```

## **Contents of .htpasswd**

- This file contains a set of names and passwords

nunki.usc.edu(13) : more .htpasswd

admin:kjpokufbLdrXU

chungw:2Ds5QooEqT1HM

csci351:ykGZC.bMvV8SY

csci351notes:7814Vi6oJtq9c

csci565:t.OSxBunNhigg

csci571:NUKyNCCLvgLH.

nunki.usc.edu(13) : more .htgroup

admin: admin csci571 csci565

student: admin csci571 csci565

# Adding a Smart Error Page

- When someone requests a page or directory that does not exist it is better to return an error page than a 404 Not Found message
- For an example try: `http://www.usc.edu/x.html`
- Look at `/conf/httpd.conf`
- find the section that describes the properties of your domain, e.g.

```
<VirtualHost www.test.com>
ServerAdmin webmaster@test.com
DocumentRoot /web/test
ServerName www.test.com
ErrorLog /log/test/error_log
transferLog /log/test/access_log
AgentLog /log/test/agent_log
RefererLog /log/test/refer_log
</VirtualHost>
```

# Adding a Smart Error Page

- Insert the line

```
ErrorDocument 404 /error-page.html
```

- place the error page at

DocumentRoot/ErrorDocument or in this case  
/web/test/error-page.html

Sample contents of error-page.html:

```
<HTML><HEAD><TITLE>Test Error Page</TITLE></HEAD>
<BODY>Error. You have requested a page that
cannot be found. You may wish to return to the
home page by
clicking here.
</BODY></HTML>
```

# Apache Approaches to Server-Side Processing

- **CGI** - the standard way to invoke an external process, e.g., perl & perl.exe
- **Fast CGI, Java Servlets** - Apache passes a request to another running server which executes the request and returns the result
  - see mod\_fastcgi at <http://www.fastcgi.com>
- **Interpreters** embedded in the server itself.  
There are roughly two categories
  - Modules that answer or modify requests directly, e.g., mod\_perl (Perl)
  - Modules aimed to process commands embedded in HTML pages before serving it to the client. e.g., **mod\_php** (PHP)

# Virtual Hosting

- *Virtual hosting* is a way of setting up your server so that it can appear to be multiple Web sites at once, depending upon how it's accessed by clients
- ISPs do this a lot
- The Apache Web server software allows you to create as many such virtual site identities as your system's limits will allow;
- Example of a virtual host definition.

```
<VirtualHost 10.0.0.1>
 ServerName WWW.Mydomain.Com
 ServerAdmin Webmaster@Mydomain.Com
 DocumentRoot /home/httpd/mydomain
 CustomLog logs/mydomain_log CLF
</VirtualHost>
```

## Virtual Hosting (cont'd)

- there are two ways to do virtual hosting
  - *address-based* virtual hosting uses IP addresses
  - *name-based* hosting allow a single IP address to have multiple identities, e.g.

```
NameVirtualHost 10.0.0.1 this example assigns
<VirtualHost 10.0.0.1> two domain names to the
 ServerName WWW.Mydomain.Com same IP address
 ServerAlias WWW.X.Com WWW.Y.Org
 ServerAdmin Webmaster@Mydomain.Com
 DocumentRoot /home/httpd/mydomain
 CustomLog logs/mydomain_log_CLF
</VirtualHost>
<VirtualHost 10.0.0.1>
 ServerName WWW.Other.Org
 ServerAlias WWW.Z.Net WWW.A.Org
 ServerAdmin Webmaster@Other.Org
 DocumentRoot /home/httpd/other
 CustomLog logs/other_log_CLF
</VirtualHost>
```

# Content Negotiation

- The client provides a set of preferences concerning the preferred attributes of a document, and the server tries to come up with the best match from the versions available
  - If there's only one version of a document, then that's what the server is going to send
- The set of all versions of a document that are available to the server is called the resource's list of *variants*.
- The ways in which they differ, such as by language or encoding, are called its *dimensions*.
- The Apache server can negotiate along the
  - content-type,
  - encoding, and
  - language dimensions

# Content Negotiation

- The primary use of negotiation on the Web today is to select translations of text.
  - Typically, a work is prepared in a default language such as English, and the result put in a file named `text.html` or some such. Then translated versions are prepared, and each is put into an appropriate-named file: `text.html.da` for Danish, `text.html.fr` for French, `text.html.en-gb` for English (Great Britain dialect), and so on.
- You can either prepare a list of variants yourself, or you can allow Apache's `mod_negotiation` module to calculate the list itself by scanning the directory.
- The latter method is more common, since it is easier and automatically pick up new variants as soon as they are available, but it does impose a slight performance penalty due to the directory scan.

# **Apache as a Proxy Server**

- proxy servers are frequently used in fire walled environments to provide a single point of contact between the users inside and the Web outside the firewall.
  - The users ask for a page, the browser sends the request to the proxy server, the proxy server sends the request to the origin server, and the data flows back the same way.
- A second function commonly associated with proxy servers is caching.
- To use the proxy functions in your Apache Web server, the `mod_proxy` module must be activated, either by being statically linked in (less common than formerly), or turned on at run-time with directives, e.g.

**AddModule mod\_proxy.c**

# Apache as a Proxy Server, Proxy Directives

- **ProxyRequests** – This is the 'master directive,' which enables and disables proxy operation overall. It takes a single keyword argument, which must be either On or Off
- **ProxyRemote** – tells the Apache server that certain requests are to be forwarded to yet another system rather than being handled locally, allowing for cascading of proxies
- **ProxyBlock** – allows you to selectively block the fetching of certain resources through the proxy. It takes a list of space-separated case-insensitive phrases or URI substrings which should be blocked.
  - "ProxyBlock foo" would block all requests in which "foo" appeared, in any combination of upper- and lower-case, including "http://foo.com/bar/" and "http://bar.org/zardofoo.html". You can block all proxy requests with "ProxyBlock \*"

# Controlling the Apache Cache

- **CacheRoot** – Defines the directory that's the top of the cache hierarchy
  - Apache uses its own naming conventions for the stuff it stores.
- **CacheSize** – Specifies the largest number of kilobytes of space you want to allocate to the proxy cache.
- **CacheDirLevels** – Controls how deep the cache directory tree is allowed to grow
- **CacheDirLength** – Used to define the number of characters used in naming each level of cache subdirectory
- **NoCache** – Identifies proxy-accessed documents that shouldn't be cached. The format is the same as for the ProxyBlock directive.

## Proxy Host Example

- Usually, a proxy server is set up as a completely separate virtual host from the rest of the server's operation (which is likely to be handling normal requests).
- You can do this with a <VirtualHost> stanza. The <IfModule> container will only be evaluated if the proxy module is part of the running server

```
<IfModule mod_proxy.c>
 Listen 10.0.0.1:8080
 <VirtualHost 10.0.0.1:8080>
 ProxyRequests On
 DocumentRoot /home/httpd/htdocs/proxy
 </VirtualHost>
</IfModule>
```

# Analyzing Web Server Log Files

- A sample entry from a log file

```
Host date/time method file HTTP ver status bytes
marmot.usc.edu - - [17/Feb/2009:23:21:09 -0800] "GET" / HTTP/1.0" 200 1316
```

- There are different log file formats
  - W3C Extended Log File Format
    - Date/time, server port, URL and query, protocol version, user agent and cookie, referrer, etc.
  - Microsoft IIS log file format
    - IP, user, date/time, server name, server IP, bytes received, bytes sent, protocol, method, etc.
  - Other file formats
    - Cerfnet, apache extended, CERN common, Lotus Domino, Netscape Common, etc.

## Definition of Terms

- A *hit*: a request for a page, image, cgi script, etc.
- A *page view*: a request for an html page
- Unique host vs. unique visitor: aol.com is a unique host, so this statistic may not be useful; cookies are best way to track unique visitors
- An *impression*: is a view of an advertisement
- A *click-through* is when an advertisement is actually clicked on

## Common Log Format

- Each visit to your site produces a record of the request in log file
- log files in Apache are typically stored in the logs directory
- Some fields include:
  - Host client hostname or IP address
  - ident remote identity, often a dash
  - usr authenticated user name
  - time date/time request is received
  - req HTTP request
  - s1 server's HTTP response status code
  - c1 Content-length
- Example entries from access log

marmot.usc.edu - - [17/Feb/2008:23:21:09 -0800]  
"GET / HTTP/1.0" 200 1316

128.125.229.195 - - [23/Feb/2008:10:32:09 -0800]  
"GET / HTTP/1.1" 200 1316

## Other Entries in access Log

- ciscoserv-229-180.usc.edu - - [17/Feb/2008:23:16:51 -0800] "GET /cgi-bin/secure/admin/student.pl HTTP/1.1" 401 362
- 128.125.229.195 - admin [23/Feb/2008:10:56:21 -0800] "POST /cgi-bin/secure/admin/student.pl HTTP/1.1" 200 807
- 209.85.18.67 - - [27/Feb/2008:01:12:38 -0800] "GET /ftp/templates.zip HTTP/1.1" 200 30780
- nunki.usc.edu - - [28/Feb/2008:00:48:22 -0800] "GET / HTTP/1.0" 200 955
- nunki.usc.edu - - [28/Feb/2008:00:48:26 -0800] "GET /cgi-bin/Count.cgi?df=count.dat HTTP/1.0" 200 207

## access\_log Contents

- Here are some lines from a single user session

```
lvl-sun702.usc.edu - - [09/Feb/2008:19:15:36
-0800] "GET /newsdigest/index.html HTTP/1.0"
200 3630
```

```
lvl-sun702.usc.edu - - [09/Feb/2008:19:15:37
-0800] "GET / HTTP/1.0" 200 955
```

```
lvl-sun702.usc.edu - - [09/Feb/2008:19:15:39
-0800] "GET /newsdigest/index.html HTTP/1.0"
200 3630
```
- A *user session* includes all hits and requests made by a user terminated by “n” minutes of inactivity

## Sample error\_log Contents

- Here are some lines from the error\_log file

- The file was not found**

```
[Mon Jan 29 11:47:36 2008] httpd: access to /home/scf-13/csci665/www/pleader/horohome.html failed for internet-gw.watson.ibm.com, reason: No file matching URL: /pleader/horohome.html from -
```

- **authorization entered is incorrect**

```
[Thu Sep 24 10:17:41 2008] [error] [client 128.125.50.47] user csci571 not found: /cgi-bin/secure/test-cgi
```

- **Perl script fails**

```
[Sun Sep 30 01:15:01 2008] [error] [client 71.140.65.229] Premature end of script headers: /auto/home-scf-22/csci571/WebServer/Apache_1.3.36/cgi-bin/cgipm/test6.pl
```

## More Examples of Error Log File Entries

- Undefined subroutine &main::Export\_to\_Excel called at /home/scf-03/csci571/WebServer/apache\_1.2.5/cgi-bin/admin/student\_db.pl line 75.
- [Tue Dec 29 02:07:47 2008] access to /auto/home-scf-03/csci571/WebServer/apache\_1.2.5/htdocs/robots.txt failed for 193.189.227.21, reason: File does not exist
- [Tue Dec 29 20:11:22 2008] send directory lost connection to client 128.125.223.237

# **Performance and Security Considerations**

- Don't log more than needed
- Look for 401 (unauthorized) and 403 forbidden status codes
- Look at the response times
  - Which parts of the web site have the longest response times?
  - Where are the bottlenecks
- Look for failed page hits (404, 5xx)
- Look for cached hits
- Look for browsers and platforms of clients

# Tools to Analyze Log Files

- Some commercially available tools for IIS
  - **WebTrends**  
<http://www.webtrends.com>  
See also  
[http://cs-server.usc.edu:45678/resources/web\\_servers/webtrends/report2.html](http://cs-server.usc.edu:45678/resources/web_servers/webtrends/report2.html)
  - Marketware, webmanager, MS site server, LiveStats

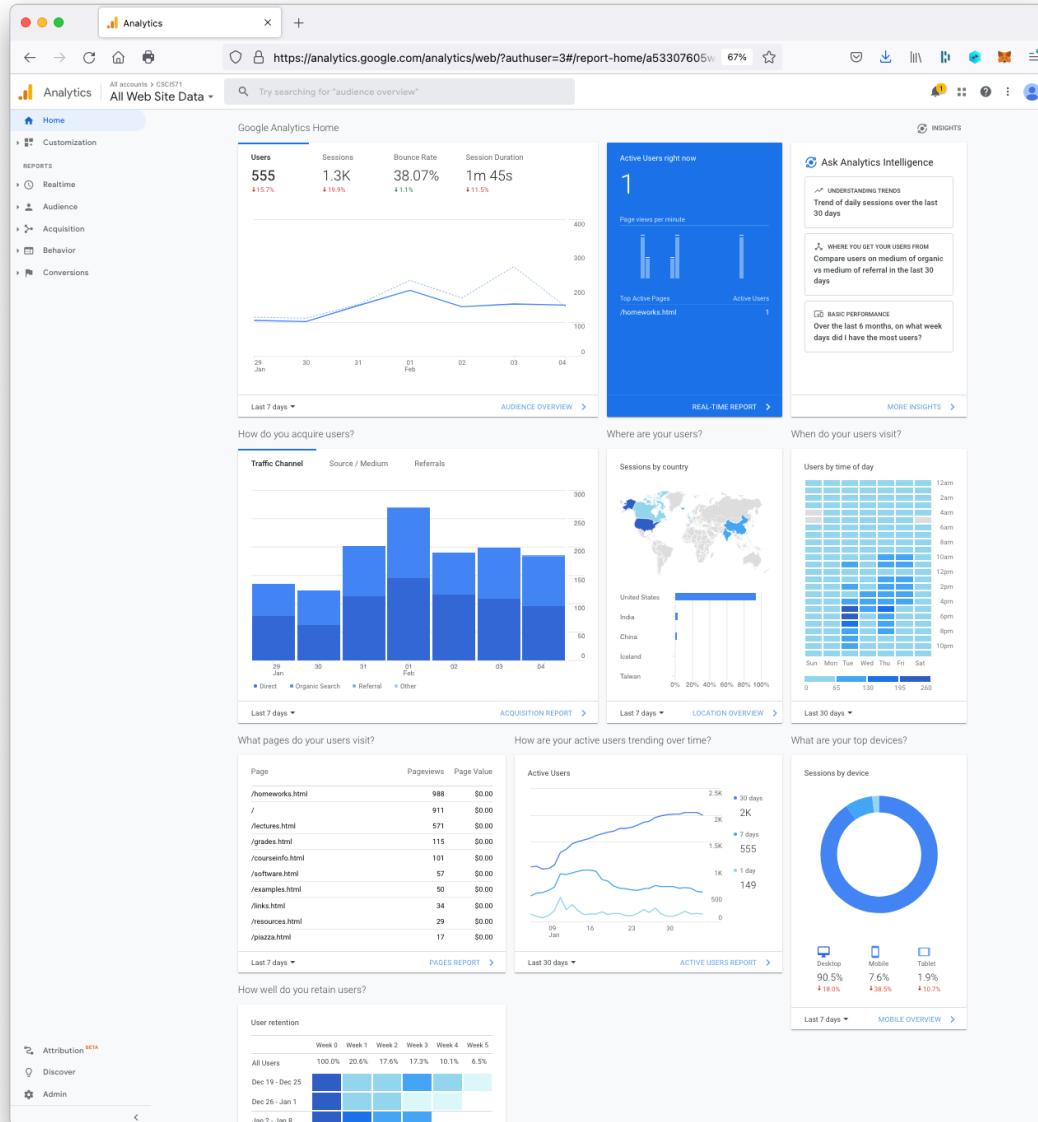
# Google Analytics

- For any web server
  - **Google Analytics**. See:  
<http://www.google.com/analytics/>
- Install **global site tag**, after <head> tag, on every site page.  
<https://developers.google.com/tag-platform/gtagjs>

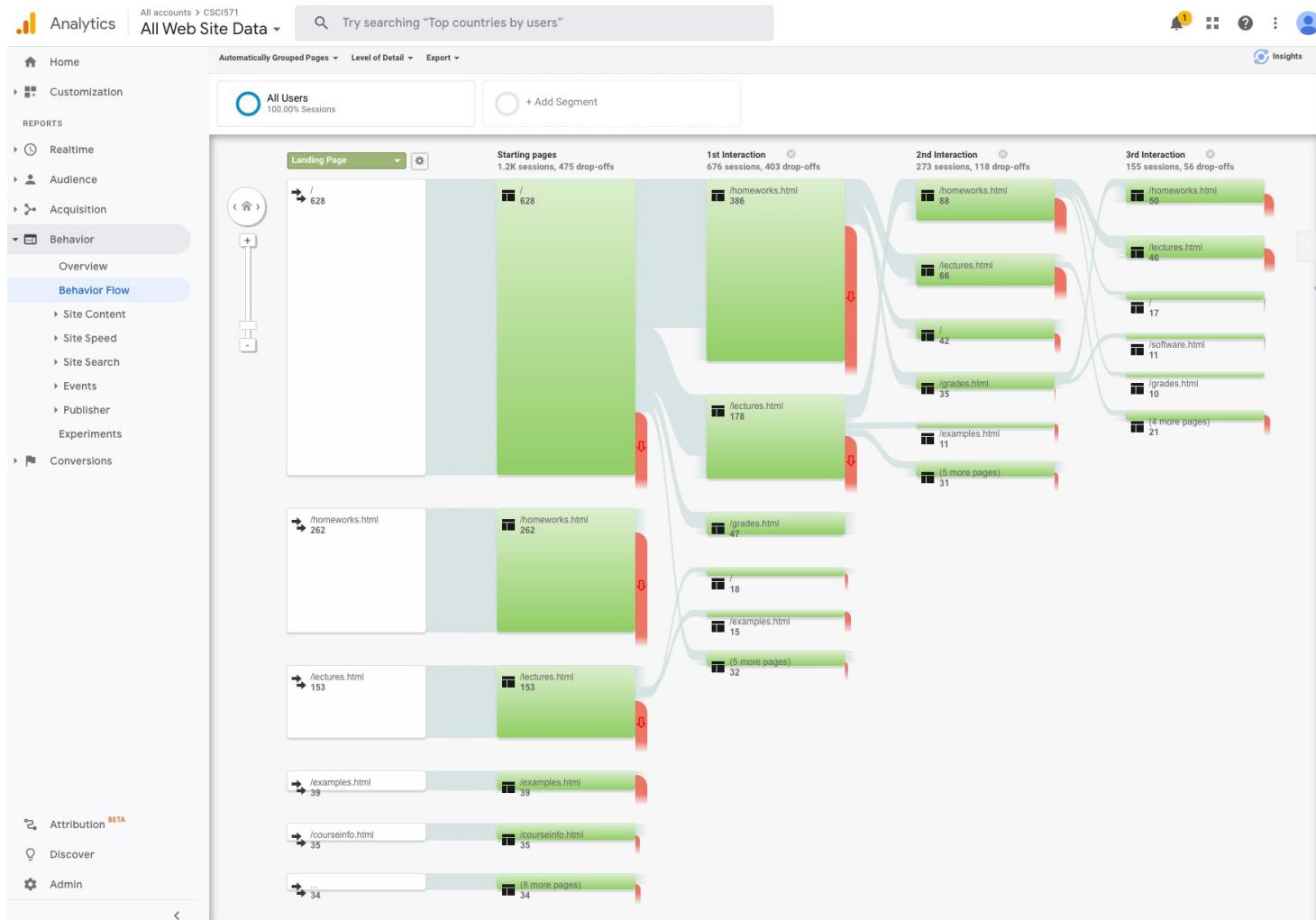
```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=GA_MEASUREMENT_ID"></script>
<script>
 window.dataLayer = window.dataLayer || [];
 function gtag(){window.dataLayer.push(arguments);}
 gtag('js', new Date());

 gtag('config', 'GA_MEASUREMENT_ID');
</script>
```

# Google Analytics Dashboard



# Google Analytics Behavior Flow



# **Lecture**

## **JavaScript and Browser Manipulation**

# Testing the Speed of Your JavaScript (2009)

CSCI 571 Home Page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Gmail - Inbox (1... Department Of ... CSCI 571 ... Learning JavaSc... JavaScript Obj...

<http://www.scf.usc.edu/~csd571/> Google

**SunSpider JavaScript Benchmark Results**

Run Again

<http://www2.webkit.org/perf/sunspider-0.9/sunspider-results.html?%7B%23d-c>  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

RESULTS (means and 95% confidence intervals)

Total: 5103.6ms +/- 15.7%

3d:  
cube:  
morph:  
raytrace:  
  
access:  
binary-trees:  
fannkuch:  
nbody:  
nsieve:  
  
bitops:  
3bit-bits-in-byte:  
bits-in-byte:  
bitwise-and:  
nsieve-bits:  
  
controlflow:  
recursive:  
  
crypto:  
aes:

Done 0.419s YSlow

Firefox, v 3.0  
5103ms

SunSpider JavaScript Benchmark Results

Run Again

<http://www2.webkit.org/perf/sunspider-0.9/sunspider-results.html?%7B%23d-c>  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

RESULTS (means and 95% confidence intervals)

Total: 6202.6ms +/- 7.2%

3d:  
cube:  
morph:  
raytrace:  
  
access:  
binary-trees:  
fannkuch:  
nbody:  
nsieve:  
  
bitops:  
3bit-bits-in-byte:  
bits-in-byte:  
bitwise-and:  
nsieve-bits:  
  
controlflow:  
recursive:  
  
crypto:  
aes:

Safari v 3.2  
6202ms

SunSpider JavaScript Benchmark Results - Windows Internet Explorer

File Edit View Favorites Tools Help

Google

**SunSpider JavaScript Benchmark Results**

Run Again

<http://www2.webkit.org/perf/sunspider-0.9/sunspider-results.html?%7B%23d-c>  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

RESULTS (means and 95% confidence intervals)

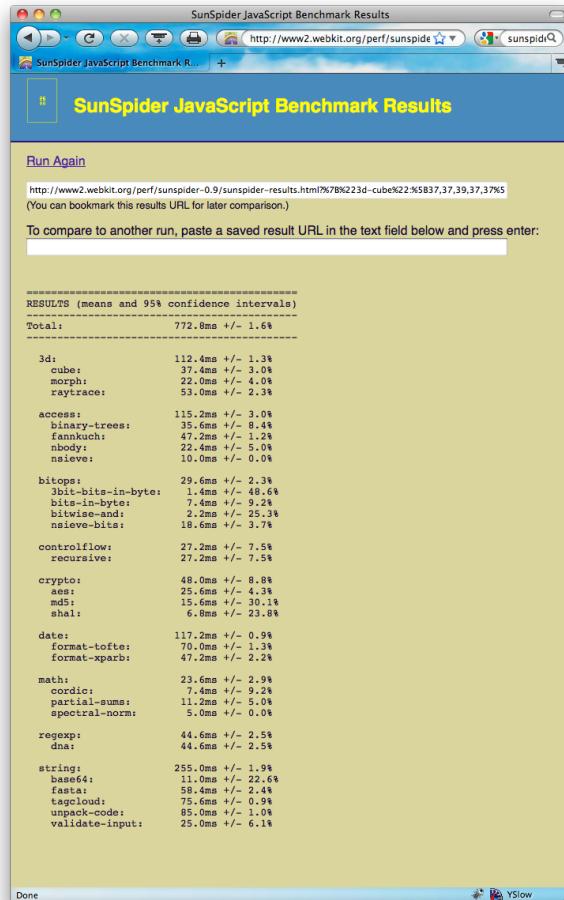
Total: 45928.2ms +/- 3.8%

3d:  
cube:  
morph:  
raytrace:  
  
access:  
binary-trees:  
fannkuch:  
nbody:  
nsieve:  
  
bitops:  
3bit-bits-in-byte:  
bits-in-byte:  
bitwise-and:  
nsieve-bits:  
  
controlflow:  
recursive:  
  
crypto:  
aes:

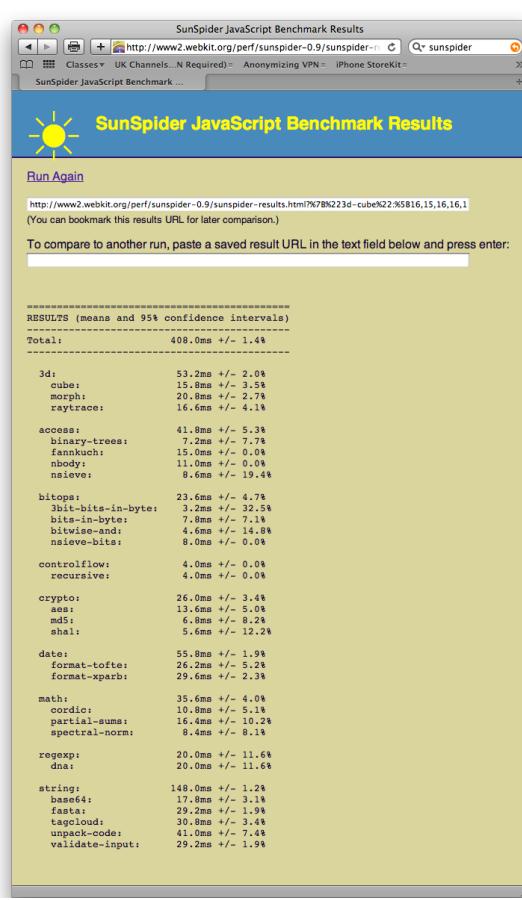
Done Internet 100%

I.E. 7.0  
45,928ms

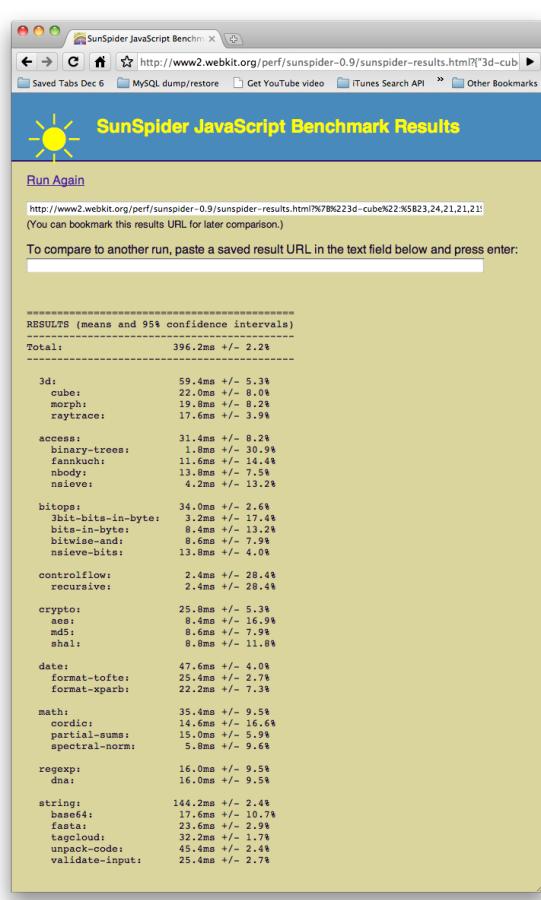
# Testing the Speed of Your JavaScript (2010)



Firefox, v 3.6  
772 ms

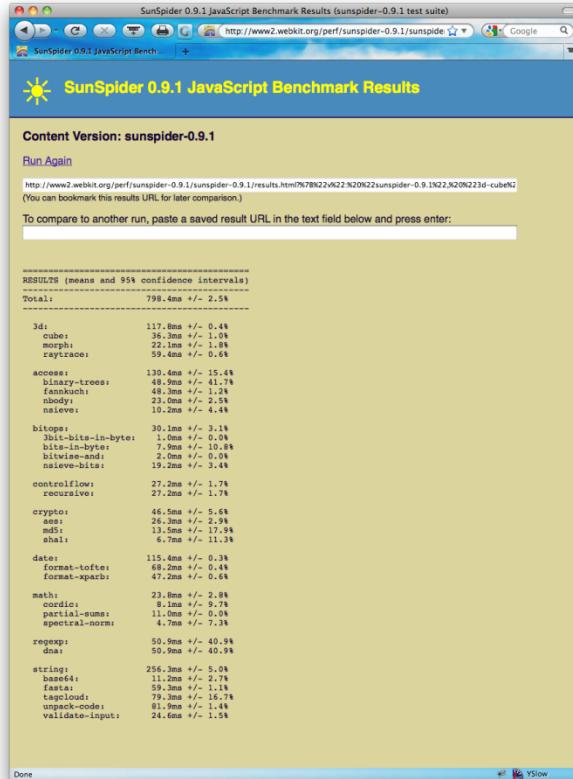


Safari v 4.0  
408 ms

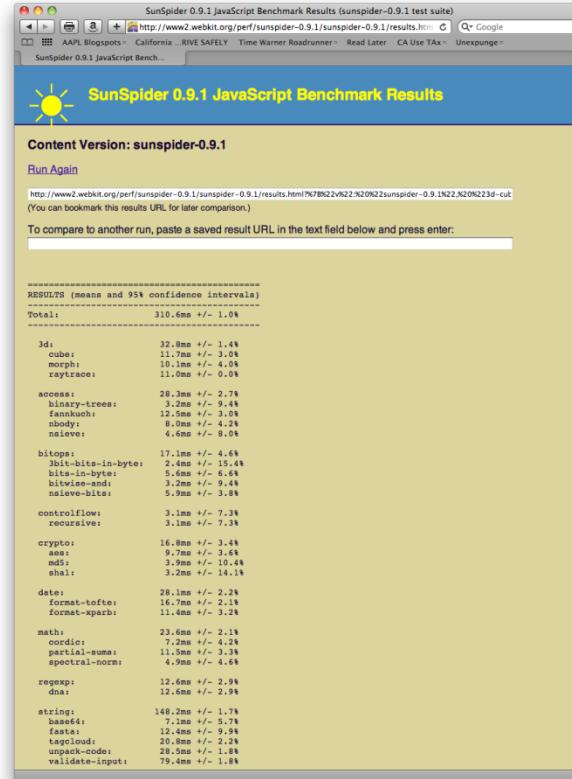


Chrome v 4.0  
396 ms  
Chrome v 14.0  
214 ms

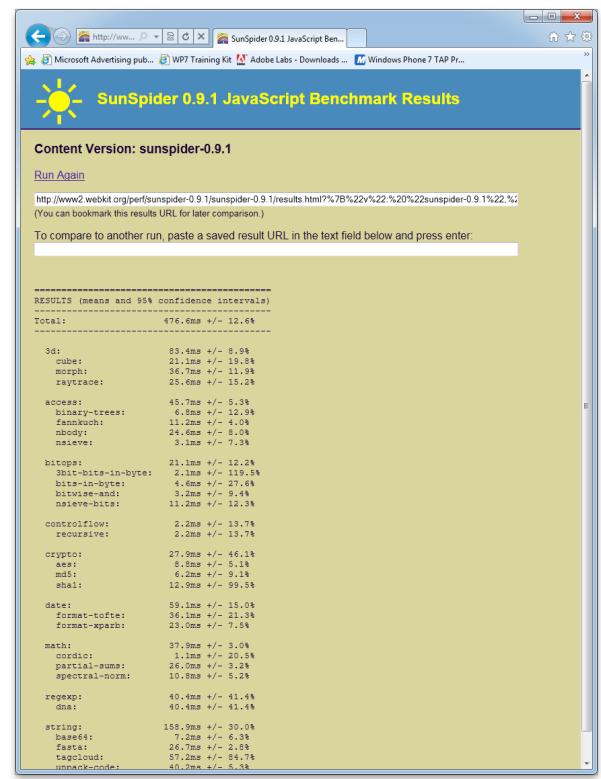
# Testing the Speed of Your JavaScript (2011)



Firefox, v 3.6.13  
798 ms  
Firefox, v.6.0.2  
187 ms



Safari v 5.0.3  
310 ms  
Safari v. 5.1  
198 ms



IE v 9.0 (beta)  
476 ms  
IE v 9.0  
314 ms

# Testing the Speed of Your JavaScript (2012)

**SunSpider 0.9.1 JavaScript Benchmark Results**

**Content Version:** sunspider-0.9.1

[Run Again](#)

<http://www.webkit.org/perf/sunspider-0.9.1/sunspider-0.9.1/results.html?7B%22>  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

**RESULTS (means and 95% confidence intervals)**

Total:	166.9ms +/- 16.6%
3d:	26.7ms +/- 2.5%
cube:	10.8ms +/- 2.8%
morph:	5.5ms +/- 6.8%
raytrace:	10.4ms +/- 3.5%
access:	16.0ms +/- 5.2%
binary-trees:	2.5ms +/- 14.2%
fannkuch:	6.3ms +/- 5.5%
nbody:	3.3ms +/- 10.5%
nsieve:	3.8ms +/- 7.9%
bitops:	9.2ms +/- 6.1%
3bit-bits-in-byte:	0.8ms +/- 37.7%

Firefox 16  
166 ms

**SunSpider 0.9.1 JavaScript Benchmark Results**

**Content Version:** sunspider-0.9.1

[Run Again](#)

<http://www.webkit.org/perf/sunspider-0.9.1/sunspider-0.9.1/results.html?7B%22>  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

**RESULTS (means and 95% confidence intervals)**

Total:	149.3ms +/- 1.5%
3d:	22.1ms +/- 2.4%
cube:	7.0ms +/- 0.8%
morph:	6.4ms +/- 5.8%
raytrace:	8.7ms +/- 4.0%
access:	12.9ms +/- 4.1%
binary-trees:	1.3ms +/- 26.6%
fannkuch:	6.2ms +/- 4.9%
nbody:	3.3ms +/- 10.5%
nsieve:	2.1ms +/- 10.8%
bitops:	10.9ms +/- 6.5%
3bit-bits-in-byte:	0.7ms +/- 49.3%
bits-in-byte:	3.9ms +/- 5.8%
bitwise-and:	2.5ms +/- 15.1%
nsieve-bits:	3.8ms +/- 7.9%

Chrome 21.0  
149 ms

**SunSpider 0.9.1 JavaScript Benchmark Results**

**Content Version:** sunspider-0.9.1

[Run Again](#)

<http://www.webkit.org/perf/sunspider-0.9.1/sunspider-0.9.1/results.html?7B%22>  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

**RESULTS (means and 95% confidence intervals)**

Total:	141.8ms +/- 1.2%
3d:	19.8ms +/- 1.5%
cube:	8.2ms +/- 0.8%
morph:	2.1ms +/- 10.8%
raytrace:	9.0ms +/- 0.0%
access:	18.8ms +/- 2.0%
binary-trees:	2.2ms +/- 13.7%
fannkuch:	6.8ms +/- 4.4%
nbody:	7.8ms +/- 10.8%
nsieve:	1.8ms +/- 16.7%
bitops:	11.8ms +/- 4.4%
3bit-bits-in-byte:	0.9ms +/- 25.1%
bits-in-byte:	2.6ms +/- 14.2%
bitwise-and:	3.0ms +/- 10.8%
nsieve-bits:	1.9ms +/- 0.0%
controlflow:	1.0ms +/- 0.0%
recursive:	1.0ms +/- 0.0%
crypto:	9.8ms +/- 5.3%
ass:	3.8ms +/- 7.9%

Internet Explorer 9.0  
141 ms

**SunSpider 0.9.1 JavaScript Benchmark Results**

**Content Version:** sunspider-0.9.1

[Run Again](#)

<http://www.webkit.org/perf/sunspider-0.9.1/sunspider-0.9.1/results.html?7B%22>  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

**RESULTS (means and 95% confidence intervals)**

Total:	138.0ms +/- 6.2%
3d:	23.1ms +/- 32.5%
cube:	9.3ms +/- 80.2%
morph:	6.0ms +/- 0.0%
raytrace:	7.8ms +/- 3.9%
access:	12.4ms +/- 4.0%
binary-trees:	1.3ms +/- 26.6%
fannkuch:	5.5ms +/- 6.8%
nbody:	2.4ms +/- 10.8%
nsieve:	3.0ms +/- 0.0%
bitops:	7.8ms +/- 7.2%
3bit-bits-in-byte:	0.9ms +/- 25.1%
bits-in-byte:	2.1ms +/- 10.8%

Safari 6.0  
138 ms

Summary  
over the  
years

	Firefox	Safari	I.E.	Chrome
2009	5,103	6,202	45,928	
2010	772	408		214
2011	187	198	314	
2012	166	138	141	149

# Mobile Speed of JavaScript (2012-2014)

●●○○ AT&T LTE 3:58 PM ↗ 100%

webkit.org

[Run Again](#)

http://www.webkit.org/perf/runspider-1.0.1/runspider-1.0.1-results.html?%7B%22v%22%7D%22runspider-1.0.1%22%20%2234-cube%22%7D%2215.39.39.41.41.41.42.42.40.3  
(You can bookmark this results URL for later comparison.)

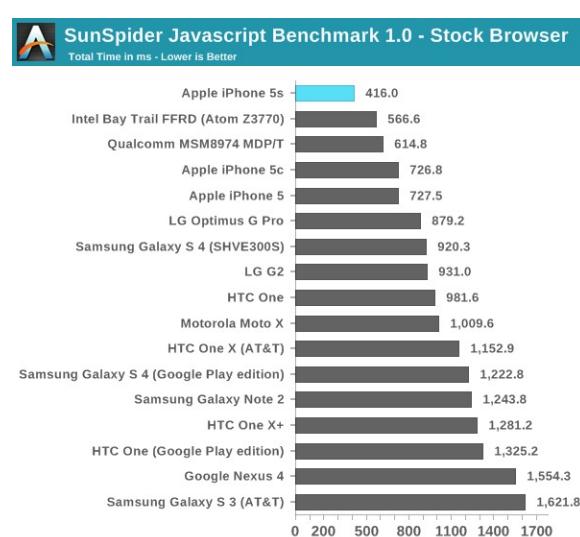
To compare to another run, paste a saved result URL in the text field below and press enter:

=====  
RESULTS (means and 95% confidence intervals)  
=====

Total:	718.5ms +/- 1.4%
3d:	117.8ms +/- 5.3%
cube:	40.3ms +/- 2.2%
morph:	33.5ms +/- 3.1%
raytrace:	44.0ms +/- 14.9%
access:	62.4ms +/- 2.4%
binary-trees:	9.9ms +/- 2.2%
fannkuch:	27.4ms +/- 4.5%
nbody:	16.2ms +/- 1.9%
nsieve:	8.9ms +/- 5.9%
bitops:	36.7ms +/- 2.3%
3bit-bits-in-byte:	2.6ms +/- 10.8%
bits-in-byte:	13.3ms +/- 2.6%
bitwise-and:	7.9ms +/- 5.1%
nsieve-bits:	12.7ms +/- 5.3%
controlflow:	9.3ms +/- 5.2%
recursive:	9.3ms +/- 5.2%
crypto:	60.3ms +/- 4.1%
aes:	30.4ms +/- 5.3%
md5:	18.0ms +/- 11.2%
sha1:	11.9ms +/- 7.7%
date:	88.8ms +/- 1.5%
format-tofte:	47.2ms +/- 2.1%
format-xparb:	41.6ms +/- 1.4%
math:	71.1ms +/- 3.2%
cordic:	15.1ms +/- 2.7%
partial-sums:	42.2ms +/- 4.4%
spectral-norm:	13.8ms +/- 4.1%
regexp:	36.6ms +/- 2.3%
dns:	36.6ms +/- 2.3%
string:	235.5ms +/- 2.3%
base64:	20.2ms +/- 4.3%
fastaa:	46.1ms +/- 2.3%
tagcloud:	48.8ms +/- 8.8%
unpack-code:	93.7ms +/- 1.6%
validate-input:	26.6ms +/- 5.8%

=====

< >



Safari, iPhone 5S  
416ms

●●○○ AT&T LTE 11:35 AM ↗ 80%

webkit.org

[Run Again](#)

http://www.webkit.org/perf/runspider-1.0.2/runspider-1.0.2-results.html?%7B%22v%22%7D%22runspider-1.0.2%22%20%2234-cube%22%7D%2215.18.14.14.15.15.14.14.15.  
(You can bookmark this results URL for later comparison.)

To compare to another run, paste a saved result URL in the text field below and press enter:

=====  
RESULTS (means and 95% confidence intervals)  
=====

Total:	352.7ms +/- 2.0%
3d:	49.7ms +/- 4.8%
cube:	14.8ms +/- 5.9%
morph:	14.7ms +/- 11.9%
raytrace:	19.2ms +/- 15.8%
access:	44.3ms +/- 7.9%
binary-trees:	5.4ms +/- 15.5%
fannkuch:	18.8ms +/- 16.4%
nbody:	12.8ms +/- 24.0%
nsieve:	7.3ms +/- 8.1%
bitops:	25.1ms +/- 13.2%
3bit-bits-in-byte:	3.2ms +/- 61.8%
bits-in-byte:	7.3ms +/- 9.3%
bitwise-and:	5.2ms +/- 5.8%
nsieve-bits:	9.3ms +/- 29.9%
controlflow:	4.6ms +/- 24.5%
recursive:	4.6ms +/- 24.5%
crypto:	28.2ms +/- 11.9%
aes:	14.7ms +/- 21.7%
md5:	7.2ms +/- 16.1%
sha1:	6.3ms +/- 7.7%
date:	39.7ms +/- 4.2%
format-tofte:	23.1ms +/- 4.0%
format-xparb:	16.6ms +/- 7.4%
math:	23.2ms +/- 2.4%
cordic:	8.1ms +/- 7.7%
partial-sums:	11.1ms +/- 3.7%
spectral-norm:	4.0ms +/- 16.8%
regexp:	15.5ms +/- 3.3%
dns:	15.5ms +/- 3.3%
string:	123.4ms +/- 2.2%
base64:	10.3ms +/- 3.2%
fastaa:	17.8ms +/- 3.2%

=====

< >

Safari, iPhone 6  
352ms

Safari, iPhone 5  
718ms

# JetStream (2015-2018) – JetStream2 (2020-2021)



Safari 9, iPhone 6s  
111.75 score  
Safari 10, iPhone 7  
164.79 score  
Safari 11, iPhone X  
216.74 score  
Safari 12, iPhone XS  
261.75



Safari 10, OS X  
155.76 score  
Safari 11, OS X  
246-255.60  
Chrome 45, OS X  
151.62 score

# browserbench.org

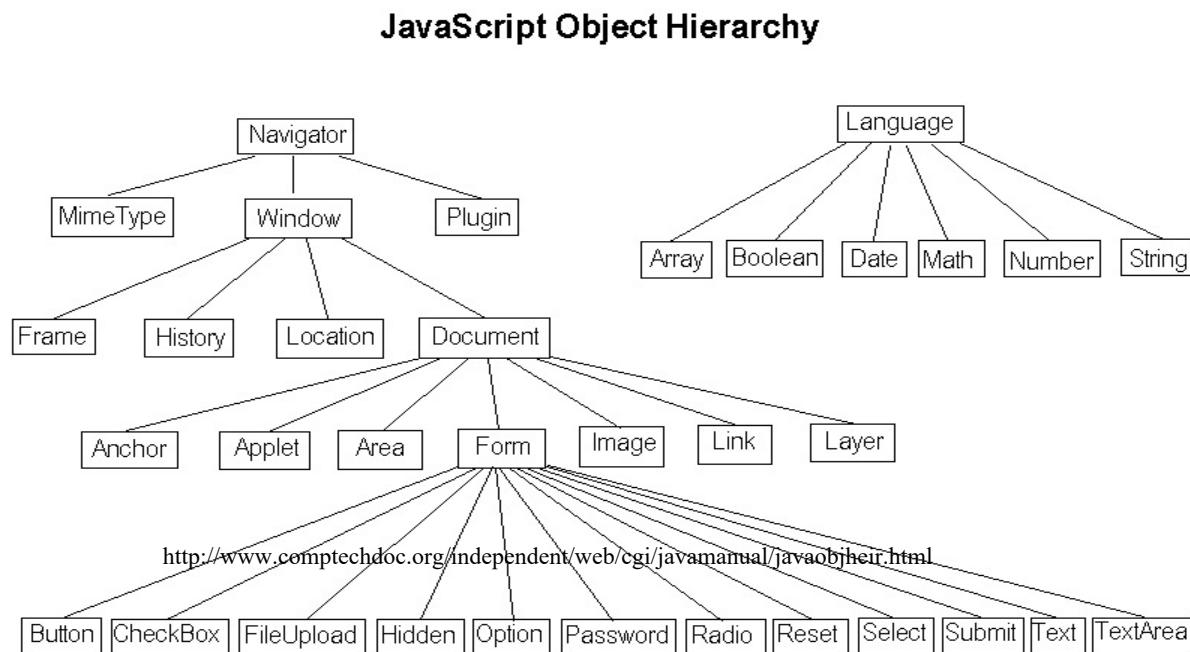


Safari, iPhone 11 Max Plus,  
iOS 14, 148.98 score  
Firefox 81, Mac mini,  
macOS 10.15, 90 score  
Safari, iPhone 12 Max Plus,  
iOS 14.6, 181.65 score

Safari. iPhone 13 Max Plus  
iOS 15.3, 220.41 score

# Javascript Object Hierarchy

JavaScript contains a set of built-in objects related to the browser. Each object includes a set of properties and methods



# The Window Object

- The **window** object is the highest level built-in JavaScript object; it corresponds to the browser window
- Some properties of the Window object include:
  - `window.closed` – a boolean indicating if the window is closed or not
  - `window.history` – returns the history object
  - `window.location` – returns the location object
  - `window.navigator` – returns the navigator object
  - `Window.parent` – returns the parent window of the current window
  - `Window.self` – returns the current window
  - `Window.status` – sets the text in the statusbar window
- Some methods of the Window object include:
  - `open()` – opens a new window
  - `blur()` – removes focus from the current window
  - `close()` – closes the current window
  - `focus()` – sets the focus to the current window
  - `resizeBy()` – resizes the window by the specified pixels
- For a complete list see, e.g., [http://www.w3schools.com/jsref/obj\\_window.asp](http://www.w3schools.com/jsref/obj_window.asp)

# Examples Using the Window Object

- To create a new window that shows only the toolbar and status bar and is resizable

```
window.open("newURL", "New Window", "toolbar,status,resizable")
```

- The height and width defaults are the same as the browser
- The new window is positioned in the upper left-hand corner of the screen
- A call to window.open() returns a value of the new window's object; this should always be assigned to a variable, e.g.

```
newWindow = window.open("", "")
if (newWindow != null) {
 newWindow.document.write("<HTML><HEAD>
 <TITLE>Hi</TITLE></HEAD><BODY>Hello
 World</BODY></HTML>") }
```

- Try [https://csci571.com/examples/js/js\\_22\\_2.html](https://csci571.com/examples/js/js_22_2.html)

## Parameters of `windows.open()`

- The `open()` method includes three top level parameters:

```
newWin = open(url, name, [features, [replace]]);
```

- **newWin**: is a reference to the new window object
- **url**: a string URL value to be fetched by the new window (can be empty)
- **name**: this name can be used with the HTML target attribute (do not use spaces in the name)
- **features**: a string that specifies the desired browser features (syntax: feature=value)
- **replace**: allows new entries to be made to the browser history

# More on the features argument of the windows.open method

- The ‘features’ string is a comma separated list of items (original list)
  - toolbar Back, Forward Buttons
  - location the URL input field
  - directories What's New, What's Cool, etc
  - status browser status line
  - menubar browser menu bar
  - scrollbars enables scrollbars when needed
  - resizable allows the window to be resizable
  - **width, height** window dimensions in pixels
- When the ‘features’ string is absent, the new browser window has all of the standard controls
- New, much larger set of recent ‘features’ available at:  
<https://developer.mozilla.org/en-US/docs/Web/API/Window/open>
- Chrome does not support most of the ‘features’
- Support varies by browser. Width & Height supported by all browsers

# Sample Code to open a window with attributes:

width=640, height=480, URL bar, browser toolbar, Menus, Directory buttons, Window status bar, window resizable, window scrollable

- **Here is the code**
- ```
window.open('http://www.usc.edu', 'Sam',
'location=yes,toolbar=yes,menubar=yes,directories=yes,status=yes,resizable=y
es,scrollbars=yes,height=480,width=640', false);
```
- **Three possible ways to invoke the code above:**
 1. Triggered at page load time
 - **<body onload="[open new window code];"> </body>**
 2. Triggered by clicking on a hyperlink
 - **click me to open a new window!**
 3. Triggered by clicking on a button
 - **<input type="button" value="click me to open a new window" onclick="[open new window code];" />**

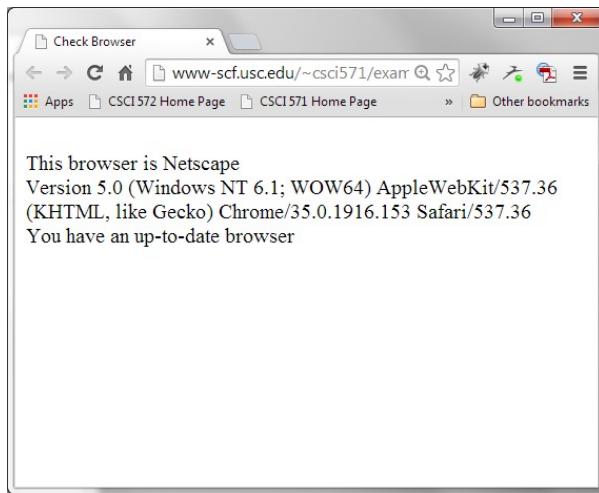
Determining the Browser

- navigator is a built-in object with properties that describe the browser
 - **navigator.appName** is a string with the browser name
 - **navigator.appVersion** is a string with the version number
 - to determine the correct version, you may need to convert from string to number; parseFloat returns a number from a string, and ignores any part of the string after the number
 - **navigator.cookieEnabled** determines whether cookies are enabled
 - **navigator.language** returns the language of the browser
 - **navigator.userAgent** returns the user-agent header sent by the browser

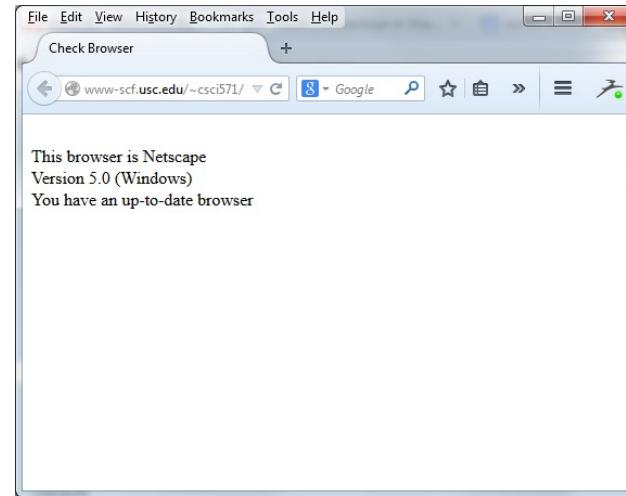
Determining the Browser Version

```
<HTML>
<HEAD>
<TITLE>Check Browser</TITLE>
</HEAD>
<BODY>
<SCRIPT language=JavaScript>
document.write("<BR> This browser is " + navigator.appName);
document.write("<BR> Version " + navigator.appVersion);
if (parseFloat(navigator.appVersion) >= 7)
    { document.write("<BR> You have an up-to-date browser"); }
</SCRIPT>
</BODY>
</HTML>
```

Example: Browser Output

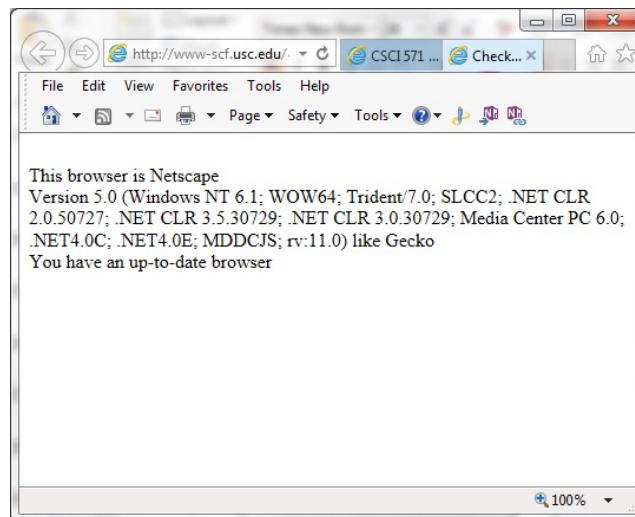


Chrome



Firefox

It is an historical anomaly that they all return Netscape; to determine the browser the appname value must be carefully analyzed, or use userAgent



Internet Explorer

Avoiding Errors

- Using newer functionalities of JavaScript, not fully supported by all browsers, will need an extra check to avoid run-time errors.
- One can refer to any name in a conditional and if it is undefined the conditional returns false

```
 newItem = something;
```
- Can also be used to test for new JavaScript APIs

produces an error if something is undefined

```
if ( something ) { newItem = something; }
```

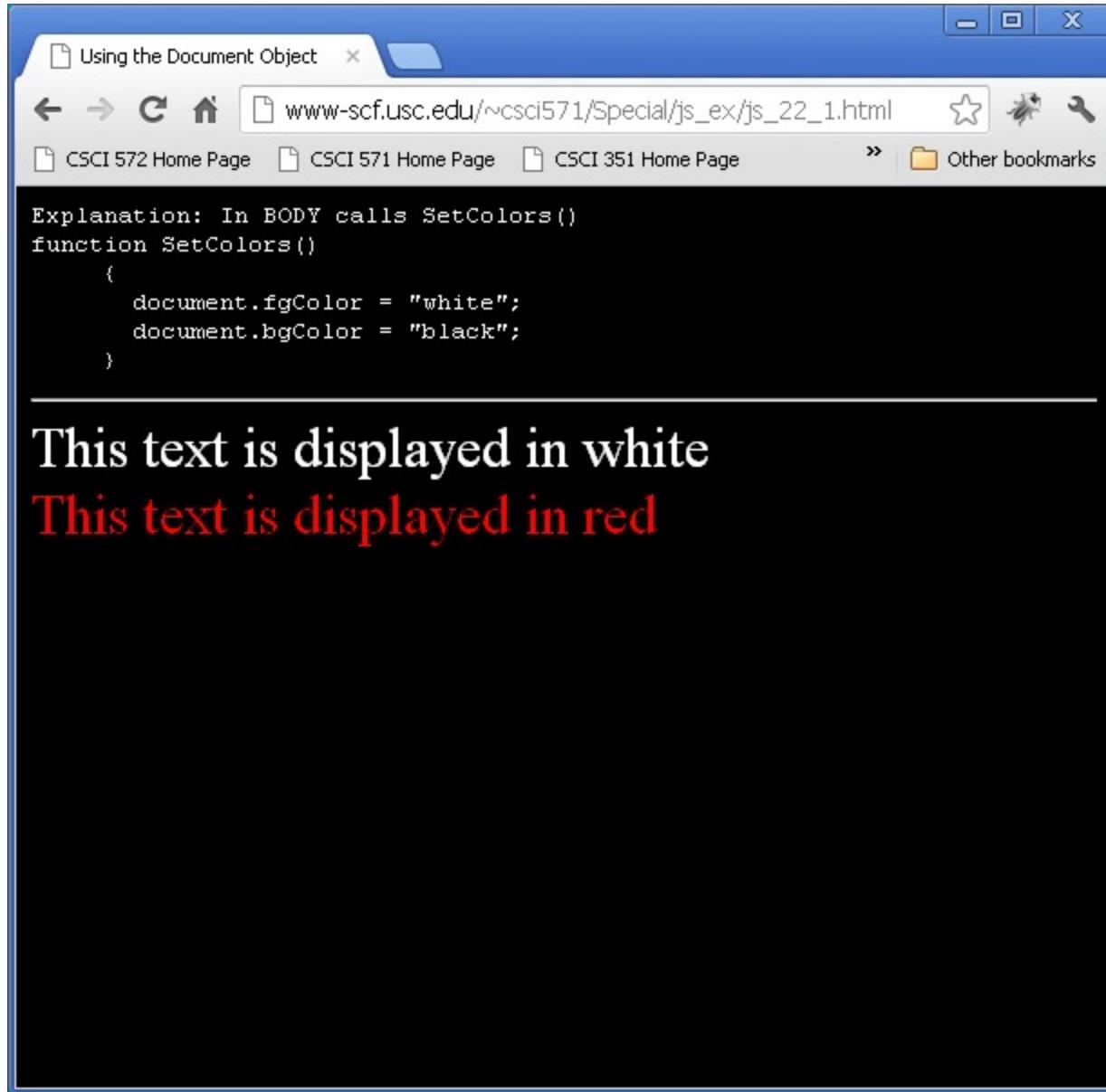
does not produce an error, but only changes newItem if something exists

The Document Object

- Each HTML document loaded into a browser window becomes a Document object.
- The Document object provides access to all HTML elements in a page, from within a script
- Some properties of the document object include:
 - `Document.anchors` - returns a collection of all anchors in the document
 - `Document.applets` - returns a collection of all applets in the document
 - `Document.body` - returns the body element of the document
 - `Document.cookie` - returns all name/value pairs of cookies in the document
 - `Document.forms` - returns a collection of all forms in the document
 - `Document.images` - returns a collection of all the images in the document
 - `Document.lastModified` - returns the date/time the document was last modified
- Some methods of the document object include:
 - `Document.close()` - closes the output stream previously opened
 - `Document.open()` - opens an output stream to collect the output from `document.write`
 - `Document.write()` - writes HTML expressions or JavaScript to a document

Setting Document Object Attributes

```
<HTML><HEAD><TITLE>Using the Document Object</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function SetColors() {
    document.fgColor = "white";
    document.bgColor = "black"; }
</SCRIPT>
</HEAD>
<BODY><SCRIPT LANGUAGE="JavaScript">
SetColors()
</SCRIPT>
<font size=+3>This text is displayed in white<br>
<font color=red>This text is displayed in
    red</font></font>
</BODY></HTML>
```



The Location Object

- A reference to the URL of the current document
- Properties: hash, host, hostname, top, status, defaultStatus, window
- No Methods and no Event Handlers
- Example

`http://www.site.com:8080/ads/newitems.html#widget1`

Property	Value
protocol	“http”
hostname	“www.site.com”
port	“8080”
host	“www.site.com:8080”
pathname	“/ads/newitems.html”
hash	“#widget1”
href	<code>http://www.big.com:8080/ads/newitems.html#widget1</code>

History Object Example

- The history object contains the URLs visited by the user within a browser window
- The history object is part of the window object

```
<HTML><HEAD><TITLE>Using the History Object</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function showHistNumber() {
    var histNumb = window.history.length;
    alert("you have already visited " + histNumb + " web pages
so far");
}
</SCRIPT></HEAD>
<BODY><FORM>
<INPUT TYPE="button" NAME="history" VALUE="myHistory"
onClick="showHistNumber()">
</FORM></BODY></HTML>
```

Browser Output

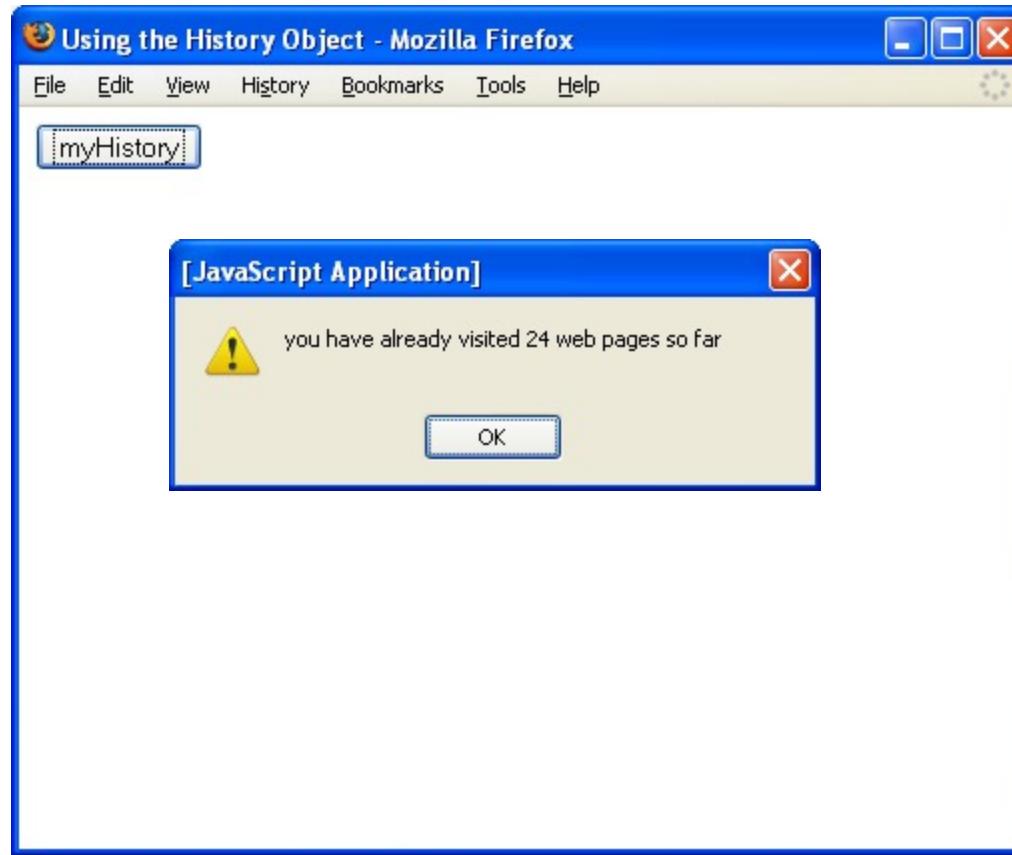


Image Object

- Each image in an HTML document has an associated JavaScript object
 - the properties of the object include
 - **width** and **height** in pixels
 - **src**, URL of the image file
 - Assigning to the src changes the image
 - **complete**, true after an image finishes loading
 - **alt**, text string returned if image is unavailable
- To manipulate an image in JavaScript one can refer to it EITHER by its NAME attribute or by a built-in array of images that is automatically created by JavaScript, e.g.
 - **document.images.name**
 - **document.images[index]** where index is either a number or a string containing the name of the image
- An image object can also be created in JavaScript

```
var myImage = new Image();
```

 - this can be used to load images into a document that are not visible in the displayed document
 - pre-fetching to make images appear immediately; coming up is an example

Example Working with Radio Buttons

```
<HTML><HEAD><TITLE>Beatle Picker</TITLE>
<SCRIPT LANGUAGE="javaScript">
function processData(form) {
for(var j = 0 ; j < form.Beatles.length ; ++j) if(form.Beatles[j].checked) {
    var RadioValue = form.Beatles[j].value; break;}
alert("You selected " + form.name + " form and Beatle " + "\n" + RadioValue
 + " and Beatle song " + document.forms[0].song.value );}
</SCRIPT></HEAD><BODY><FORM NAME="Abbey Road">
Choose your favorite Beatle: <BR>
<INPUT TYPE=radio NAME="Beatles" VALUE="Lennon" CHECKED="true">John<BR>
<INPUT TYPE=radio NAME="Beatles" VALUE="MCcartney">Paul <BR>
<INPUT TYPE=radio NAME="Beatles" VALUE="Harrison">George <BR>
<INPUT TYPE=radio NAME="Beatles" VALUE="Starr">Ringo <BR>
Enter the name of your favorite Beatles song:<BR>
<INPUT TYPE="text" NAME="song"><P>
<INPUT TYPE="button" NAME="process" VALUE="process Request...">
onClick="processData(this.form)">
</FORM></BODY></HTML>
```

See samples under "JavaScript Objects at
<http://csci571.com/examples.html#js>

Beatle Picking Form

The screenshot shows a web browser window titled "Beatle Picker". The URL in the address bar is www-scf.usc.edu/~csci571/Special/js_ex/js_22_6.html. The page content is as follows:

Forms Processing Example: Beatle Picker

Choose your favorite Beatle:

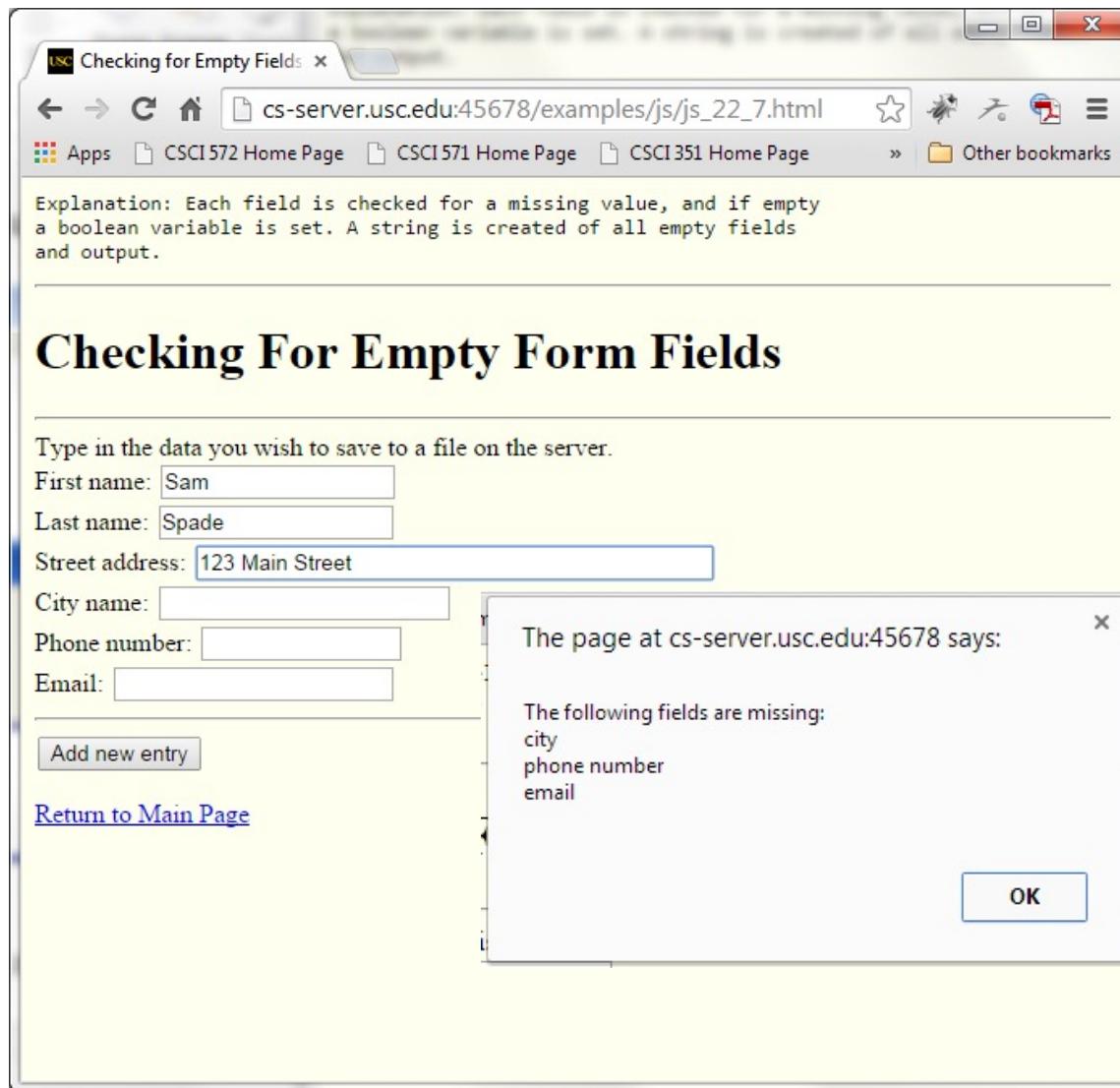
John
 Paul
 George
 Ringo

Enter the name of your favorite Beatles song:

[Return to Main Page](#)

A modal dialog box is displayed in the center of the page, titled "The page at www-scf.usc.edu says:". The message in the dialog is: "You selected Lennon and Beatle song let it be". There is an "OK" button at the bottom right of the dialog.

Checking for Empty Form Fields



Form Elements Checking (cont'd)

```
<html><head><title>Checking for Empty Fields</title><script language=JavaScript>
    function checkEmpty() { /* This function checks all of the fields of the form and
notifies the client
                which if any, form fields are empty. It returns a 1 if all the fields
are full, and a 0 otherwise. */
var firstname_filled = lastname_filled = streetaddress_filled = city_filled =
phonenumbers_filled = youremail_filled = false;
var blank="";
        if (document.myform.firstname.value != blank) firstname_filled=true;
        if (document.myform.lastname.value != blank) lastname_filled=true;
        if (document.myform.streetaddress.value != blank) streetaddress_filled=true;
        if (document.myform.city.value != blank) city_filled=true;
        if (document.myform.phonenumbers.value != blank) phonenumbers_filled=true;
        if (document.myform.youremail.value != blank) youremail_filled=true;
if ( (firstname_filled) && (lastname_filled) && (streetaddress_filled) && (city_filled) &&
(phonenumbers_filled) && (youremail_filled) )           { alert("No missing fields");
return(true); }
                else /* check which fields are missing */
var alertstring="The following fields are missing:\n";
        if (!firstname_filled) alertstring=alertstring + "first name\n";
        if (!lastname_filled) alertstring=alertstring + "last name\n";
        if (!streetaddress_filled) alertstring=alertstring + "street address\n";
        if (!city_filled) alertstring=alertstring + "city\n"; if (!phonenumbers_filled)
alertstring=alertstring + "phone number\n";
if (!youremail_filled) alertstring=alertstring + "email\n";
        alert(alertstring); return(false); } }</script></head>
```

Form Elements Checking (cont'd)

```
<BODY bgcolor="#ffffee">
<pre>Explanation: Each field is checked for a missing value, and if empty a
boolean variable is set. A string is created of all empty fields and output.
</pre><hr>
<H1><FONT color="#000000">Checking For Empty Form Fields</H1></FONT><HR>
Type in the data you wish to save to a file on the server
<FORM NAME="myform">
    First name:      <INPUT type="text" name="firstname" size=16 ><br>
    Last name:       <INPUT type="text" name="lastname" size=16 ><br>
    Street address: <INPUT type="text" name="streetaddress" size=41 ><br>
    City name:       <INPUT type="text" name="city" size=21 ><br>
    Phone number:   <INPUT type='text' name='phonenumbers' size=13 ><br>
    Email:           <INPUT type='text' name='youremail' size =20><br> <hr>
                      <INPUT type='button' value='Add new entry'
onClick="checkEmpty()" ><br>
</FORM>
</BODY>
</HTML>
```

Note: **HTML5** provides the **REQUIRED** attribute for many Elements

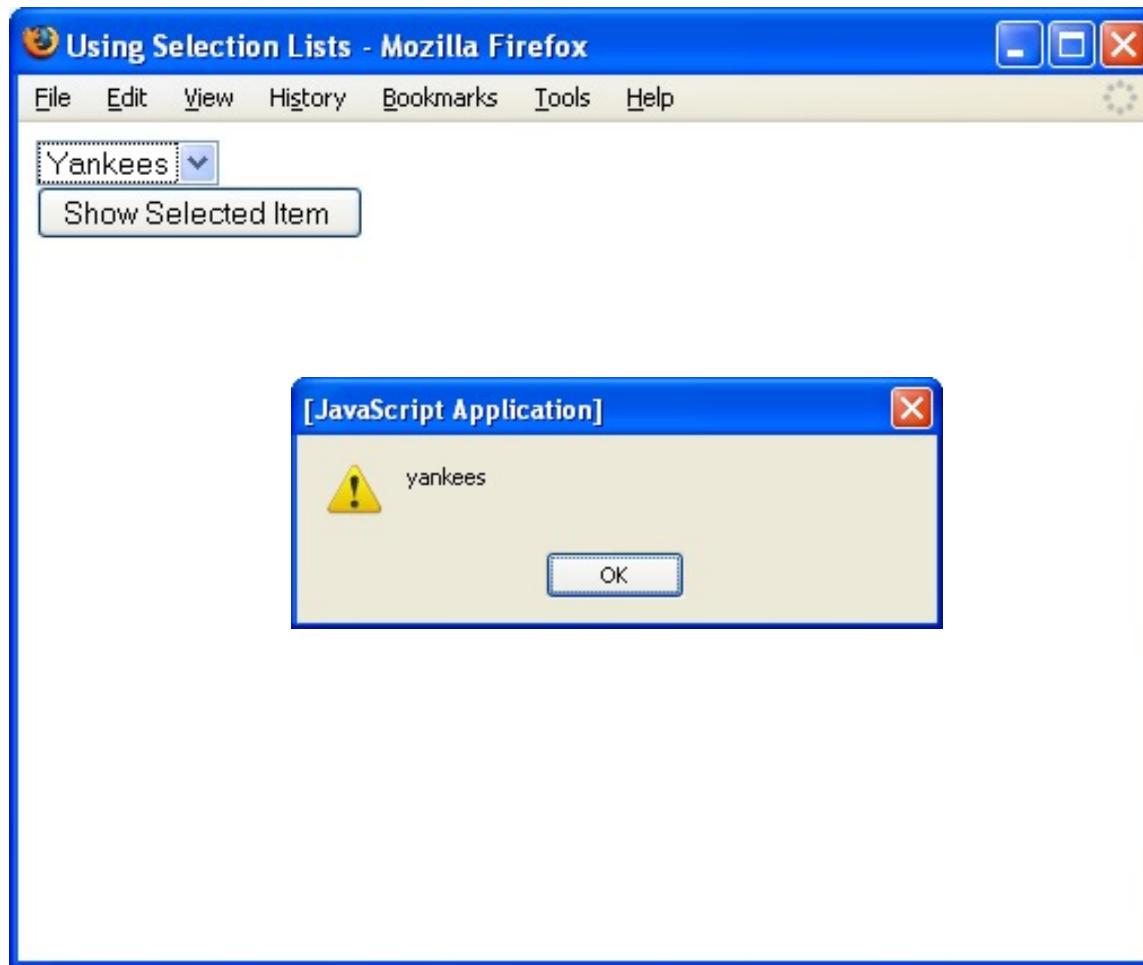
Use libraries to support older browsers. See:

<http://www.useragentman.com/blog/2010/07/27/creating-cross-browser-html5-forms-now-using-modernizr-webforms2-and-html5widgets-2/>

Example Selecting Using Selection Lists

```
<HTML><HEAD><TITLE>Using Selection Lists</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function getSelectValue(selectObject) {
    return
    selectObject.options[selectObject.selectedIndex].value
}
</SCRIPT></HEAD>
<BODY>
<FORM NAME="myform">
<SELECT NAME="teams">
<OPTION value="dodgers">Dodgers
<OPTION value="yankees">Yankees
<OPTION value="angels">Angels
</SELECT><BR>
<INPUT TYPE="button" value="Show Selected Item "
onClick="alert(getSelectValue(this.form.teams))">
</FORM>
</BODY></HTML>
```

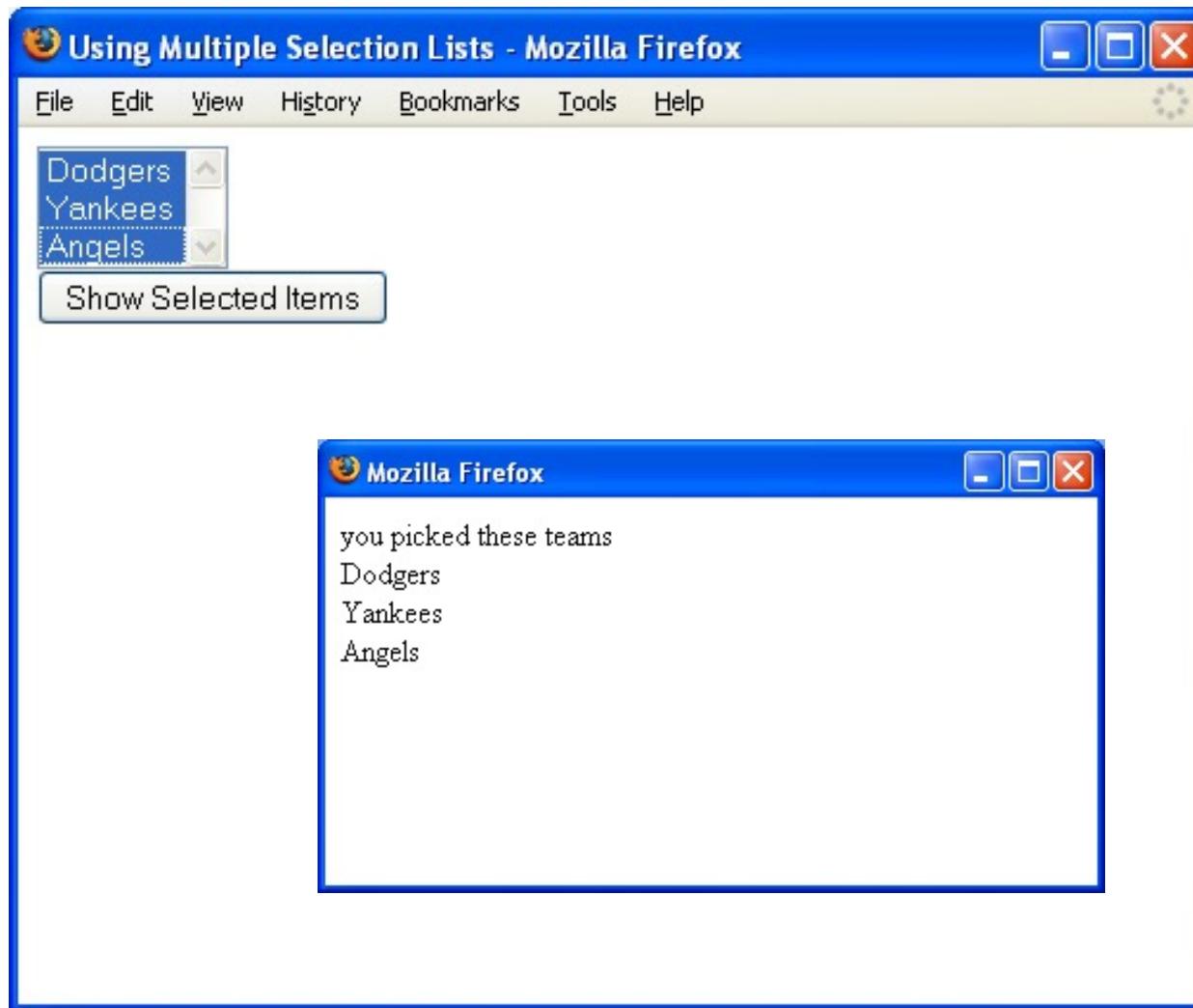
Browser Output



Example Selection Using Multi-selection Lists

```
<HTML><HEAD><TITLE>Using Multiple Selection Lists</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function displaySelectionValues(objectName) { var ans = ""
for (var i = 0; i < objectName.length; i++) {
    if (objectName.options[i].selected) {
        ans += objectName.options[i].text + "<BR>" }
myWin = window.open("", "Selections",
"height=200,width=400")
myWin.document.write("you picked these teams<BR>")
myWin.document.write(ans) }
</SCRIPT></HEAD><BODY>
<FORM NAME="myform">
<SELECT NAME="teams" size=3 multiple>
<OPTION value="dodgers">Dodgers<OPTION value="yankees">Yankees
<OPTION value="angels">Angels </SELECT><BR>
<INPUT TYPE="button" value="Show Selected Items"
onClick="displaySelectionValues(this.form.teams)">
</FORM></BODY></HTML>
```

Browser Output



JavaScript: addEventListener

- The `addEventListener()` method attaches an event handler to the specified element.
 - it does not overwrite existing event handlers.
 - You can add many event handlers to one element.
 - You can add event listeners to any DOM object not only HTML elements. i.e the window object
- **Example: Add an event listener that fires when a user clicks a button:**

```
document.getElementById("myBtn").addEventListener("click", displayDate);
```

- **Example: Use the alert function to print "Hello World!" when the user clicks on an element:**

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

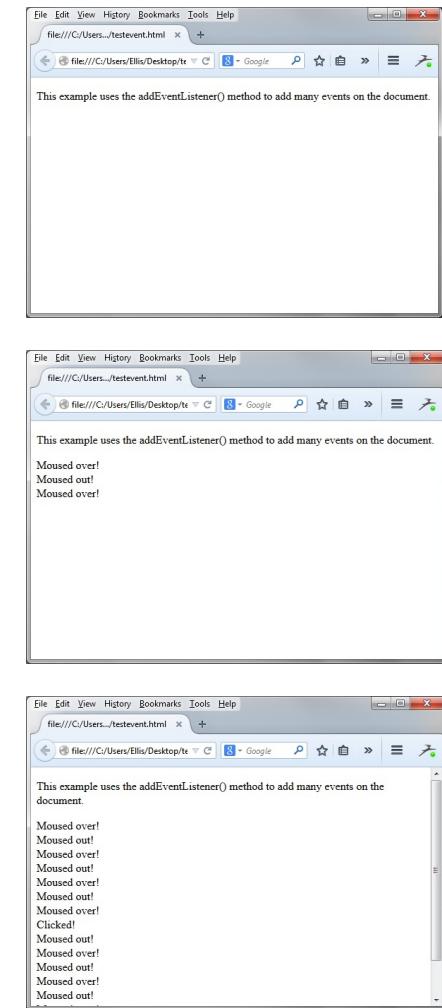
- **Example: on a click event change the associated text to Hello World; note the reference to an external function**

```
document.addEventListener("click", myFunction);
function myFunction() {
    document.getElementById("demo").innerHTML = "Hello World";
}
```

Example: Adding 3 Events to a Document

```
<!DOCTYPE html><html><body>
<p>This example uses the addEventListener() method to add many events on the document.</p>
<p id="demo"></p>
<script>
document.addEventListener("mouseover", myFunction);
document.addEventListener("click", mySecondFunction);
document.addEventListener("mouseout", myThirdFunction);
function myFunction() {
    document.getElementById("demo").innerHTML += "Moused over!<br>" }
function mySecondFunction() {
    document.getElementById("demo").innerHTML += "Clicked!<br>" }
function myThirdFunction() {
    document.getElementById("demo").innerHTML += "Moused out!<br>" }
</script></body></html>
```

See: <http://csci571.com/examples/js/Add3Events.html>



JavaScript Language Issues

Outline

- Functions as first-class objects
- Functions as methods
- Functions as constructors
- Functions have unlimited arguments
- By-value vs. By-reference
- Data Types
- Strings
- Expressions and Operator Anomalies
- Regular Expressions

Functions

- Functions are a type of object

```
function twice(x) { return x * x }
```

creates an object whose variable name is twice

- Therefore, functions can be treated as an object

- assigned to new variables

- stored in arrays

- assigned to properties of objects

- passed as arguments to functions

- Example of function names

```
var twice_again = twice;
```

```
var sum = twice(5) + twice_again(10);
```

Methods of Functions

- One can associate a function with an object
- If `func()` is a function, and `obj` is an object,

`obj.method = func;` assigns the function as an object method

- To invoke the method, do

`obj.method()`

- the `this` keyword refers to object `obj`
- Example

```
function compute_area() { return this.length *  
this.width}  
  
var rect = new Object();  
  
rect.length = 10; rect.width = 20;  
  
rect.area = compute_area; //passing the method  
  
var area = rect.area(); //invoke the method
```

Special Purpose Functions - Constructors

- Used with new to initialize fields of an object
- Example

```
function Rectangle(l, w)  
{ this.length = l; this.width = w;  
this.area = compute_area; }  
  
...  
  
var rect1 = new Rectangle(150, 100);  
  
var rect1_area = rect1.area;
```

- **this** refers to the newly created object
- Rectangle creates properties length, width, and method area

By-Value versus By-Reference

By-Value

By-Reference

copy (as in variable assignment)

value is copied reference to value is copied

pass (as in an actual parameter to a formal parameter)

a new copy of the value reference to the value is passed
is passed to the function to the function

compare (as in a conditional clause)

two different values two references are compared
are compared

Handling Data Types

	Copied by	Passed by	Compared by
numbers	value	value	value
boolean	value	value	value
string	immutable	immutable	value
object	reference	reference	reference
array	reference	reference	reference
function	reference	reference	reference

- Changes to object properties are visible outside the function
- See “JavaScript: the Definitive Guide”:
http://docstore.mik.ua/orelly/webprog/javascript/ch11_02.htm

JavaScript Strings

- Strings are **immutable**, once created they can never be changed
- You can **search** a string and **extract** substrings, but you **cannot modify** a string. And **cannot delete it!**
- “immutable” means that once you instantiate the object, you **can't change its properties.**
- So, when calling methods on a string, JavaScript will return the modified string, but it won't change the initial string
- Now this doesn't mean that you can't assign a new string object to the str variable. You just can't change the current object that str references.
- One generally builds a string through concatenation
- JavaScript runtimes include a **garbage collector**
- See: <https://javascript.info/garbage-collection>

```
var newWebPage = ""  
  
newWebPage += "<HTML><HEAD>"  
  
newWebPage += "<TITLE>A Sample Page</TITLE></HEAD>"  
  
newWebPage += "<BODY>My Home Page</BODY>"  
  
newWebPage += "</HTML>"
```

Properties of Strings

- Strings have a length property

```
"Lincoln".length // result = 7
```

```
"Four score".length //result = 10
```

```
"One\nTwo".length // result = 7
```

```
"".length // result = 0
```

- Some String methods

```
string.toLowerCase(); string.toUpperCase()
```

```
string.indexOf(searchString [, startIndex]) //returns  
index value of char within string where searchString  
begins
```

```
string.charAt(index) //returns the one char at  
position index
```

```
string.substring(indexA, indexB) //returns characters  
of string between indexA and indexB
```

Expressions and Operators

- Most of the C/C++ operators (lowest to highest order)

comma	,
assignment	= += -= *= /= %= <<= >>= >>>= &= ^= =
conditional	? :
logical or	
logical and	&&
bitwise or	
bitwise xor	^
bitwise and	&
equality	== !=
comparison	< <= > >=
bitwise shift	<< >> >>>
addition/subtraction	+ -
multiply/divide	* / %
negation/increment	! ~ - ++ --
call, data structure()	[] .

Operator Anomalies

- Implicit conversions

"a" * "b" is illegal

"3" * "5" is legal, result is 15, not "15"

- Plus used for concatenation

"a" + "b" is legal "ab"

"a" + 1 is legal "a1"

- Right associativity makes sense

i = j = k = m; is the same as i = (j = (k = m));

d = e ? f : (g ? h : (i ? j : k));

Equality Operator

- **`==` (`!=`) returns true or false**
- **Recall**
 - numbers, strings, booleans are compared by value
 - objects, arrays and functions are compared by reference
- **Implication**
 - to test if two objects have identical values, or to test if two arrays have identical values, check the fields element-by-element
- Some examples
 - "1" == 1 evaluates to true
 - true == 1 evaluates to true
 - false == 0 evaluates to true
- The identity (==>) operator behaves identically to the equality (==) operator **except** no type conversion is done, and the types must be the same to be considered equal, e.g. if `x = 5`, then
 - `x == 8` is false
 - `x == 5` is true
 - `x ==> "5"` is false
 - `x ==> 5` is true

Regular Expressions in JavaScript

- Introduced in Javascript 1.2
 - [Netscape >=4 & IE >=5.5]
- Modeled after Perl's regular expressions
- There is a regular expression object, with properties and methods
- The constructor function for a regular expression is RegExp
- A regular expression object is created using the new operator
- See
http://www.w3schools.com/jsref/jsref_obj_regex.asp

Regular Expressions in JavaScript

- Constructing a regular expression
 - `var re = /ab+c/` initializes `re` to an expr
 - `re = new RegExp("ab+c")` compiles the expr
- **test** and **exec** are methods of object `RegExp`
 - `exec` executes a search for a match in a string; it returns an array
 - `test` tests for a match and returns true/false
 - Both `exec` and `test` set properties during execution
- **match** and **search** are methods of `String`
 - `match` executes a search for a match in a string and returns an array or null
 - `search` tests for a match in a string and returns the index or -1

Regular Expression Examples

- Uses exec method to locate a string

```
<script language=javascript1.2>  
myRe = /d(b+)d/g;           //a d, 1 or more b, and a d  
myArray = myRe.exec("cdbbdbsbz");  
</script>
```

- An alternate approach

```
<script language=javascript1.2>  
myArray = /d(b+)d/g.exec("cdbbdbsbz");  
</script>
```

- An alternate approach

```
<script language=javascript1.2>  
myRe = new RegExp ("d(b+)d", g:);  
myArray = myRe.exec("cdbbdbsbz");  
</script>
```

Parenthesized Substring Matches

- Example that uses the replace method to switch the words

```
<script language=javascript1.2>
re = /(\w+) \s (\w+)/ ;
str = "John Smith";
newstr = str.replace(re, "$2, $1");
document.write(newstr)
</script>
```

This prints "Smith, John"

Regular Expression Flags

- Regular expressions have two flags that permit global and case insensitive searching
- **g** stands for **global** match; **i** stands for **ignore case**
- Syntax

```
re = /pattern/[g|i|gi]  
re = new RegExp("pattern", ['g'|'i'|'gi'])
```

- Example that creates a regular expression that looks for one or more characters followed by a space, throughout the string

```
<script language=javascript1.2>  
re = /\w+\s/g;  
str = "fee fi fo fum";  
myArray = str.match(re);  
document.write(myArray);  
</script>
```

- Output is ["fee ", "fi ", "fo "]

Example

- User enters a **phone number**, click Test, and script checks validity

```
<script language=javascript1.2>
re = /\(?(\d{3})\)?([-\/\.\.])\d{3}\1\d{4}/
function testInfo() {
RegExp.input=document.myform.Phone.value;
OK = re.exec(RegExp.input)
if (!OK)
    window.alert(RegExp.input + "isn't a phone
number with area code")
else
    window.alert("Thanks, your phone number is" +
OK[0]) }
</script>
```

Enter your phone number with area code and then click Test

```
<form name=myform><input name=Phone>
<input type=button name=PhoneButton value="Test"
onClick=testInfo(this);>
</form></ html>
```

Using Regular Expressions for Password Validation

```
<script type="text/javascript">
function checkForm(form) {
if(form.username.value == "") {
    alert("Error: Username cannot be blank!");
    form.username.focus(); return false;
}
re = /^[^\w+$/;
if(!re.test(form.username.value)) {
    alert("Error: Username must contain only letters, numbers and underscores!"); form.username.focus(); return false;
}
if(form.pwd1.value != "" && form.pwd1.value == form.pwd2.value) {
    if(form.pwd1.value.length < 6) { alert("Error: Password must contain at least six characters!");
        form.pwd1.focus(); return false;
    }
    if(form.pwd1.value == form.username.value) { alert("Error: Password must be different from Username!");
        form.pwd1.focus(); return false;
    }
re = /[0-9]/;
if(!re.test(form.pwd1.value)) { alert("Error: password must contain at least one number (0-9)!"); form.pwd1.focus(); return false;
}
re = /[a-z]/;
if(!re.test(form.pwd1.value)) { alert("Error: password must contain at least one lowercase letter (a-z)!");
    form.pwd1.focus(); return false;
}
re = /[A-Z]/;
if(!re.test(form.pwd1.value)) { alert("Error: password must contain at least one uppercase letter (A-Z)!");
    form.pwd1.focus(); return false;
}
} else { alert("Error: Please check that you've entered and confirmed your password!"); form.pwd1.focus(); return false;
}
alert("You entered a valid password: " + form.pwd1.value); return true; } </script>
```



Simplifying the Regular Expression

- The code on the previous slide is fine in that it checks everything that we wanted to check but uses a lot of code to test each requirement individually and present different error messages.
- Simpler way to accomplish it, with a single regular expression
- See: <http://www.the-art-of-web.com/javascript/validate-password/>
- Guide here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
- Consider the following:

```
<script type="text/javascript">
// at least one number, one lowercase, one uppercase letter
// at least six characters
var re = /(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}/;
var validPassword = re.test(input);
</script>
```

the thing that is new is the **notation ?=**

?=.*\d means at least one character that is a number

and

?=.*[a-z] means at least one character that is a lower case letter

and

?=.*[A-Z] means at least one character that is an upper case letter

and

.{6,} means at least 6 characters

Implementation

```
<script type="text/javascript">
function checkPassword(str) {
    var re = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])\w{6,}$/; return re.test(str); }
function checkForm(form) {
    if(form.username.value == "") { alert("Error: Username cannot be blank!");
                                    form.username.focus(); return false; }
    re = /^[\w+$/;
    if(!re.test(form.username.value)) {
        alert("Error: Username must contain only letters, numbers and underscores!");
        form.username.focus(); return false; }
    if(form.pwd1.value != "" && form.pwd1.value == form.pwd2.value) {
        if(!checkPassword(form.pwd1.value)) {
            alert("The password you have entered is not valid!");
            form.pwd1.focus(); return false; }
        } else { alert("Error: Please check that you've entered and confirmed your password!");
                  form.pwd1.focus(); return false; }
    return true; }
</script>
<form method="POST" action="form-handler.php" onsubmit="return checkForm(this);">
    <p>Username: <input type="text" name="username"></p>
    <p>Password: <input type="password" name="pwd1"></p>
    <p>Confirm Password: <input type="password" name="pwd2"></p>
    <p><input type="submit"></p> </form>
```

PassPhrases (Preview)

3. A passphrase is like a password but longer and more secure. It is an encryption key that you memorize.
- **Problem:** how do you come up with easy-to-memorize but very secure passphrases? Just thinking of one is incredibly hard, especially if your adversary really is capable of one trillion guesses per second.
- Using an entirely random sequence of characters it might be very secure, but it's also agonizing to memorize.
- **Solution:** use Diceware:
<https://en.wikipedia.org/wiki/Diceware>
- Shakespeare quotes are not good passphrases because they lack something called entropy (randomness)
- Diceware is based on a word list which contains 7,776 English words — 37 pages of it printed:
<http://world.std.com/~reinhold/dicewarewordlist.pdf>
- Roll a dice 5 times to select one word at a time for your passphrase
- If you are worried about the NSA, come up with a 7-word passphrase: there is a one in 1,719,070,799,748,422,591,028,658,176 chance that an attacker will pick your passphrase each try.
- At one trillion guesses per second — per Edward Snowden's January 2013 warning — it would take an average of 27 million years to guess this passphrase.
- This one does it: "bolt vat frisky fob land hazy rigid"
- **Recommendation:** use a password database (KeePassX) locked up with a master "diceware" passphrase:
<https://www.keepassx.org>
- See: <https://theintercept.com/2015/03/26/passphrases-can-memorize-attackers-cant-guess/>

PassPhrases (cont'd)

3. In August 2017, the original author of **NIST 2003 password guidelines**, Bill Burr, regrets making the error of recommending passwords with:

- ▶ Uppercase and lowercase.
- ▶ Letters and Numbers.
- ▶ Special characters.
- ▶ A minimum of 8 characters
- ▶ See:
<https://www.wsj.com/articles/the-man-who-wrote-those-password-rules-has-a-new-tip-n3v-r-m1-d-1502124118>
- ▶ The NIST 2017 “Digital Identity Guidelines” now recommends “diceware” passphrases:
<https://pages.nist.gov/800-63-3/>
- ▶ A comic poster is available here:
<https://xkcd.com/936/>
- ▶ A passphrase generator is available here:
<https://www.rempe.us/diceware/#eff>
- ▶ A password checker is available here:
<http://csci571.com/zxcvbn/demo/index.html>

JavaScript: `use strict`

- The idea of “strict” mode:
 - in normal JavaScript, mistyping a variable name creates a new global variable. In strict mode, this will throw an error, making it impossible to accidentally create a global variable.
 - In normal JavaScript, a developer will not receive any error feedback assigning values to non-writable properties.
 - In **strict mode**, any assignment to a non-writable property, a getter-only property, a non-existing property, a non-existing variable, or a non-existing object, will throw an error.
- Strict mode is declared by adding "`use strict`"; to the beginning of a JavaScript file, or a JavaScript function.
- Declared at the beginning of a JavaScript file, it has global scope (all code will execute in strict mode).
- Declared inside a function, it has local scope (only the code inside the function is in strict mode).

Common JavaScript Mistakes

1. **Undefined** may not be **null**

- In JavaScript something that has not been assigned to is not null, but undefined. Undefined is different from null when using `!=` but not when using the weaker `==` because JavaScript does some implicit casting in the later case
- For details see <http://weblogs.asp.net/bleroy/archive/2005/02/15/373815.aspx>

2. You cannot **overload** a function

- If you try to define two different functions with the same name but with different arguments, assuming the proper function will be called when the number of arguments match (this is overloading), then your assumption is incorrect. JavaScript will simply the latest-defined version of the function and call it;
- If a parameter is omitted it is undefined

3. **Undeclared** variables are **global**

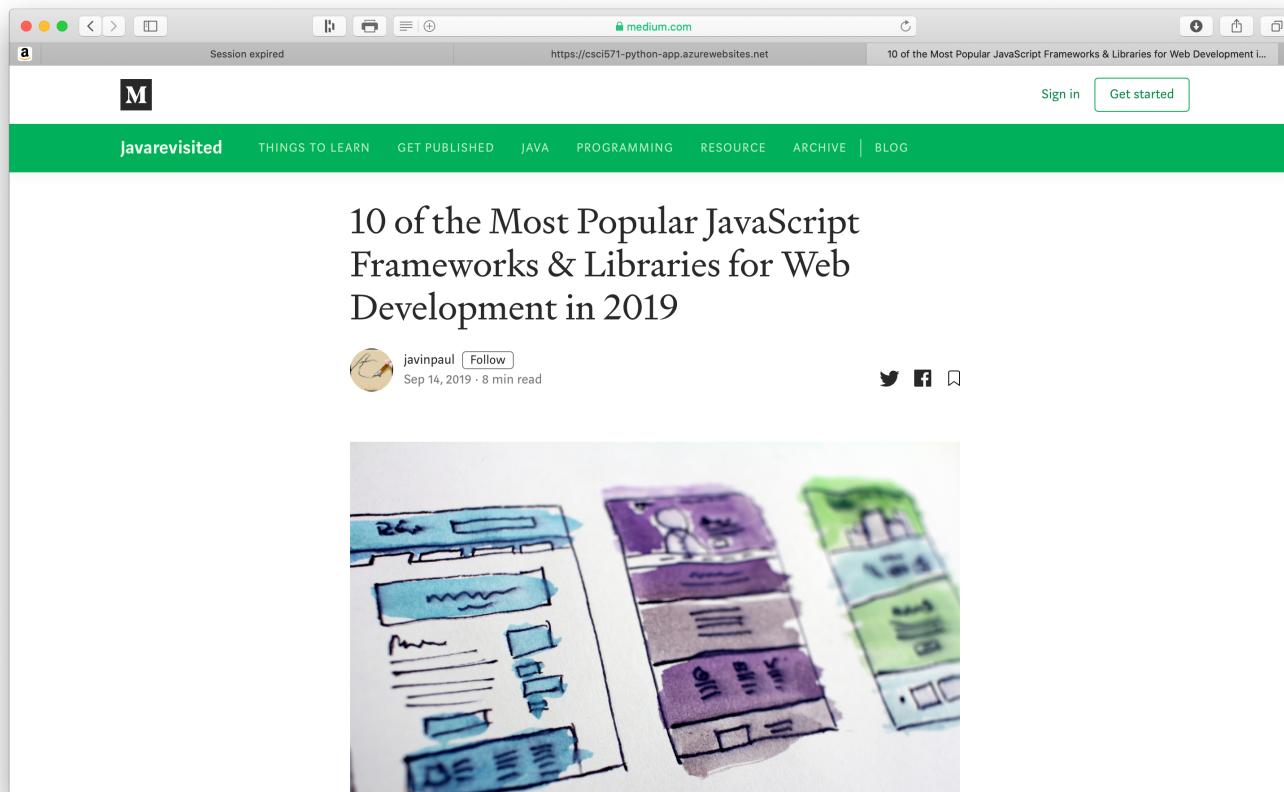
- If a variable is NOT declared using `var`, then it is global. Two variables of the same name, both undeclared will create conflicts that are hard to debug

JavaScript Libraries

- See

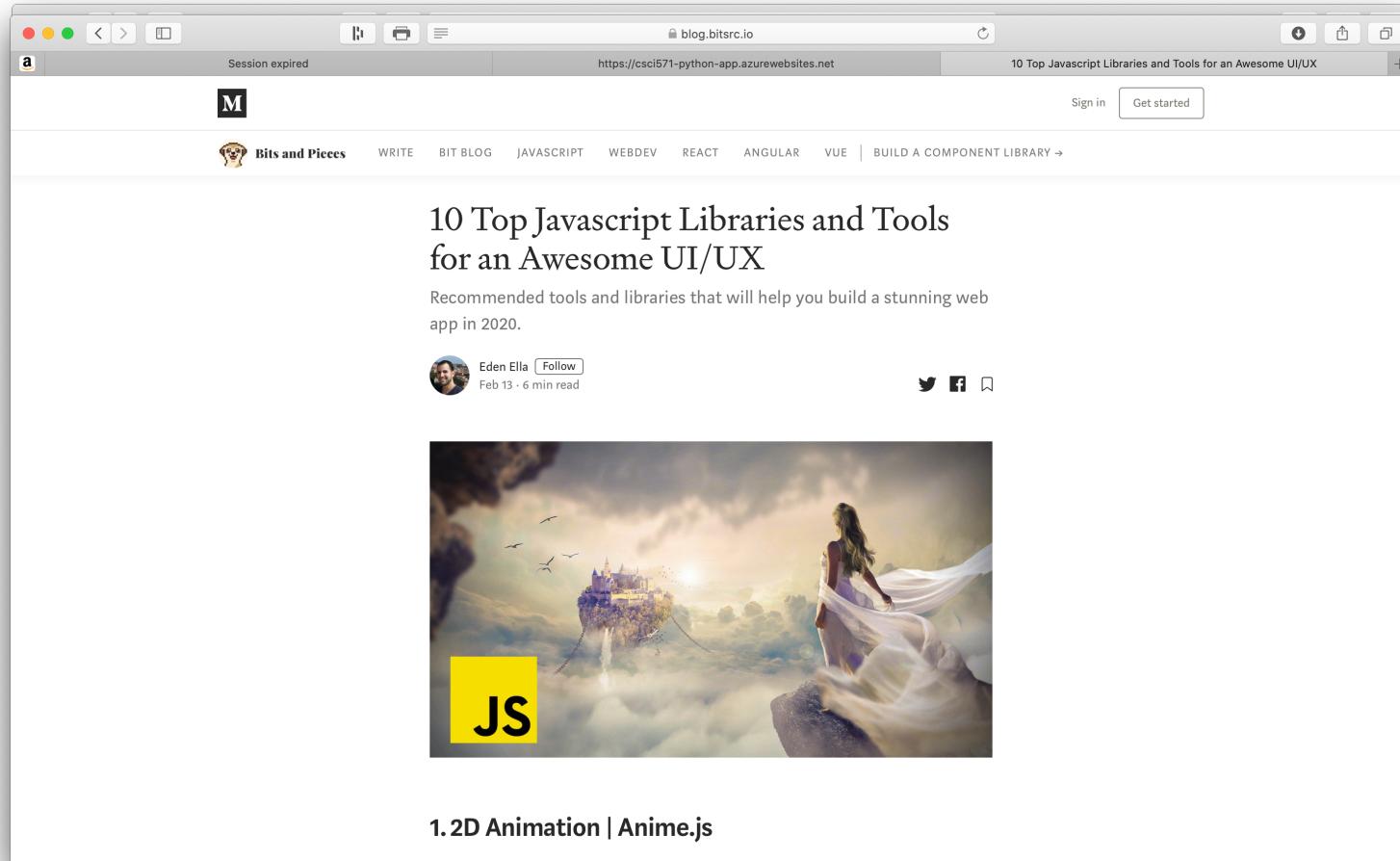
[https://medium.com/javarevisited/10-of-the-most-popular-javascript-frameworks-libraries-for-web-development-in-2019-a2c8cea68094\](https://medium.com/javarevisited/10-of-the-most-popular-javascript-frameworks-libraries-for-web-development-in-2019-a2c8cea68094)

- Each library listed includes a link to library home page



JavaScript UI/UX Libraries

- These library are good to build stunning looking UIs
- See <https://blog.bitsrc.io/10-top-javascript-libraries-and-tools-for-an-awesome-ui-ux-828a314752cc>



How do you judge a JavaScript programmer by only 5 questions?

- **1. Can you explain the difference between “call” and “apply” to me?**

The answer to this question is a bit of a factoid, so that someone can answer it doesn't give you any information, but if they cannot, it gives a truckload. Almost all JavaScript programmer that has written a library or two (which most curious ones will, after programming it a few years) will know this.

Addendum: Several people are calling #1 into question. I must be very clear here that I stand firm on this one. If you haven't used apply, you are most likely missing out on the most powerful and overlooked aspects of the language. It's also an indicator that you haven't tried your hand at building a library yet, because when building libraries, apply and call are very commonly used.

- **2. Can you explain “map” to me?**

Map is a an extremely useful functional programming concept that any computer science person will know. If someone doesn't know this, it's a sign that they lack an understanding of computer science and/or lack an understanding of the language. In addition, the explanation itself will give you a sense of how much the person knows about the language just in the way they talk. If the person does well on this question, ask about reduce as a follow up. If you do not know what map is, it means that you have done zero functional programming and you're missing out. Severely.

- **3. Can you explain “bind” to me?**

This is a really great question, because it delves into the concept of this. You can basically drill the interviewee for quite some time on this, as it is a very large subject. You'll get a good sense of a programmer by having this discussion.

- **4. Can you explain how “closures” work to me?**

This is a great question to ask programmers that claim to be experienced in general, but not with JavaScript. Closures are a general programming concept that is extraordinarily important in JavaScript. If they understand closures well, they will learn JavaScript pretty quickly.

- **5. Can you please tell me a story about a JavaScript performance problem that you've encountered and how you approached solving it?**

This will tell you a lot about how much programming a person has actually done, in their own words. A big one to keep an eye out for is that they should be praising the Google Developer tools, and not rely too much on theoretical time complexity.

Lecture

The HTTP Protocol & Networking Overview

What Does the WWW Server Do?

- Enables browser requests
- Mainly provides
 - Support for retrieving hypertext documents
 - Manages access to the Web site
 - Provides several mechanisms for executing server-side scripts
 - Common Gateway Interface (CGI)
 - Application Programmers Interface (API)
 - produces log files and usage statistics

How Does a Web Server Communicate?

- Web browsers and servers communicate using the **HyperText Transfer Protocol** (HTTP)
- HTTP is a **lightweight** protocol
 - different from the ftp protocol
 - ftp sessions are long lived and there are two connections, one for control, one for data
- Current HTTP protocol is version 1.1
- W3C updates to HTTP (last update: June 2014):
 - <http://www.w3.org/Protocols/>
- HTTP 2.0 under the IETF httpbis Working Group
 - <http://datatracker.ietf.org/wg/httpbis/charter/>
- HTTP/2 Home page:
 - <https://http2.github.io/>

HTTP History

- The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.
- The first version of HTTP, HTTP/0.9, was a simple protocol for raw data transfer across the Internet.
- **HTTP/1.0**, is defined by RFC 1945, see
 - <http://www.w3.org/Protocols/rfc1945/rfc1945>
- HTTP/1.0 allows messages to be in the format of **MIME**-like messages, containing meta-information about the data transferred and modifiers on the request/response semantics.
- **HTTP/1.1**, is defined by RFCs 7230-7237 (supersedes RFC 2616) , see
 - <http://tools.ietf.org/html/>
- HTTP/1.1 extends the protocol to handle:
 - the effects of hierarchical **proxies**
 - **caching**
 - the need for **persistent connections**
 - **virtual hosts**

HTTP History (cont'd)

- **HTTP/2** is being worked on by **IETF Working Group**:
 - <http://tools.ietf.org/wg/httpbis/>
- HTTP/2 started as a copy of **Google SPDY** ("SPeeDY").
- HTTP/2 designed to speed up websites far larger than 10 years ago, using hundreds of requests/connections.
- One major feature of HTTP/2 is **header compression**:
 - <https://httpwg.org/specs/rfc7541.html>
- Google has dropped SPDY from Chrome and adopted HTTP/2:
 - <http://techcrunch.com/2015/02/09/google-starts-fading-out-spdy-support-in-favor-of-http2-standard/>
- See also:
 - <https://en.wikipedia.org/wiki/HTTP/2>
- Seen RFC 7540 (HTTP/2) & 7541 (HPACK):
 - <https://httpwg.org/specs/rfc7540.html>
- Dozens of implementations already available, including **Apache (2.4+)**, **Apache-Tomcat (8.5+)**, **Nginx (1.9.5+)**, etc.:
 - <https://github.com/http2/http2-spec/wiki/Implementations>
- **HTTP/3** is already being worked on!

MIME MEDIA TYPES

- HTTP tags all data that it sends with its **MIME type**
- HTTP sends the MIME type of the file using the line
Content-Type: mime type header

- For example, here are 2 MIME type messages

Content-type: image/jpeg

Content-length: 1598

- Some important MIME types are

- text/plain, text/html
- image/gif, image/jpeg
- audio/basic, audio/wav, audio/x-pn-realaudio
- model/vrml
- video/mpeg, video/quicktime, video/vnd.rn-realmedia, video/x-ms-wmv
- application/*, application-specific data that does not fall under any other MIME category, e.g., application/vnd.ms-powerpoint

Multipurpose Internet Mail Extensions

- MIME is an Internet standard for **electronic mail**
 - Traditional e-mail was limited to ASCII text, limited line length, and limited size
- MIME has extended Internet e-mail to include
 - Unlimited text line and message length
 - Messages with multiple body parts or objects enclosed
 - Messages that point to files on another server and are automatically retrievable
 - International character sets in addition to US-ASCII
 - Formatted text including multiple font styles
 - Images
 - Video clips
 - Audio messages
 - Application-specific binary data
- It was formalized in RFC 2046

Facts About MIME

- MIME converts data that uses all **eight bits** into **7-bit ASCII**, sends it, and reconverts it at the other end. See:

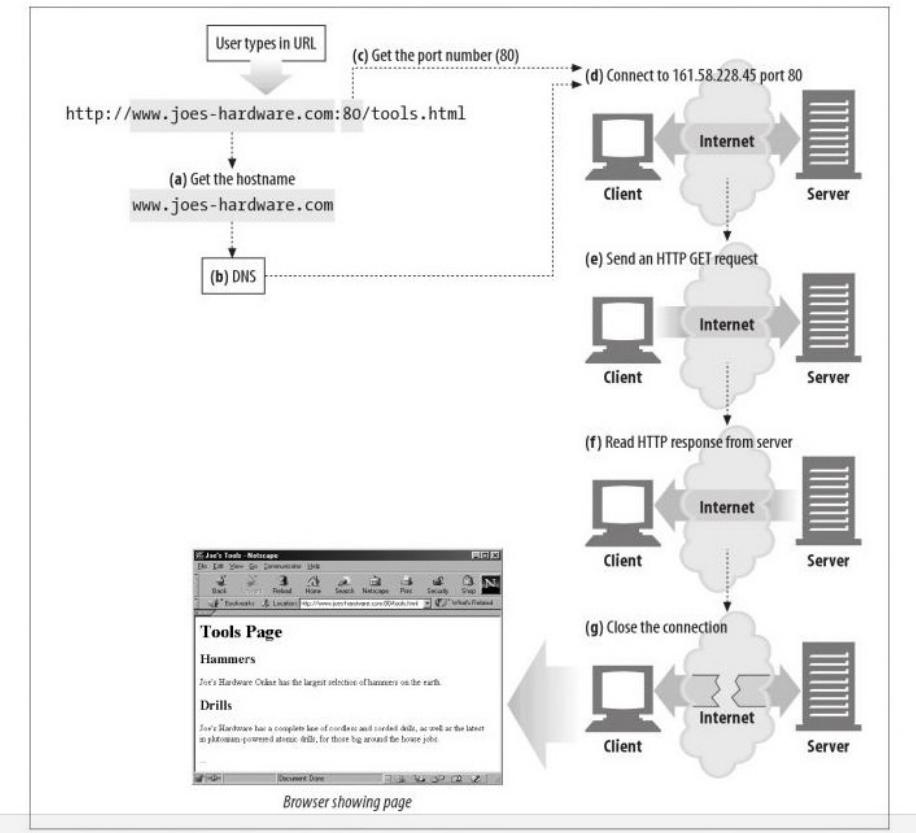
<https://tools.ietf.org/html/rfc1652>

- MIME headers at the front of the file define the type of data the message includes, e.g., here are a set of MIME types describing an attachment at an ftp site

```
Content-type: Message/External-Body  
name="classnotes.ps"  
site="ftp.usc.edu"  
access-type=anon-ftp  
directory="pub/cs665"  
mode="image"  
permission="read"  
expiration="Wed, 15 Mar 2009 07:00:00 -0400 (PST)"
```

Description of a Browser Server Interaction

- (a) The browser extracts the server's hostname from the URL.
- (b) The browser converts the server's hostname into the server's IP address.
- (c) The browser extracts the port number (if any) from the URL.
- (d) The browser establishes a TCP connection with the web server.
- (e) The browser sends an HTTP request message to the server.
- (f) The server sends an HTTP response back to the browser.
- (g) The connection is closed, and the browser displays the document.



An HTTP 1.0 “default” Scenario

- Communication takes place over a TCP/IP connection, generally on port 80

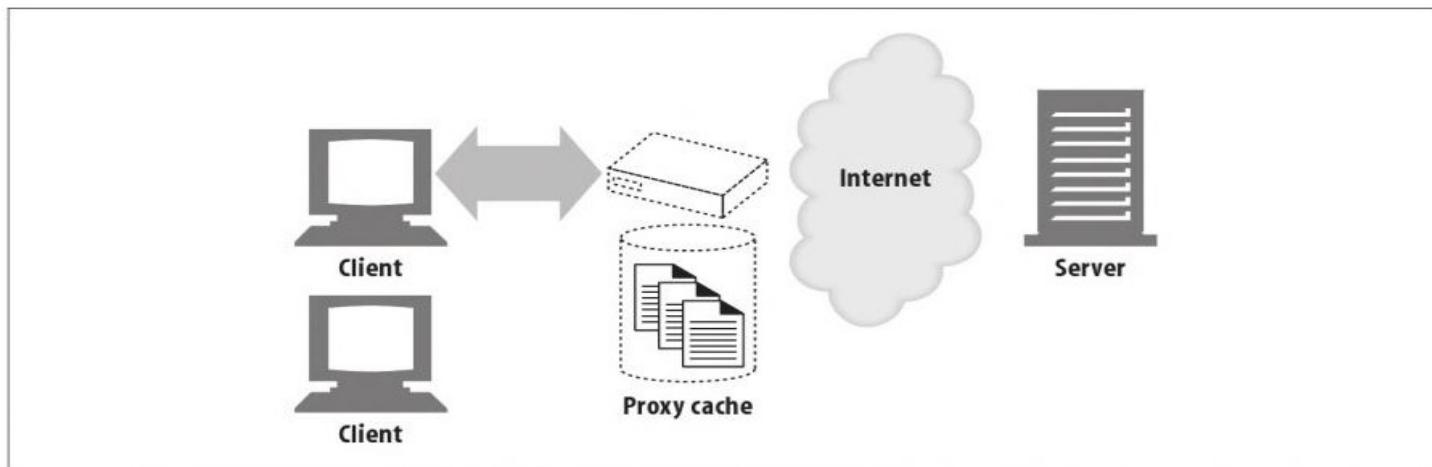
Client action	Server response
1. Client opens a connection	Server responds with an acknowledgment
2. Client sends HTTP request for HTML document	Server responds with the document and closes the connection
3. Client parses the HTML document and opens a new connection; it sends a request for an image	Server responds with the inlined image and closes the connection
4. Client opens a connection and sends another request for another image	Server sends the inlined image and closes the connection

A More Complicated HTTP Scenario

- Actually, communication between a browser and a web server can be much more complicated; communication can go between one or more **intermediaries**.
- There are three common forms of intermediary: proxy, gateway, and tunnel.
 - A **proxy** is a forwarding agent, receiving requests for a URI in its absolute form, rewriting all or part of the message, and forwarding the reformatted request toward the server identified by the URI.
 - A **gateway** is a receiving agent, acting as a layer above some other server(s) and, if necessary, translating the requests to the underlying server's protocol.
 - A **tunnel** acts as a relay point between two connections without changing the messages; tunnels are used when the communication needs to pass through an intermediary (such as a firewall) even when the intermediary cannot understand the contents of the messages.

Caching Proxies

A web cache or caching proxy is a special type of HTTP proxy server that keep copies of popular documents that pass through the proxy (**"forward" proxy**). The next client requesting the same document can be served from the cache's personal copy.

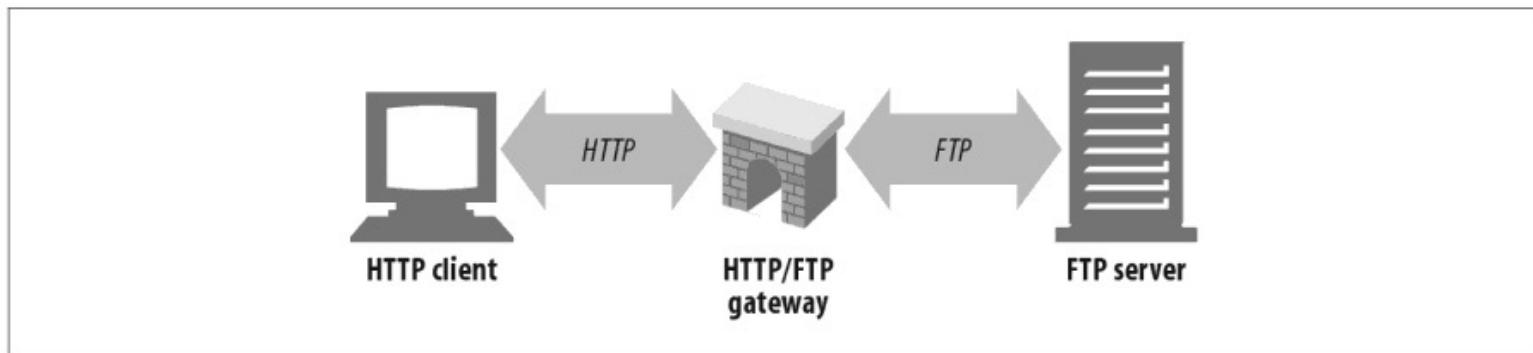


Gateways

Gateways are special servers that act as intermediaries for other servers.

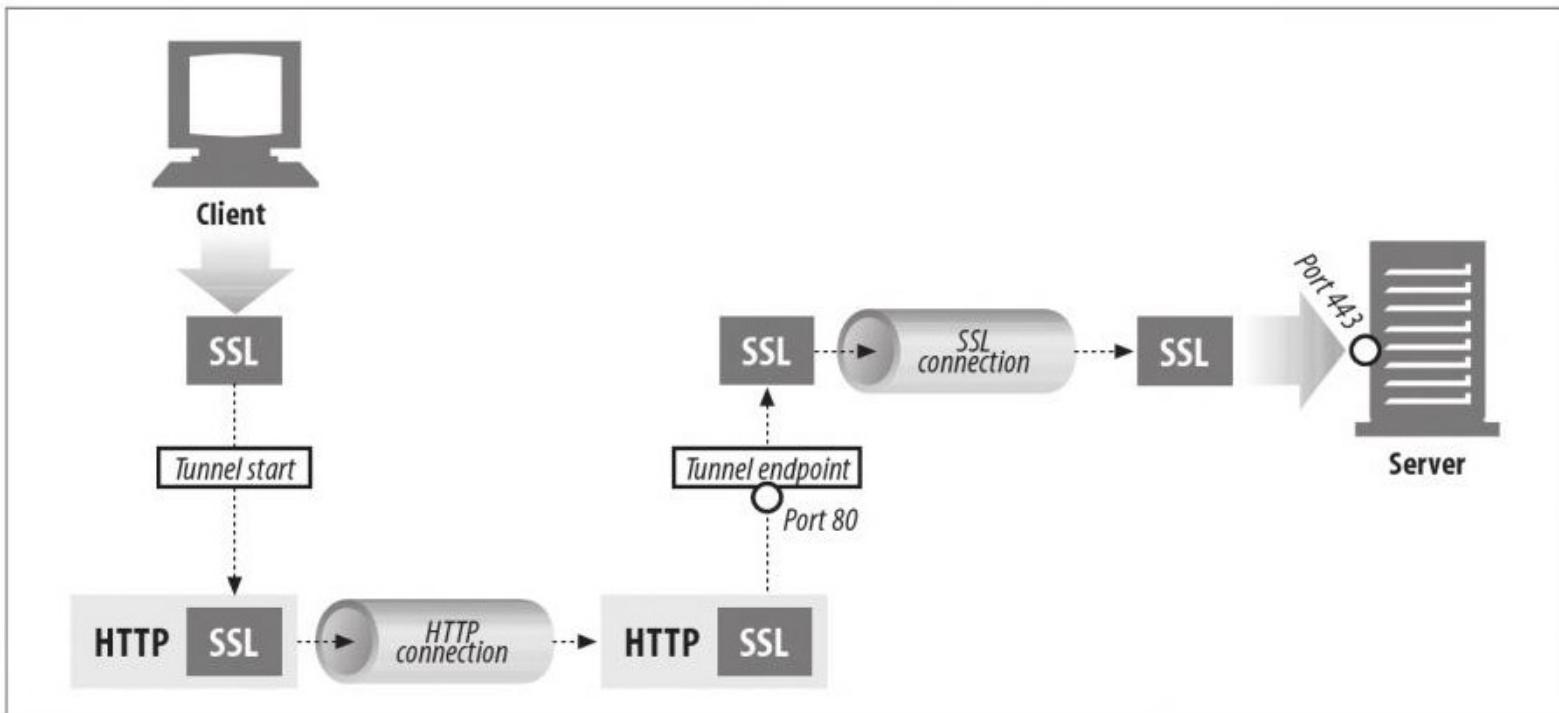
They are often used to **convert HTTP traffic to another protocol**. A gateway always receives requests as if it was the origin server for the resource. The client may not be aware it is communicating with a gateway.

For example, an HTTP/FTP gateway receives requests for FTP URIs via HTTP requests but fetches the documents using the FTP protocol. The resulting document is packed into an HTTP message and sent to the client.



Tunnels

Tunnels are HTTP applications that, after setup, blindly relay raw data between two connections. HTTP tunnels are often used to transport non-HTTP data over one or more HTTP connections, without looking at the data. A **VPN** is an example of a tunnel.



The Most General HTTP Scenario

Communication between browser and server should be regarded as a **request chain** goes left to right ---->

UA ---v--- **A** ---v--- **B** ---v--- **C** ---v--- **O**

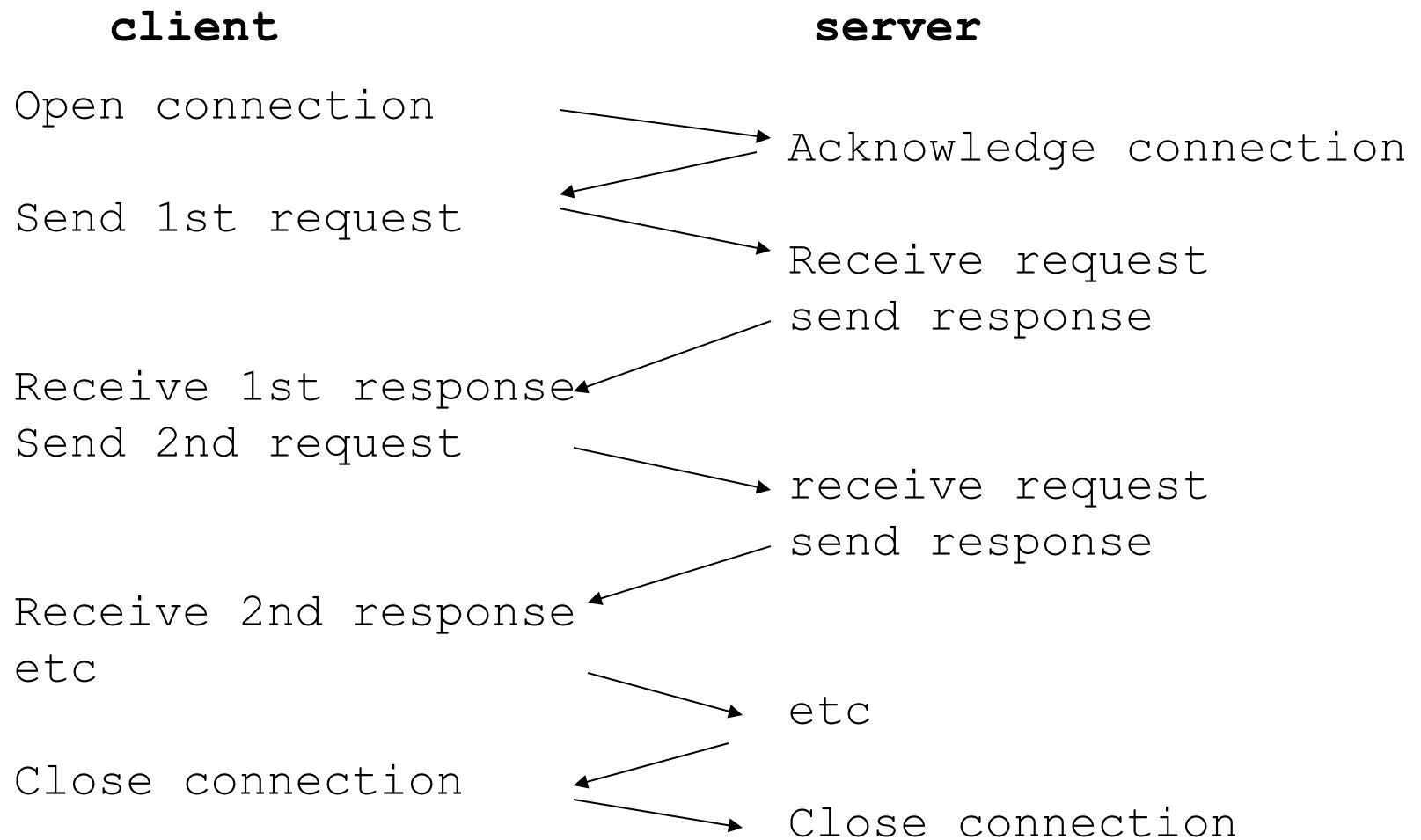
<----- and a response chain goes right to left

- **A**, **B**, and **C** are three intermediaries between the user agent and origin server. A request or response message that travels the whole chain will pass through four separate connections.
- **UA** stands for User Agent, typically a browser
- **O** stands for the origin server; the server that actually delivers the document

Persistent Connections

- In the original HTTP protocol, each request was made over a new connection
 - so, an HTML page with n distinct graphic elements produced $n+1$ requests
- TCP uses a **three-way handshake** when establishing a connection, so there is significant latency in establishing a connection
 - client sends SYN, server replies ACK/SYN, client responds with ACK
- HTTP 1.0 introduced a **keep-alive** feature
 - the connection between client and server is maintained for a period of time allowing for multiple requests and responses
 - a.k.a. **Persistent connection**

HTTP/1.0 Keep Alive Connections



HTTP/1.1 Keep Alive Extensions

- **Persistent connections** are now the default
- Request Header to set timeout (in sec.) and max. Number of requests, before closing:
Keep-Alive: timeout=5, max=1000
- Client and server must explicitly say they do NOT want persistence using the header
Connection: close
- HTTP permits multiple connections in parallel
 1. client requests a page and server responds
 2. client parses page and initiates 3 new connections, each requesting a different image
- Above scheme is NOT always faster, as multiple connections may compete for available bandwidth
- Generally, browsers severely limit multiple connections and servers do as well

Example of a GET Request

- Clicking on a link in a web page or entering a URL in the address field of the browser causes the browser to issue a GET request, e.g.
- Suppose the user clicks on the link below:
click here
- The request from the client may contain the following lines

```
GET /html/file.html HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:15.0) Gecko/20100101
           Firefox/15.0.1
Referer: http://www.usc.edu/html/prevfile.html
If-Modified-Since: Wed, 11 Feb 2009 13:14:15 GMT
{there is a blank line here which terminates the input}
```

Response of the Server to GET

- In response to the previous client request, the server responds with the following

HTTP/1.1 200 OK

Date: Monday, 29-May-09 12:02:12 GMT

Server: Apache/2.0

MIME-version: 1.0

Content-Type: text/html

Last-modified: Sun, 28-May-09 15:36:13 GMT

Content-Length: 145

{a blank line goes here }

{the contents of file.html goes here }

Client HTTP Requests

- The general form of an HTTP request has four fields:

HTTP_method, identifier, HTTP_version, Body

- **HTTP_Method** says what is to be done to the object specified in the URL; some possibilities include GET, HEAD, and POST
- **identifier** is the URL of the resource or the body
- **HTTP_version** is the current HTTP version, e.g. HTTP/1.1
- **Body** is optional text

HTTP Request Methods

- Most common HTTP request methods are
 - **GET**, retrieve whatever information is identified by the request URL
 - **HEAD**, identical to GET, except the server does not return the body in the response
 - **POST**, instructs the server that the request includes a block of data in the message body, which is typically used as input to a server-side application
 - **PUT**, used to modify existing resources or create new ones, contained in the message body
 - **DELETE**, used to remove existing resources
 - **TRACE**, traces the requests in a chain of web proxy servers; used primarily for diagnostics
 - **OPTIONS**, allows requests for info about the server's capabilities

HTTP Headers

- HTTP/1.1 divides headers into four categories:
 - **general**, present in requests or responses
 - **request**, present only in requests
 - **response**, present only in responses
 - **entity**, describe the content of a body
 - extension, new headers not already defined
- Each header consists of a name followed by a colon, followed by the value of the field, e.g.

Date: Tue, 3 Oct 2009 02:16:03 GMT

Content-length: 12345

Content-type: image/gif

Accept: image/gif, image/jpeg, text/html

Examples of HTTP Headers - Request

- **Accept: text/html, image/***
indicates what media types are acceptable
- **Accept-Charset: iso-8859-5**
indicates acceptable character sets. By default all are acceptable
- **Accept-Encoding: compress, gzip**
indicates acceptable encodings
- **Accept-Language: en, fr=0.5**
indicates language preferences, English preferred, but French also accepted
- **Authorization:**
used to pass user's credentials to the server

Examples of HTTP Headers - Request

- **From: name@site.com**
requesting user's email address, rarely present
- **Host: www.usc.edu:8080**
hostname and port of the requesting URL
- **Referer: http://www.usc.edu/index.html**
the URL of the document that contains the reference to the requested URL
- **User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:18.0) Gecko/20100101 Firefox/18.0**
reports the client software name and version and possibly platform, see
<http://www.javascriptkit.com/javatutors/navigator.shtml>

Byte Range Headers

- Requests

- **If-Range: "entity-tag"**

used with byte range requests to guarantee that any new byte range responses are generated from the same source object. The entity-tag is quoted

- **Range: bytes=0-512, 2048-4096**

used to request a byte range

- Responses

- **Accept-ranges: bytes**

indicates the server can respond to range requests

- Entity

- **Content-Range: 0-399/2000**

response to byte range request giving the byte ranges actually returned, e.g. the first 400 bytes of a 2000 byte document

Examples of HTTP Headers - Response

- **Age: 1246**
age in seconds since response was generated
- **Location: http://www.myco.com/page.html**
indicates that re-direction is desired
- **Public: GET, HEAD, POST, OPTIONS, PUT**
methods supported by this web server
- **Server: Apache/1.3.1**
identifies the server
- **WWW-AUTHENTICATE:**
sent with 401 Unauthorized status code, it includes authorization parameters
- **Retry-after: 240**
used with Service Unavailable status, indicates requested data will be available in 4 minutes

Examples of HTTP Headers - Response

- A URL may point to a document with multiple representations: languages, formats (html, pdf), or html features based upon user-agent
- if a French version is requested and cached, then a new request may fail to retrieve the English version
- HTTP/1.1 introduces Vary: accept-language, user-agent the header specifies acceptable languages and browsers,
- **e.g., the request is**

GET http://www.myco.com/ HTTP/1.1

User-agent: Mozilla/4.5

Accept-language: en

- **the response is**

HTTP/1.1 200 OK

Vary: Accept-language

Content-type: text/html

Content-language: en

The proxy must store the fact that this doc has variants and when requested, get the proper variant

Examples of HTTP Headers - Response

- Warning: 10 proxy-id “Revalidation failed” messages indicating status information of the resource; HTTP/1.1 defines the following status codes

Code Meaning

10	Response is stale
11	Revalidation failed
12	Disconnected operation
13	Heuristic expiration
14	Transformation applied
99	Miscellaneous warning

Entity Tags

- An **ETag** or **entity tag**.
 - one of several mechanisms that HTTP provides for web cache validation, and which allows a client to make conditional requests.
 - This allows caches to be more efficient, and saves bandwidth, as a web server does not need to send a full response if the content has not changed.
- An ETag is an opaque identifier assigned by a web server to a specific version of a resource found at a URL.
 - If the resource content at that URL ever changes, a new and different ETag is assigned.
 - ETags are similar to **fingerprints**, and they can be quickly compared to determine if two versions of a resource are the same or not.
- An ETag is a serial number or a checksum that uniquely identifies the file
 - caches use the **If-None-Match** condition header to get a new copy if the entity tag has changed
 - if the tags match, then a **304 Not Modified** is returned

Examples of HTTP Headers - Entity

- **Allow:** `GET, HEAD, PUT`
lists methods supported by the URL
- **Content-Base:** `http://www.usc.edu/somedir`
all relative references are taken wrt the base
- **Content-Encoding:** `gzip`
indicates the encoding of the entity body;
content-type indicates the media after encoding
- **Content-Language:** `en`
identifies the language of the entity
- **Content-Length:** `7890`
specifies the length of the entity in bytes
- **Content-Location:** `http://www.usc.edu/myfile.htm`
specifies the URL of the accessed resource

Examples of HTTP Headers - Entity

- **Content-MD5: base-64 encoded MD5 signature**
contains the MD5 signature of the body as created by the web server
- **Content-type:text/html**
indicates the MIME type of the object
- **Etag: "7776cdb01f44354af8bfa40c56eebcb1378975"**
specifies the entity tag for the object, which can be used for re-validation; tags are unique ids determined by the server; this line is normally sent as a response
- **Expires: Wed, 30 Dec 2002 03:43:21 GMT**
specifies the expiration date/time of the object; a cached copy should not be used beyond; Expires 0/now is immediate
- **Last-Modified: Wed, 30 Dec 2002 01:20:34 GMT**
specifies the creation or last modification time of the object on the web server

HTTP Status Codes - *Informational*

- After receiving and interpreting a request message, a server responds with an HTTP response message.
- Syntax of response is

Status-Line

* (general-header | response-header | entity-header) CRLF [message-body]

where the Status line is composed of

Status-Line = HTTP-Version Status-Code Reason-Phrase CRLF

HTTP Status Codes - *Informational*

Code meaning

- | | |
|-----|--|
| 100 | Continue, the client may continue with its request; used for a PUT before a large document is sent |
| 101 | Switching Protocols, switching either the version or the actual protocol |

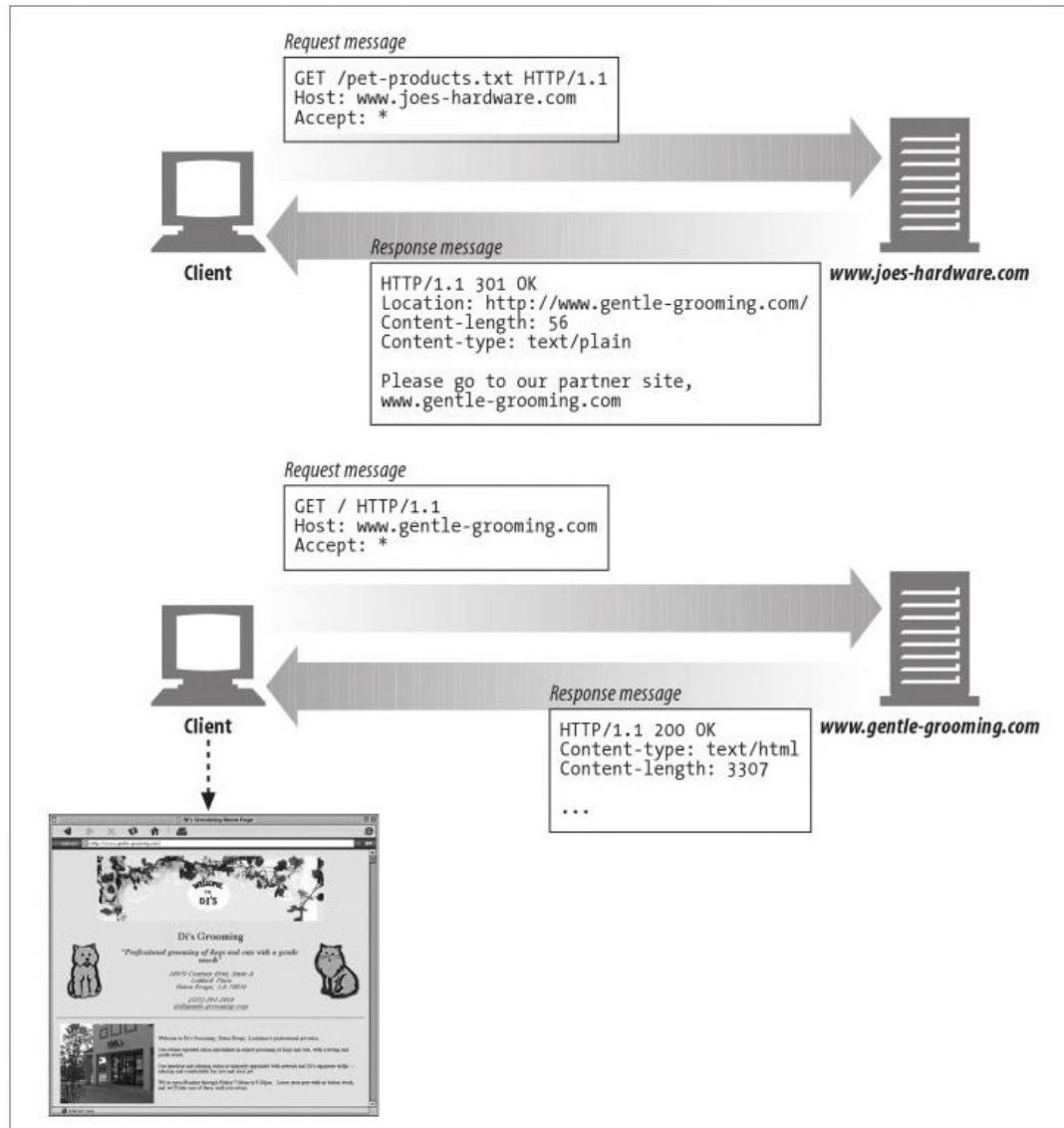
HTTP Status Codes - *Successful*

Code	meaning
200	OK, request succeeded
201	Created, result is newly created
202	Accepted, the resource will be created later
203	Non-authoritative information, info returned is from a cached copy and may be wrong
204	No content, response is intentionally blank, so client should not change the page
205	Reset Content, notifies the client to reset the current document, e.g. clear a form field
206	Partial content, e.g. a byte range response

HTTP Status Codes - *Redirection*

Code	meaning
300	Multiple choices, the document has multiple representations
301	Moved permanently, new location is specified in Location: header
302	Moved temporarily: similar to above
303	See Other: used to automatically redirect the client to a different URL
304	Not Modified: the client or proxy copy is still up-to-date
305	Use Proxy: make the request via the proxy
306	Proxy Redirection: a proposed extension to HTTP/1.1 still not specified
307	Temporary Redirect

Redirection Example



HTTP Status Codes - *Client Error*

Code	meaning
400	Bad request, server could not understand
401	unauthorized, authorization challenge
402	Payment Required, reserved for future use
403	forbidden, server refuses to fulfill request; e.g., check protections
404	Not found, document does not exist
405	Method not allowed, request method is not allowed for this URL
406	Not Acceptable, none of the available representations are acceptable to the client
407	Proxy Authentication Required, authentication is being challenged
408	Request Timeout, client did not send a request within a time given by server

HTTP Status Codes - *Client Error (cont'd)*

Code	meaning
409	conflict, the requested action cannot be performed
410	Gone, requested resource is no longer available
411	Length Required, Content-length header is missing
412	Precondition failed, a precondition for the request has failed, so request is canceled
413	request entity too large, entity is too large for server
414	Request URI too large, the request URL is too large
415	Unsupported media type, request entity is of an unsupported type

HTTP Status Codes - *Server Error*

Code	meaning
500	Internal server error, generic error code for the server
501	Not implemented, request could not be serviced because server does not support it
502	Bad gateway, intermediate proxy server received a bad response
503	Service unavailable, due to high load or maintenance on the server
504	Gateway timeout, intermediate proxy server timed out waiting for response from another server
505	HTTP version not supported

HTTP Authentication

- The web server can maintain secure directories and request authentication when someone tries to access them
- Procedure
 - web server receives a request without proper authorization
 - web server responds with 401 Authentication Required
 - client prompts for username and password and returns the information to the web server
- we will show how to cause an authentication request when we discuss the web server features



Example HTTP Basic Authentication

- Client makes a request on a secure page**

GET /secure/mypage.html HTTP/1.0

- Server responds with**

HTTP/1.0 401 Unauthorized

Server: Apache 2.0

Date: Wed, 23 Dec 2002 15:42:00 GMT

WWW-Authenticate: Basic realm="notes server"

- Client prompts for username and password and uses base-64 printable encoding; it re-issues its request, e.g.**

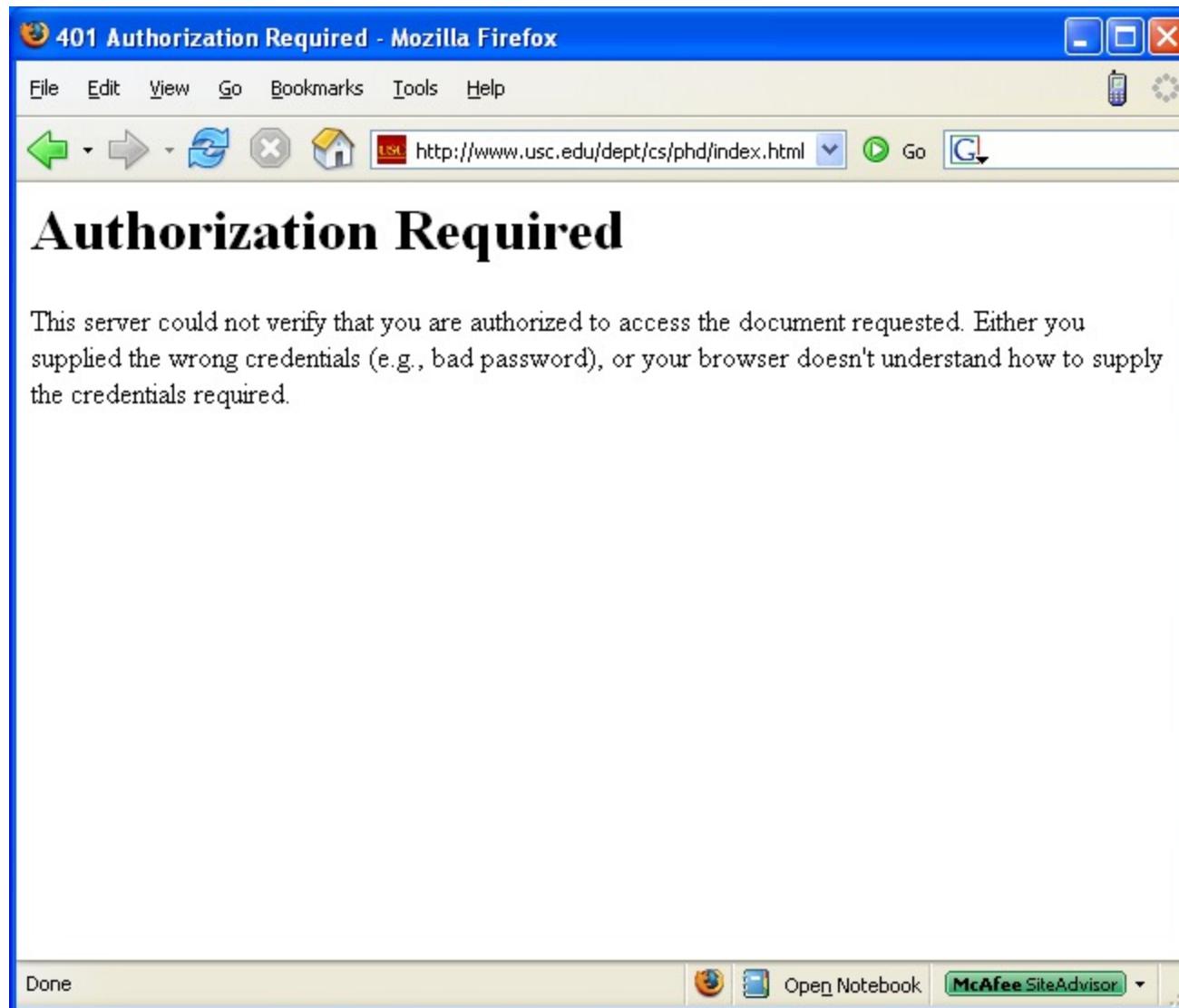
GET /secure/mypage.html HTTP/1.0

User-agent: Mozilla/4.5

Accept: text/html, image/gif, image/jpeg

Authorization: Basic MQDadmverWPUsvd=

Authorization Denied



META HTTP-EQUIV

- This is a mechanism for authors of HTML documents to set HTTP headers, in particular HTTP responses
- two common uses are
 - to set the **expiration time** of a document
 - to cause a **refresh** of a document
- E.g., here is an HTTP response header which causes the current page to be replaced by another page in 5 seconds

Refresh: 5; http://xyz.com/htdocs/mynewpage.html
this is equivalent to the following HTML

```
<META HTTP-EQUIV="Refresh" CONTENT=5,  
http://csci571.com/index.html">  
</HEAD><BODY> This page will be replaced by the  
csci571 home page in 5 seconds.
```

X-Frame-Options: SAMEORIGIN

Provides **Clickjacking protection**. Values: *deny* - no rendering within a frame, *sameorigin* - no rendering if origin mismatch, *allow-from: DOMAIN* - allow rendering if framed by frame loaded from *DOMAIN*

The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a **<frame>** or **<iframe>**. Sites can use this to avoid clickjacking attacks, by **ensuring that their content is not embedded** into other sites; options include:

DENY

The page cannot be displayed in a frame, regardless of the site attempting to do so.

SAMEORIGIN

The page can only be displayed in a frame on the same origin as the page itself.

ALLOW-FROM *uri*

The page can only be displayed in a frame on the specified origin.

See:

<https://tools.ietf.org/html/rfc7034>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/X-Frame-Options>

HTTP Strict-Transport-Security (HSTS)

HSTS is a security feature that lets a web site tell browsers that it should only be **communicated with using HTTPS**, instead of using HTTP.

The HSTS feature lets a web site inform a browser that it should never load the site using HTTP, and browser should automatically convert all attempts to access the site using HTTP to HTTPS requests instead. Supported by all "modern" browsers. Ignored when site accessed using HTTP.

Enabling this feature for your site is as simple as **returning the Strict-Transport-Security HTTP header** when your site is accessed over HTTPS:

Strict-Transport-Security: max-age=expireTime [; includeSubdomains]

expireTime

The time, in seconds, that the browser should remember that this site is only to be accessed using HTTPS.

includeSubdomains (Optional)

If this optional parameter is specified, this rule applies to all of the site's subdomains as well.

See:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security>

http://news.cnet.com/8301-1009_3-57524915-83/web-security-protocol-hsts-wins-proposed-standard-status/

Cross-origin resource sharing (CORS)

Cross-Origin Resource Sharing (CORS) allows many resources (e.g., fonts, JavaScript, etc.) on a web page to be requested across domains.

With CORS, **AJAX calls can use XMLHttpRequest across domains**. Such “cross-domain” requests would otherwise be forbidden by web browsers.

The CORS standard adds new HTTP headers. To initiate a CORS request, a browser sends the request with an “Origin” HTTP header. Suppose a page from <http://www.social-network.com> attempts to access user data from online-personal-calendar.com. If the browser supports CORS, this header is sent:

Origin: http://www.social-network.com

If the server at online-personal-calendar.com allows the request, it sends an Access-Control-Allow-Origin (ACAO) header in the response. The value of the header indicates what origin sites are allowed (*** = all sites**). For example:

Access-Control-Allow-Origin: http://www.social-network.com

If the server does not allow the CORS request, the browser will deliver an error instead of the online-personal-calendar.com response.

Firefox 3.5+, Safari 4+, Chrome3+, IE 10+, Opera 12+ support CORS. See:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS

http://enable-cors.org/server_apache.html

Networking Overview

- IP Addressing
- Routing
- DNS
- Ports
- NAT
- Subnets
- DHCP

IP Addresses

- An IP address is a unique address that is required of all computers that communicate on a network
 - IPv4 was standardized in 1983
 - IPv6 was standardized as a draft in 1998 and became an official IETF standard in 2017
- IPv4 addresses (created in 1980) consist of 32 bits separated as 4 numbers of 8 bits each
 - `128.125.253.146`
- IPv6 (IETF Draft Standard in 1998) addresses consist of 128 bits separated as 8 sets of 4 hexadecimal values
 - `fe29:392A:3396:829E:5591:40d3:3495:109A`
 - NOTE that if all four hexadecimal values are 0, the term will be omitted
 - `fe29:392A:3396::5591:40d3` (4th, 7th, and 8th terms are 0000)

IPv4 - IPv6 Address Space

- How many IPv4 addresses are there?
 - 32 bits = 2^{32} addresses
= $2^2 * 2^{30}$ addresses
= 4,294,967,296 addresses
- How many IPv6 addresses are there?
 - 128 bits = 2^{128} addresses
= $2^8 * 2^{120}$ addresses
= $256 * 2^{120}$ addresses
= 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses

$$2^{10} = 1024 = \sim 10^3 = \text{thousand}$$

$$2^{20} = 1,048,576 = \sim 10^6 = \text{million}$$

$$2^{30} = \sim 10^9 = \text{billion}$$

$$2^{40} = \sim 10^{12} = \text{trillion}$$

$$2^{50} = \sim 10^{15} = \text{quadrillion}$$

$$2^{60} = \sim 10^{18} = \text{quintillion}$$

$$2^{70} = \sim 10^{21} = \text{sextillion}$$

$$2^{80} = \sim 10^{24} = \text{septillion}$$

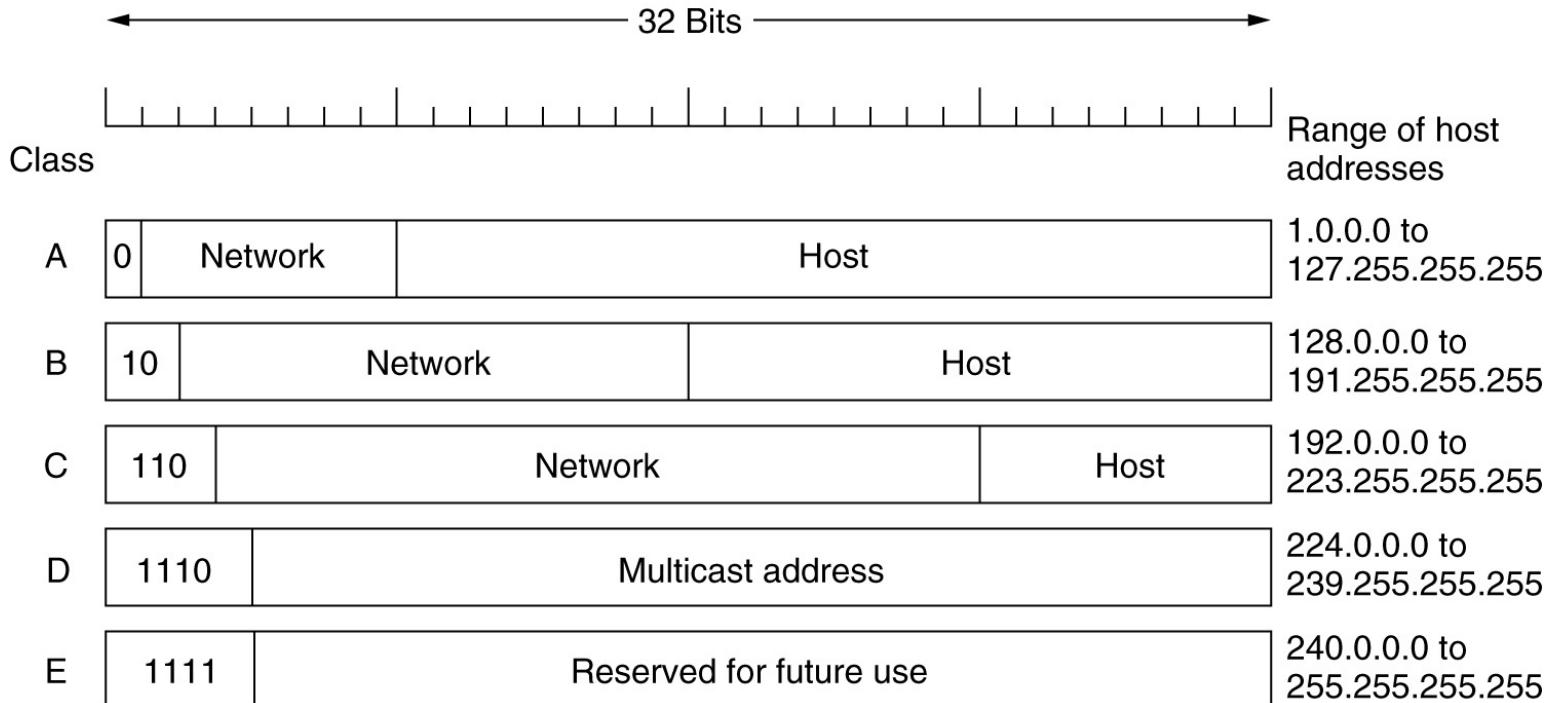
$$2^{90} = \sim 10^{27} = \text{octillion}$$

$$2^{100} = \sim 10^{30} = \text{nonillion}$$

$$2^{110} = \sim 10^{33} = \text{decillion}$$

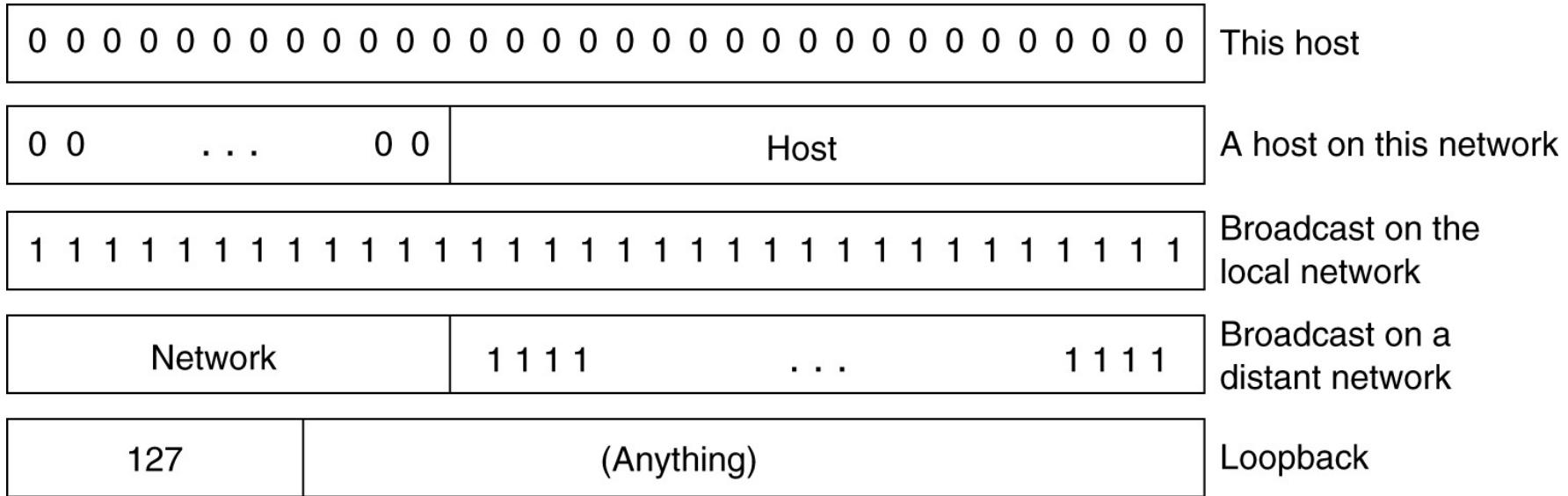
$$2^{120} = \sim 10^{36} = \text{undecillion}$$

IPv4 Classes



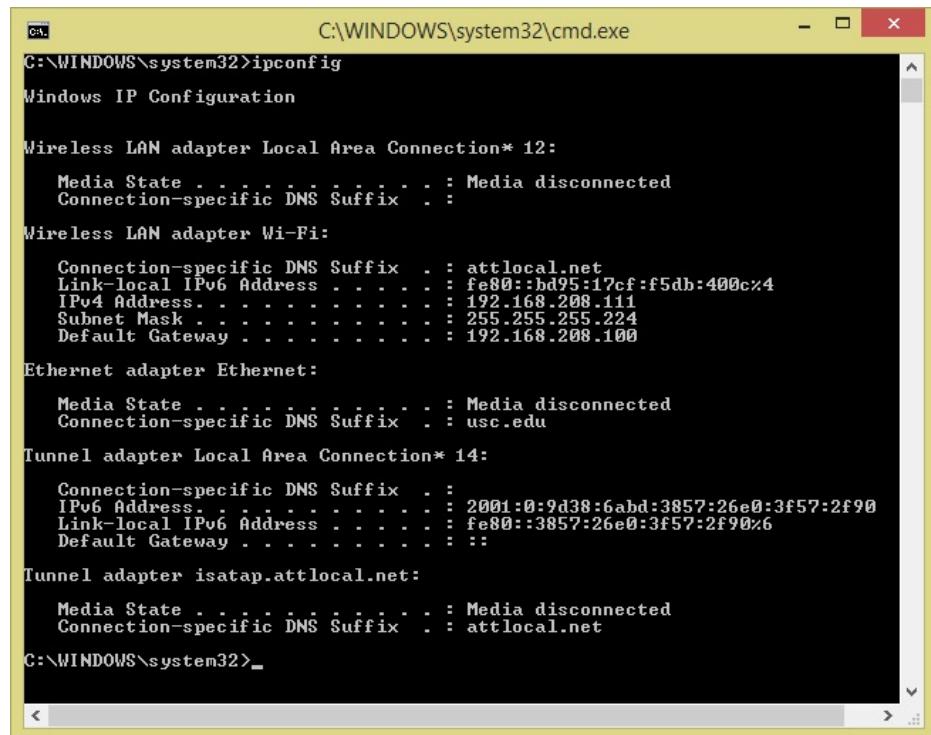
- Network numbers are managed by ICANN
- The Internet Assigned Numbers Authority (IANA) was founded by USC/ISI in 1976 by Jon Postel and transferred to ICANN in 1998
 - ICANN was founded primarily to take over control of IANA
- USC has Class B range

IPv4 Special Addresses



Finding Your IP Address

- Open a command line and type
ipconfig in Windows
ifconfig in macOS or Linux



A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The window displays the output of the "ipconfig" command. The output shows network configuration for several adapters, including "Wireless LAN adapter Local Area Connection* 12", "Wireless LAN adapter Wi-Fi", "Ethernet adapter Ethernet", and "Tunnel adapter Local Area Connection* 14". For each adapter, it lists connection-specific DNS suffixes, IPv4 and IPv6 addresses, subnet masks, and default gateways. Media state is consistently shown as "Media disconnected".

```
C:\WINDOWS\system32>ipconfig

Windows IP Configuration

Wireless LAN adapter Local Area Connection* 12:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . : attlocal.net

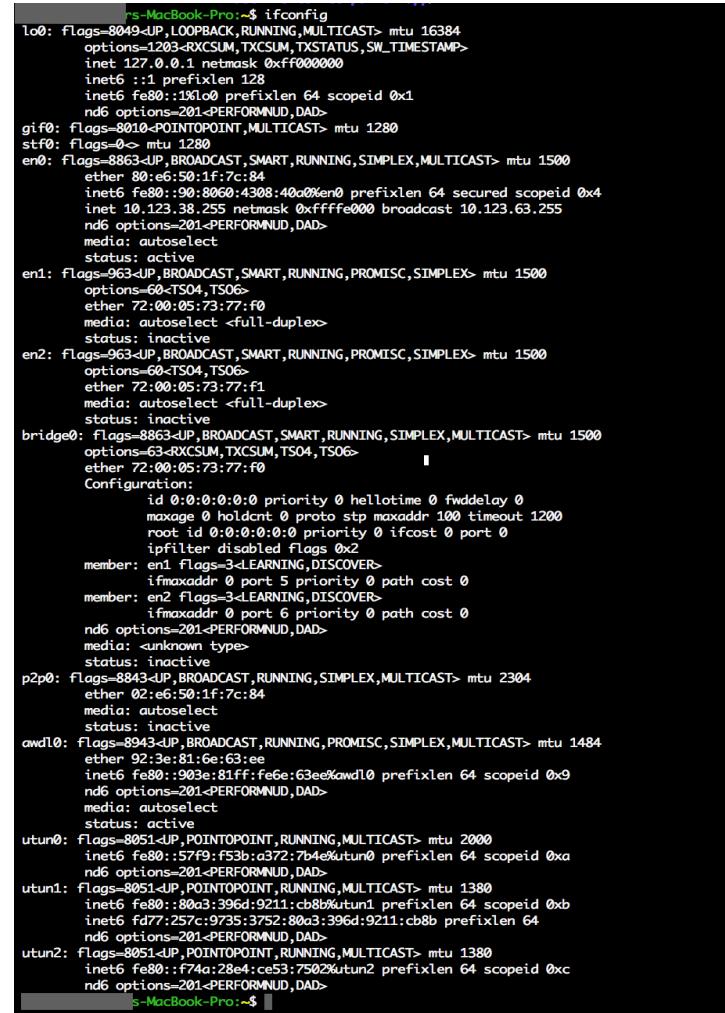
Wireless LAN adapter Wi-Fi:
  Connection-specific DNS Suffix . . . . . : attlocal.net
  Link-local IPv6 Address . . . . . : fe80::bd95:17cf:f5db:400c%4
  IPv4 Address . . . . . : 192.168.208.111
  Subnet Mask . . . . . : 255.255.255.224
  Default Gateway . . . . . : 192.168.208.100

Ethernet adapter Ethernet:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . : usc.edu

Tunnel adapter Local Area Connection* 14:
  Connection-specific DNS Suffix . . . . . : 2001:0:9d38:6abd:3857:26e0:3f57:2f90
  Link-local IPv6 Address . . . . . : fe80::3857:26e0:3f57:2f90%6
  Default Gateway . . . . . : ::

Tunnel adapter isatap.attlocal.net:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . . . . . : attlocal.net

C:\WINDOWS\system32>
```



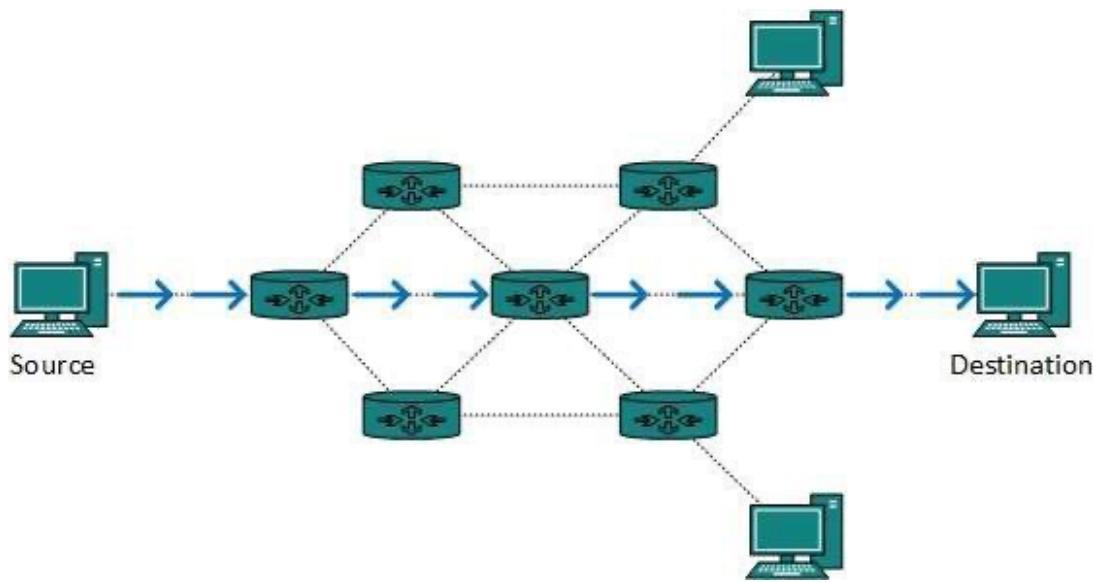
A screenshot of a macOS Terminal window titled "rs-MacBook-Pro:~\$". The window displays the output of the "ifconfig" command. The output lists various network interfaces: "lo0", "inet6 fe80::1%lo0", "gif0", "stf0", "en0", "en1", "en2", "bridge0", "p2p0", "awdl0", "utun0", "utun1", and "utun2". For each interface, it provides flags, MTU, options, and detailed information about their configuration, including ether MAC address, broadcast address, netmask, and scopeid.

```
rs-MacBook-Pro:~$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
  options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
  inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
      inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
  gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
  stf0: flags=0<inet mtu 1280
  en0: flags=8833<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:65:50:1f:7c:84
    inet6 fe80::90:8060:4308:40a0%en0 prefixlen 64 secured scopeid 0x4
    inet 10.123.38.255 netmask 0xffffffff broadcast 10.123.38.255
      nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
  en1: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
    options=60<TS04,TS06>
    ether 72:00:05:73:77:f0
      media: autoselect <full-duplex>
      status: inactive
  en2: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
    options=60<TS04,TS06>
    ether 72:00:05:73:77:f1
      media: autoselect <full-duplex>
      status: inactive
  bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=63<TXCSUM,TXSUM,TS04,TS06>
    ether 72:00:05:73:77:f0
      Configuration:
        id 0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
        maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
        root id 0:0:0:0:0 priority 0 ifcost 0 port 0
        ipfilter disabled Flags 0x2
      member: en1 flags=3<LEARNING,DISCOVER>
        ifmaxaddr 0 port 5 priority 0 path cost 0
      member: en2 flags=3<LEARNING,DISCOVER>
        ifmaxaddr 0 port 6 priority 0 path cost 0
      nd6 options=201<PERFORMNUD,DAD>
        media: <unknown type>
        status: inactive
  p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
    ether 02:e6:50:1f:7c:84
      media: autoselect
      status: inactive
  awdl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1484
    ether 92:3e:81:6e:63:ee
    inet6 fe80::903e:81ff:fe6e:63e%awdl0 prefixlen 64 scopeid 0x9
      nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
  utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
    inet6 fe80::57f9:f5b3:372:7b4%utun0 prefixlen 64 scopeid 0xa
      nd6 options=201<PERFORMNUD,DAD>
  utun1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
    inet6 fe80::80a3:396d:9211:cbb8%utun1 prefixlen 64 scopeid 0xb
      inet6 fd77:257c:9735:3752:80a3:396d:9211:cb8b prefixlen 64
      nd6 options=201<PERFORMNUD,DAD>
  utun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
    inet6 fe80::f74a:28e4:ce53:7502%utun2 prefixlen 64 scopeid 0xc
      nd6 options=201<PERFORMNUD,DAD>

s-MacBook-Pro:~$
```

Routing

- Routing is the process of sending data from one computer to another
- Routers use two primary algorithms – **Distance Vector** and **Link State**
 - The specifics of those algorithms are outside the scope of this lecture

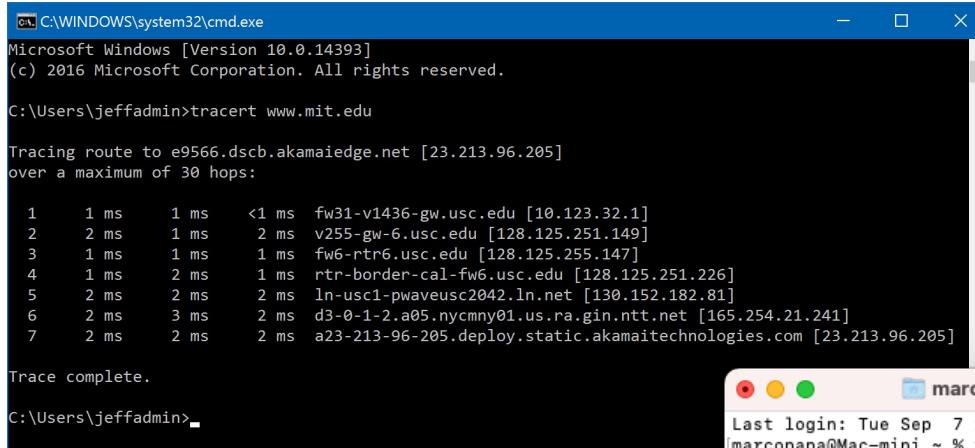


Routing



Finding a Route

- This is a very complicated process, but running **tracert** (Windows) or **traceroute** (macOS, Linux) from a command line or terminal will show you all the routers visited from your computer to a destination



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

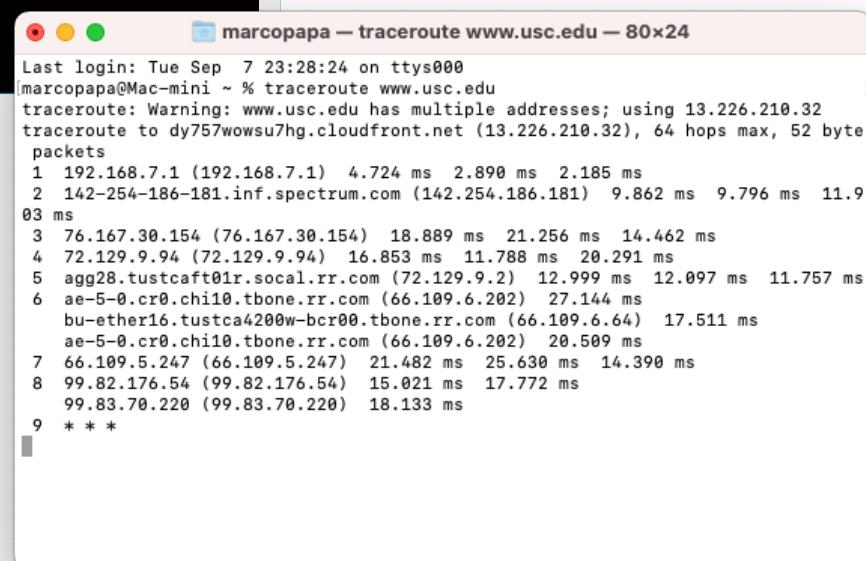
C:\Users\jeffadmin>tracert www.mit.edu

Tracing route to e9566.dscb.akamaiedge.net [23.213.96.205]
over a maximum of 30 hops:

 1  1 ms    1 ms    <1 ms  fw31-v1436-gw.usc.edu [10.123.32.1]
 2  2 ms    1 ms    2 ms  v255-gw-6.usc.edu [128.125.251.149]
 3  1 ms    1 ms    1 ms  fw6-rtr6.usc.edu [128.125.255.147]
 4  1 ms    2 ms    1 ms  rtr-border-cal-fw6.usc.edu [128.125.251.226]
 5  2 ms    2 ms    2 ms  ln-usc1-pwaveusc2042.ln.net [130.152.182.81]
 6  2 ms    3 ms    2 ms  d3-0-1-2.a05.nycmny01.us.ra.gin.ntt.net [165.254.21.241]
 7  2 ms    2 ms    2 ms  a23-213-96-205.deploy.static.akamaitechnologies.com [23.213.96.205]

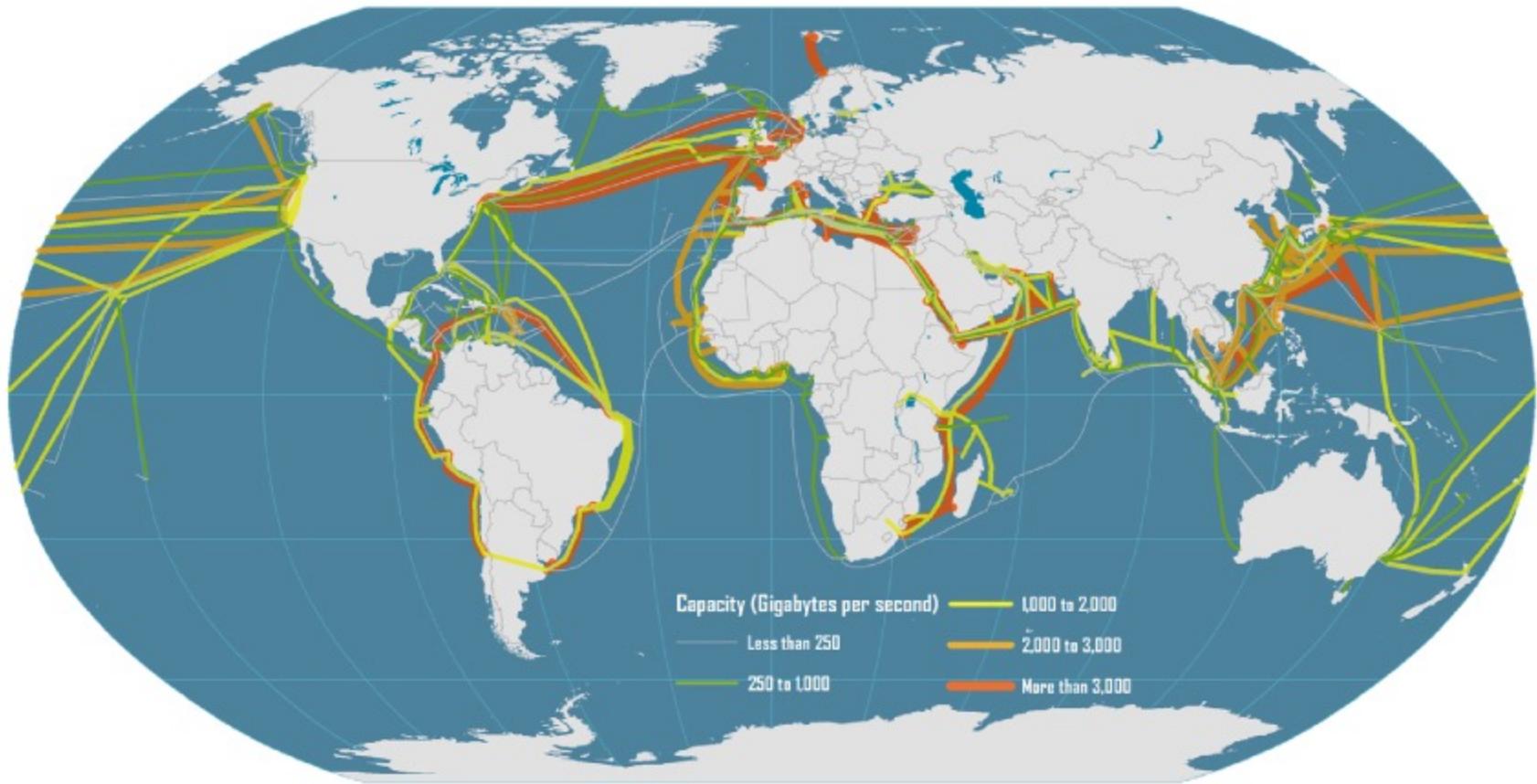
Trace complete.

C:\Users\jeffadmin>
```



```
marcopapa — traceroute www.usc.edu — 80x24
Last login: Tue Sep  7 23:28:24 on ttys000
[marcopapa@Mac-mini ~ % traceroute www.usc.edu
traceroute: Warning: www.usc.edu has multiple addresses; using 13.226.210.32
traceroute to dy757wou7hg.cloudfront.net (13.226.210.32), 64 hops max, 52 byte
packets
 1  192.168.7.1 (192.168.7.1)  4.724 ms  2.890 ms  2.185 ms
 2  142-254-186-181.inf.spectrum.com (142.254.186.181)  9.862 ms  9.796 ms  11.9
03 ms
 3  76.167.30.154 (76.167.30.154)  18.889 ms  21.256 ms  14.462 ms
 4  72.129.9.94 (72.129.9.94)  16.853 ms  11.788 ms  20.291 ms
 5  agg28.tustcaft01r.socal.rr.com (72.129.9.2)  12.999 ms  12.097 ms  11.757 ms
 6  ae-5-0.cr0.chi10.tbone.rr.com (66.109.6.202)  27.144 ms
     bu-ether16.tustca4200w-bcr00.tbone.rr.com (66.109.6.64)  17.511 ms
     ae-5-0.cr0.chi10.tbone.rr.com (66.109.6.202)  20.509 ms
 7  66.109.5.247 (66.109.5.247)  21.482 ms  25.630 ms  14.390 ms
 8  99.82.176.54 (99.82.176.54)  15.021 ms  17.772 ms
     99.83.70.220 (99.83.70.220)  18.133 ms
 9  * * *
```

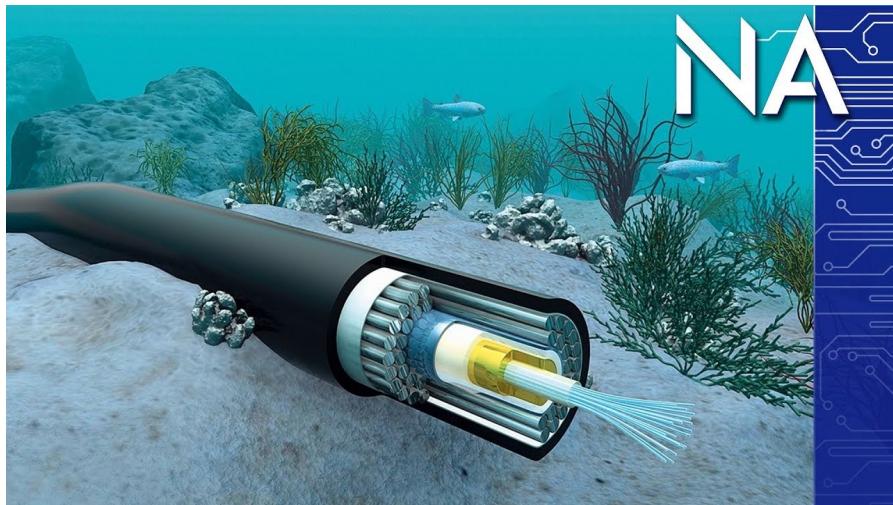
Trans-Oceanic Pipelines



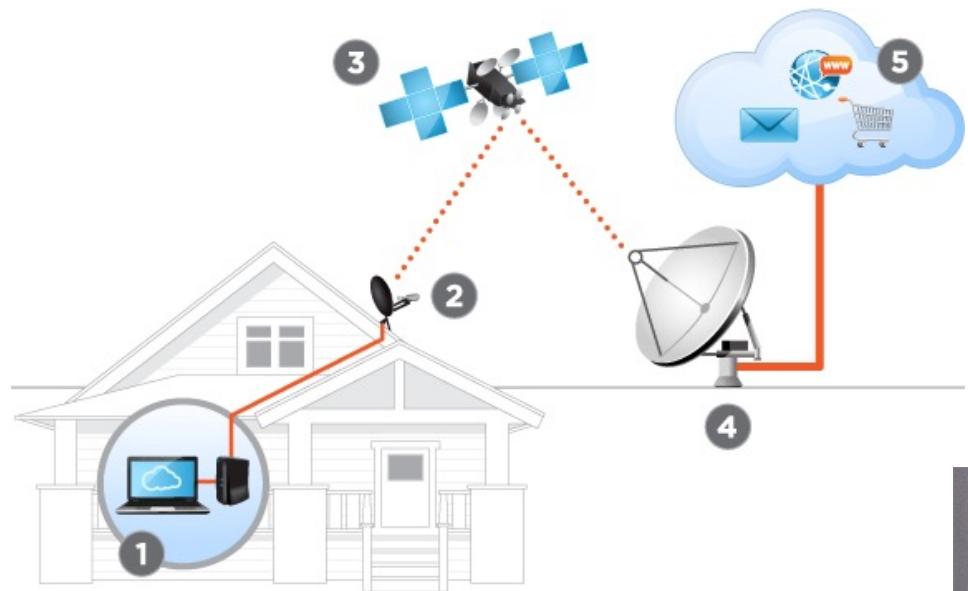
Trans-Oceanic Pipelines



Microsoft and Facebook finished MAREA, a 4000-mile-long transatlantic communication cable, transmitting 160Tb/sec, connecting Virginia to Spain around 1700 feet below the surface, in February 2018



Satellite Communication



SpaceX Starlink

STARLINK DEPLOYMENT STATUS

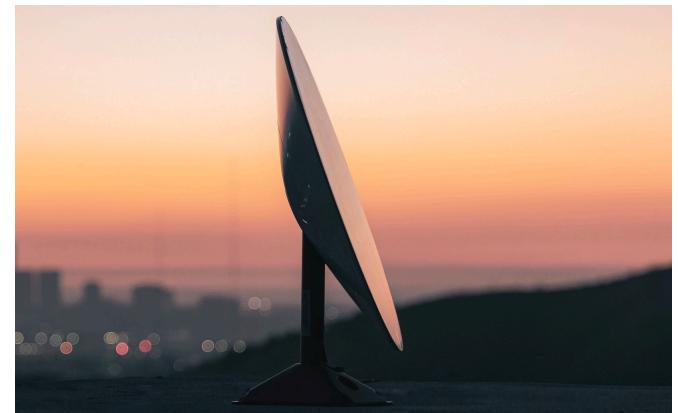
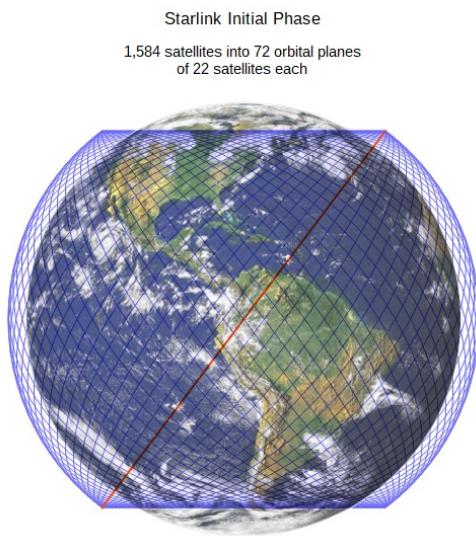
~1800 satellites launched

Starlink is rapidly rolling out internationally

- Currently serving ~140,000 users in 20+ countries (production slowed by ship shortage)
- Over 750,000 orders/deposits globally

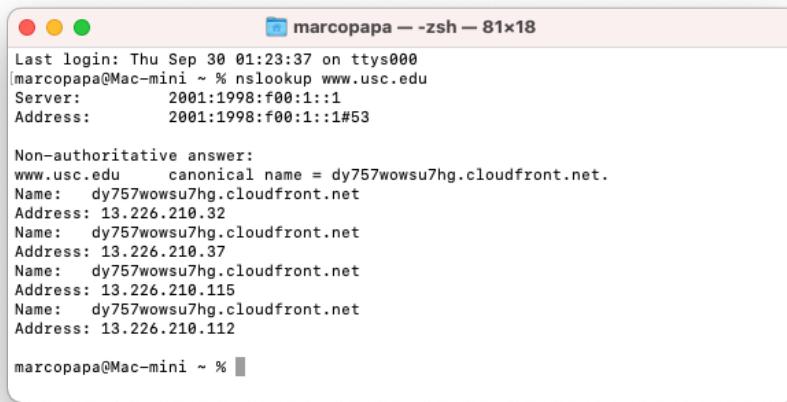


Satellite Internet Constellation
Low Earth Orbit (LEO)
1,892 satellites (as of 12/2/2021)
145,000 internet users as of 1/2022 across 25 countries
\$499 user terminal
\$99 / month
20ms latency
50-150 Mbps
<https://www.starlink.com/>



Domain Name System (DNS)

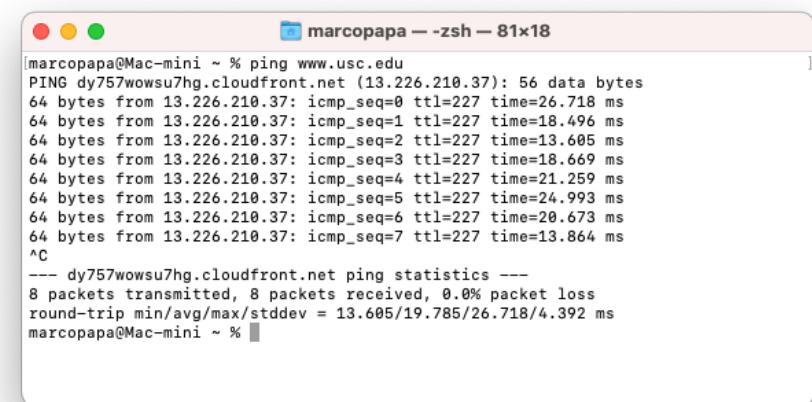
- Some servers can be identified by a hostname and domain name
 - An example would be `www.usc.edu`
 - `www` is the hostname (or an alias for a hostname)
 - `usc.edu` is the domain name
- To find the IP address of a hostname/domain name combination from a command line, run `ping` or `nslookup` from a command line or terminal



```
marcopapa@Mac-mini ~ % nslookup www.usc.edu
Server:  2001:1998:f00:1::1
Address: 2001:1998:f00:1::#53

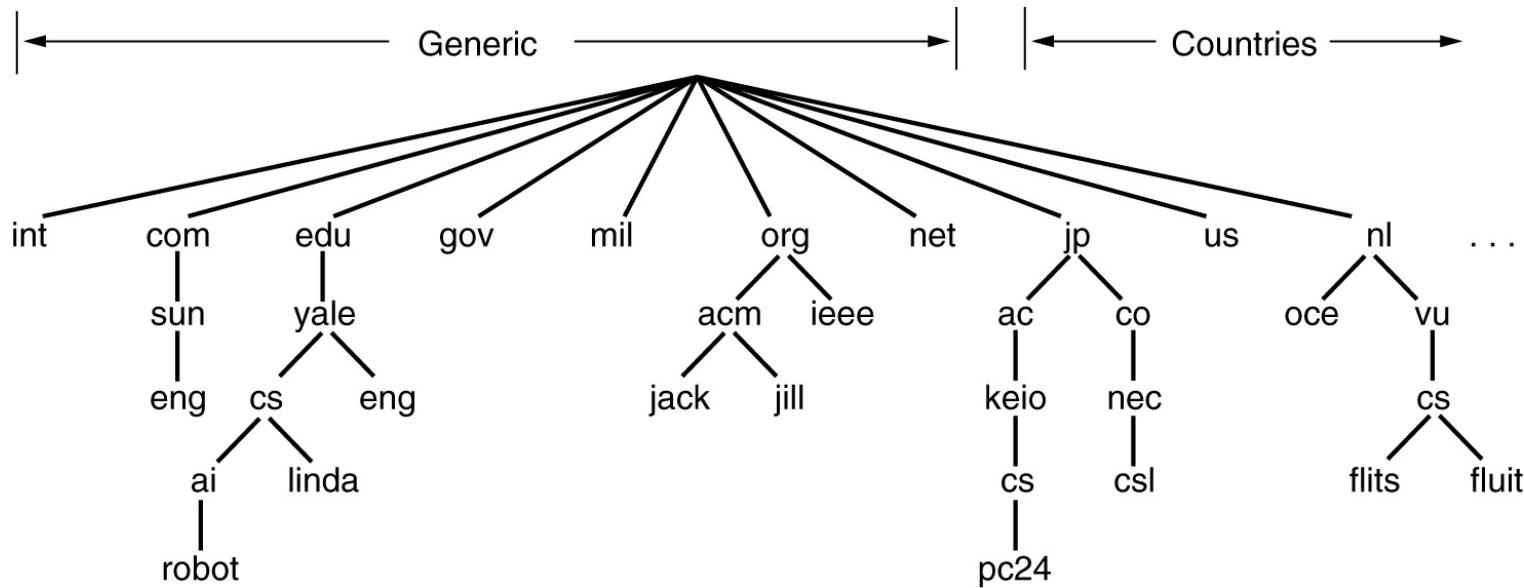
Non-authoritative answer:
www.usc.edu canonical name = dy757wowsu7hg.cloudfront.net.
Name:  dy757wowsu7hg.cloudfront.net
Address: 13.226.210.32
Name:  dy757wowsu7hg.cloudfront.net
Address: 13.226.210.37
Name:  dy757wowsu7hg.cloudfront.net
Address: 13.226.210.115
Name:  dy757wowsu7hg.cloudfront.net
Address: 13.226.210.112

marcopapa@Mac-mini ~ %
```



```
marcopapa@Mac-mini ~ % ping www.usc.edu
PING dy757wowsu7hg.cloudfront.net (13.226.210.37): 56 data bytes
64 bytes from 13.226.210.37: icmp_seq=0 ttl=227 time=26.718 ms
64 bytes from 13.226.210.37: icmp_seq=1 ttl=227 time=18.496 ms
64 bytes from 13.226.210.37: icmp_seq=2 ttl=227 time=13.605 ms
64 bytes from 13.226.210.37: icmp_seq=3 ttl=227 time=18.669 ms
64 bytes from 13.226.210.37: icmp_seq=4 ttl=227 time=21.259 ms
64 bytes from 13.226.210.37: icmp_seq=5 ttl=227 time=24.993 ms
64 bytes from 13.226.210.37: icmp_seq=6 ttl=227 time=20.673 ms
64 bytes from 13.226.210.37: icmp_seq=7 ttl=227 time=13.864 ms
^C
--- dy757wowsu7hg.cloudfront.net ping statistics ---
8 packets transmitted, 8 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 13.605/19.785/26.718/4.392 ms
marcopapa@Mac-mini ~ %
```

DNS Namespace



- The hostname/domain name combination will be mapped to an IP address through DNS servers
 - DNS is a hierarchical domain-based naming scheme implemented through a distributed database system for implementing
 - **DNS was conceived at USC's ISI by Paul Mockapetris in 1983 and managed there until 1998**
 - DNS is now managed by ICANN

Actual DNS Records

Screenshot of the GoDaddy DNS Manager - Zone File Editor interface for the domain SIGMACODING.COM.

The interface shows the following sections:

- Header:** Domain Details, DNS Manager - Zone File E..., GoDaddy.com, LLC (US), https://dns.godaddy.com/ZoneFile.aspx?zone=SIGMACODING.COM&zoneType=0&refer=dcc&prog_id, Search, Welcome: jeffream, Log Out, My Account, GoDaddy.com.
- Main Navigation:** Home, Templates, Reports, Admin, Help.
- Sub-navigation:** Dashboard, Zone File Editor, Advanced Settings.
- Advertisement:** 25% off your order of \$75 or more*, Save BIG on websites, domain names and more. SHOP NOW!
- Domain Information:** sigmacoding.com (Last saved 6/29/2014 5:55:36 PM MST) with a **Saved** button and 107 records remaining. Upgrade now.
- Record Types:**
 - A (Host):** Host @ points to 104.49.170.234 with TTL 1 Hour.
 - CNAME (Alias):** Host autodiscover points to autodiscover.outlook.com with TTL 1 Hour; Host ftp points to @ with TTL 1 Hour; Host www points to @ with TTL 1 Hour.
 - MX (Mail Exchanger):** Priority 0 Host @ points to sigmacoding-com.mail.eo.outlook.com with TTL 1 Hour.
- Filter:** Select: All, None, With Records. Options include:
 - A Record (Host) 1
 - CNAME (Alias) 3
 - MX (Mail Exchanger) 1
 - TXT (Text) 1
 - SRV (Service) 0
 - AAAA (IPv6 Host) 0
 - NS (Nameserver) 2
- Answer Center:** Links to:
 - Why should I upgrade to Premium DNS?
 - Upgrading to Premium DNS
 - What are zone files and zone records?
 - Checking Your Domain Name's Zone File Records

Copyright © 1999-2015 GoDaddy.com, LLC. All rights reserved.

Ports

- Since more than one networked program can run on a computer at the same time, we need a way to uniquely identify them
 - Ports allow us to do just that
- A client application will specify the port on the server with which to communicate, and that uniquely identifies the server application
- A port is in the range from 0 to 65535 (16 bits)
 - **0-1023** (inclusively) are **reserved** for well-known applications, so root or administrator access is required to run a program on a port in that range
 - **1024-49151** (inclusively) are registered ports and can be used by any application
 - **49152-65535** are dynamic or private ports and are typically used by the operating system when an application needs to pass an application off to a non-registered port

Well Known Ports

Port	Application
20	FTP data
21	FTP control
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
143	IMAP
443	HTTPS

- There are many others, but these are some of the more popular ones

Public and Private IP addresses

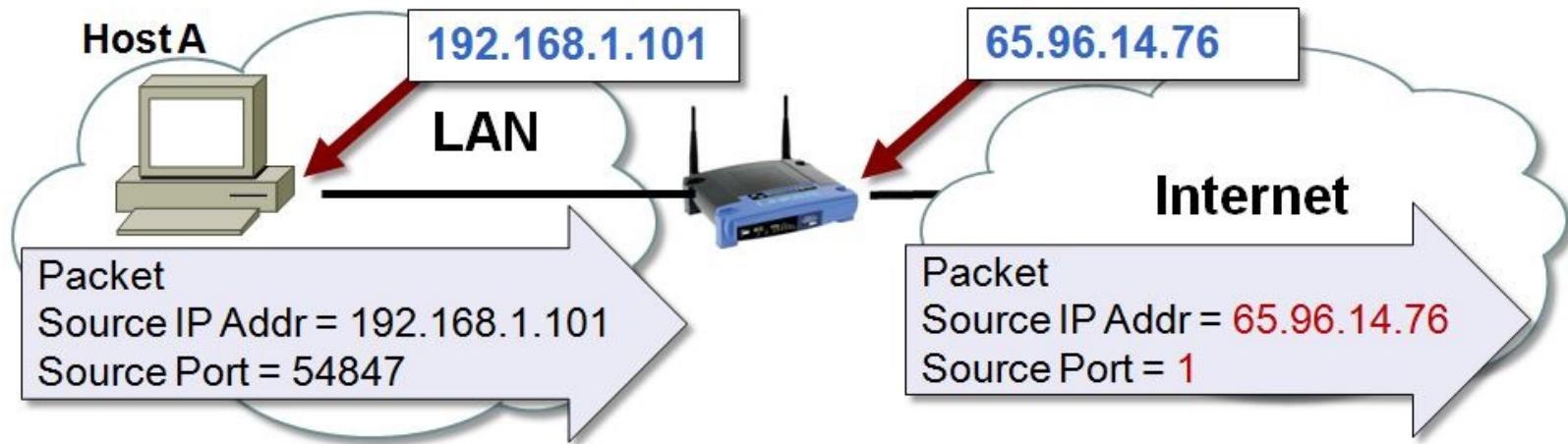
- **Public IP addresses** can be seen by any computer in the world and are required for communicating on the Internet
- Private IP addresses are typically secured behind a firewall
- Private IPv4 addresses are in the following ranges
 - **Class A** Private: **10.0.0.0 – 10.255.255.255**
 - $2^0=1$ network with 2^{24} addresses
 - **Class B** Private: **172.16.0.0 – 172.31.255.255**
 - $2^4=16$ networks with 2^{16} addresses
 - **Class C** Private: **192.168.0.0 – 192.168.255.255**
 - $2^8=256$ networks with 2^8 hosts on each network
- Private IP addresses must use NAT (see next slide) if accessing the Internet because **private IP addresses cannot communicate outside local networks**
- IPv6 doesn't have a need for private IP addresses (why?), but it still could use them if desired
- What does USC use for USC Guest Wireless and USC Secure Wireless?



NAT

- **Network Address Translation** (NAT) allows a computer to have a private IP address (IP Masquerading)
 - Private IP addresses are not able to be accessed by hosts outside the local network
 - A NAT server (usually implemented in a router) substitutes its own public IP address in place of the computer's private IP address
 - The NAT server must maintain a NAT table that links the private IP address and TCP/UDP source port combination to the destination IP address
 - Since the NAT server's public IP address has been substituted for the computer's private IP address in the packet, the response from the destination computer will come back to the NAT server
 - The port will let the NAT server know to which computer to forward the response

NAT Example



NAT Translation Table				
	Local IP Address	Source Port #	Internet IP Address	Source Port #
process X, Host A →	192.168.1.101	54,847	= 65.96.14.76	1
Host B →	192.168.1.103	24,123	= 65.96.14.76	2
process Y, Host A →	192.168.1.101	42,156	= 65.96.14.76	3
Host C →	192.168.1.102	33,543	= 65.96.14.76	4

NAT Example (cont'd)

- Host A = MacBook, 192.168.1.101
- Process X = Chrome, port 54847
- Connecting to www.usc.edu home page: 13:35.90.69:80
- NAT maps 192.168.1.101:54847 to 65.96.14.76:1
- Browser requests page from www.usc.edu:
- **Source: 65.96.14.76:1, Destination: 13:35.90.69:80**
- USC web server responds with a page:
- **Source: 13:35.90.69:80, Destination: 65.96.14.76:1**
- Router uses NAT table to map 65.96.14.76:1 to 192.168.1.101:54847
- Chrome on the MacBook receives the page

Subnets

- Since network addresses are so scarce in IPv4, we can take a few bits away from the host address to make a **subnet** within a network
- To implement subnetting, the router needs a subnet mask that indicates the split between the network/subnet combination and the host
- The **subnet mask** will consist of all 1's followed by all 0's
 - $255.255.255.0 = 11111111 \ 11111111 \ 11111111 \ 00000000$ **VALID**
 - $255.255.255.2 = 11111111 \ 11111111 \ 11111111 \ 00000010$ **NOT VALID**
- The subnet mask can also be written using slash notation
 - The number after the slash will represent the number of bits to be used in the subnet address
 - This would correspond to the number of 1's in the subnet mask
 - $255.255.255.0$ could be written as $/24$, a class C network

Subnets in IPv6

- Subnetting exists in IPv6, but because there are so many bits in the IP address, we don't need to take bits from the host
 - Bits 1-48 - Network address
 - Bits 49-64 - Subnet address
 - Bits 65-128 - Host address
- Slash notation is used to identify the number of bits used in the network and subnet, but regardless, there are always 2^{64} hosts on every subnet
 - A /64 means there is no subnetting
 - A /48 means there are 2^{16} subnets available on that network
 - This is the maximum number of subnets available on an IPv6 network

Subnet Example #1

- Assume a host has an IP address of 74.125.127.104 with a subnet mask of 255.255.255.192 (also written as 74.125.127.104/26)
 - What is the network address? (i.e., what class IP address is this?)
 - What is the subnet mask in binary?
 - How many hosts can be in the subnet?
 - 74.125.127.104 = 01001010 01111101 01111111 01101000
 - 255.255.255.192 = 11111111 11111111 11111111 11000000
- Answers:
 - **Network Address** - this is a Class A address, so the first 8 bits are allocated for the network: **74.0.0.0**
 - **Subnet Mask** in binary will have the first 26 bits as 1's with the last 6 bits as 0's
 - **# Hosts** - the last 6 bits are reserved for hosts, giving $2^6 - 2 = 62$ host addresses for this subnet
 - Remember that the address with all 0s and all 1s are reserved

Subnet Example #2

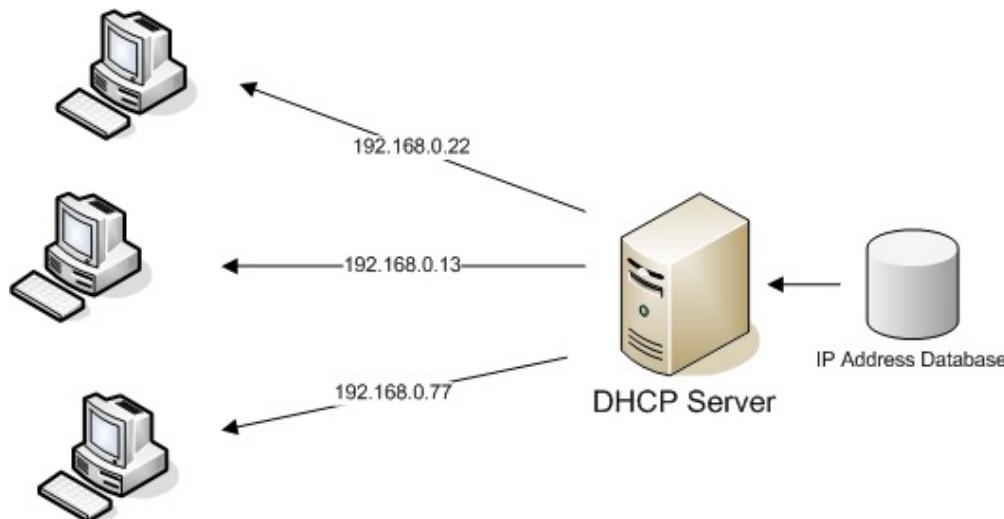
- To get the network/subnet address from an IP address and subnet mask, perform a **logical AND** operation between them

137.229.154.221	=	1000 1001 1110 0101 1001 1010 1101 1101
255.255.224.0	=	1111 1111 1111 1111 1110 0000 0000 0000
137.229.128.0	=	1000 1001 1110 0101 1000 0000 0000 0000

- The **network/subnet address** is 137.229.128.0
- This **subnet mask** provides **19** bits for the network address and 13 bits for the host address
- Another way we could have written the IP address with the subnet mask is 137.229.154.221/**19**
- Note that the subnet mask can never contain fewer 1 bits than the number of bits in the network address
 - Why does that make sense?

DHCP

- **Dynamic Host Control Protocol** (DHCP) is used for a computer or router to automatically assign IP addresses and other network configuration (such as the gateway and subnet mask) to computers on the network
 - These addresses can be private or public IP addresses
 - Most routers assign private IP addresses, such as 192.168.1.101
 - Routers often have DHCP servers built into them

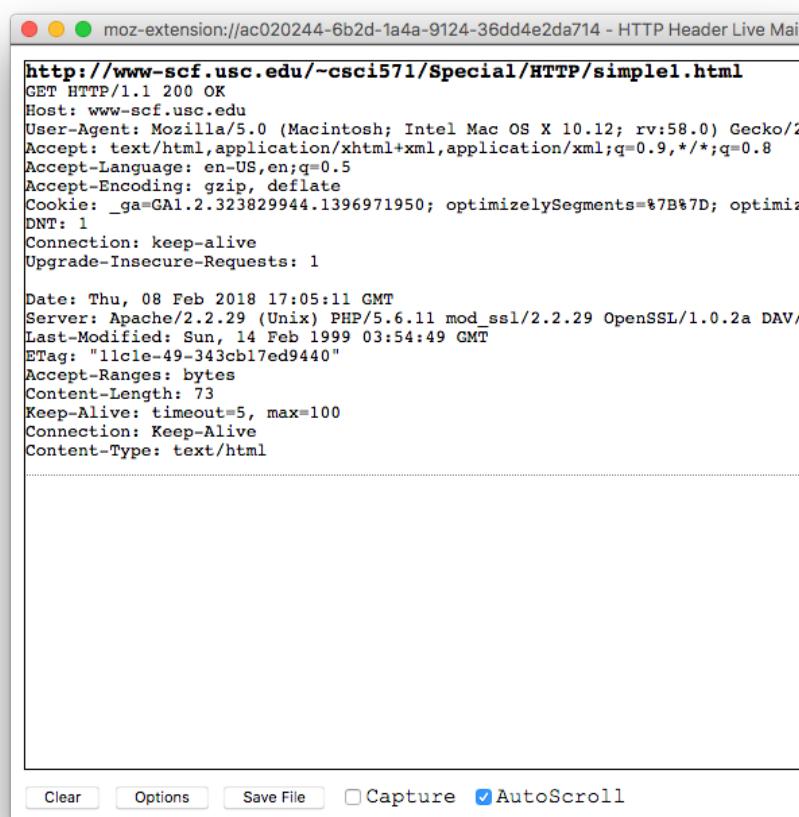


Exercises to Examine the HTTP Protocol

1. HTTP Header Live
2. Wireshark
3. Postman

Firefox “HTTP Header Live” add-on

- Download at
 - <https://addons.mozilla.org/en-US/firefox/addon/http-header-live/>
 - Compatible with Firefox Quantum 57+



The screenshot shows the 'HTTP Header Live Main' window. At the top, it displays the URL <http://www-scf.usc.edu/~csci571/Special/HTTP/simple1.html>. Below the URL, the page content is shown as a blank white area. Above the content, the full HTTP request and response headers are listed. The request headers include:

```
GET HTTP/1.1 200 OK
Host: www-scf.usc.edu
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:58.0) Gecko/2
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: _ga=GAI.2.323829944.1396971950; optimizelySegments=%7B%7D; optimiz
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

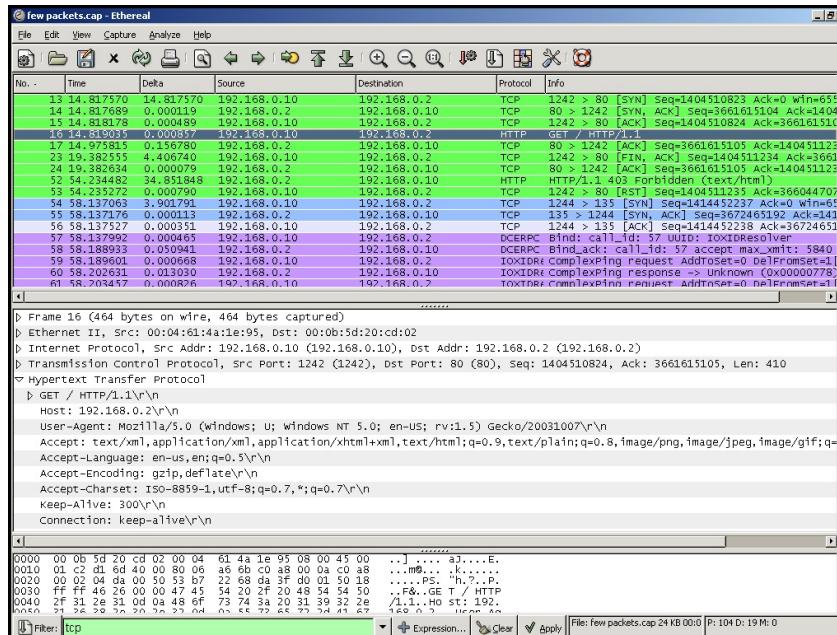
The response headers include:

```
Date: Thu, 08 Feb 2018 17:05:11 GMT
Server: Apache/2.2.29 (Unix) PHP/5.6.11 mod_ssl/2.2.29 OpenSSL/1.0.2a DAV/
Last-Modified: Sun, 14 Feb 1999 03:54:49 GMT
ETag: "11c1e-49-343cb17ed9440"
Accept-Ranges: bytes
Content-Length: 73
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

At the bottom of the window, there are several buttons: 'Clear', 'Options', 'Save File', 'Capture' (unchecked), and 'AutoScroll' (checked).

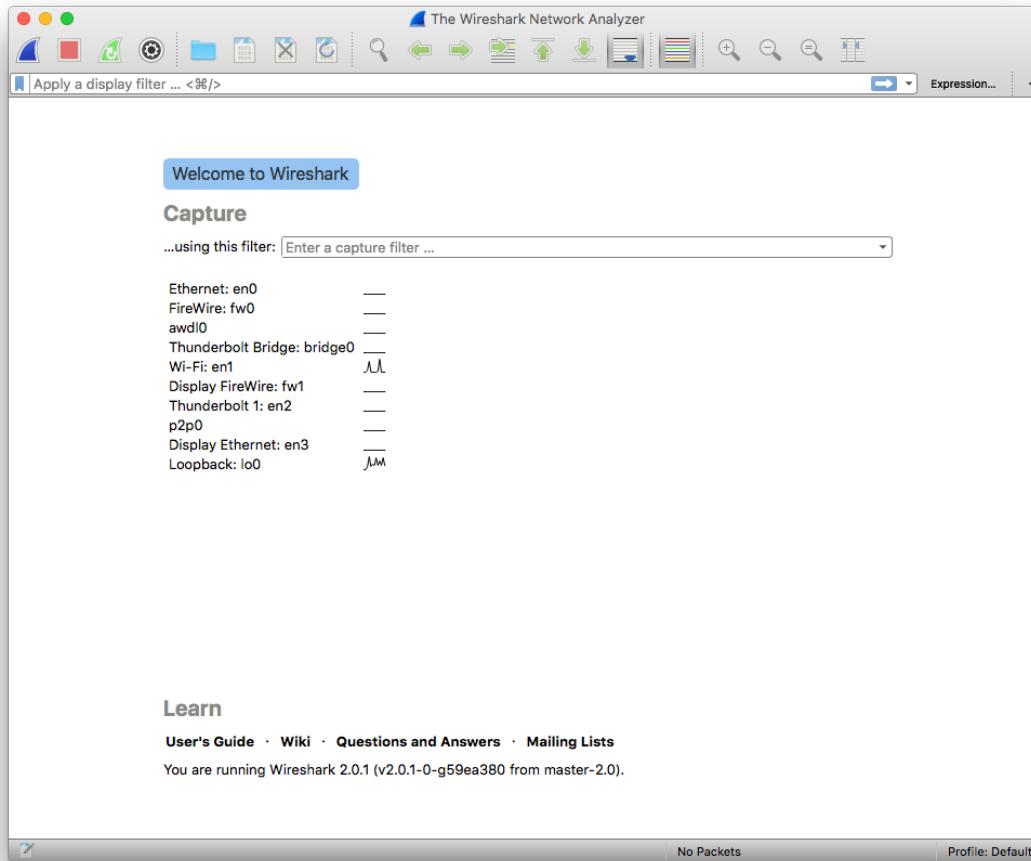
Wireshark Protocol Analyzer

- Install a protocol analyzer (a.k.a. a “sniffer”), such as **WireShark** (formerly Ethereal), available at
 - <http://www.wireshark.org/>
- Captured network data can be browsed via a GUI
- Supports 1300+ protocols
- Multiplatform: Windows, macOS, Linux, and many more



www.wireshark.org

- Download the latest “stable” release of Wireshark (currently 3.4.8)



Initial Screen (new interface)

Filter HTTP requests

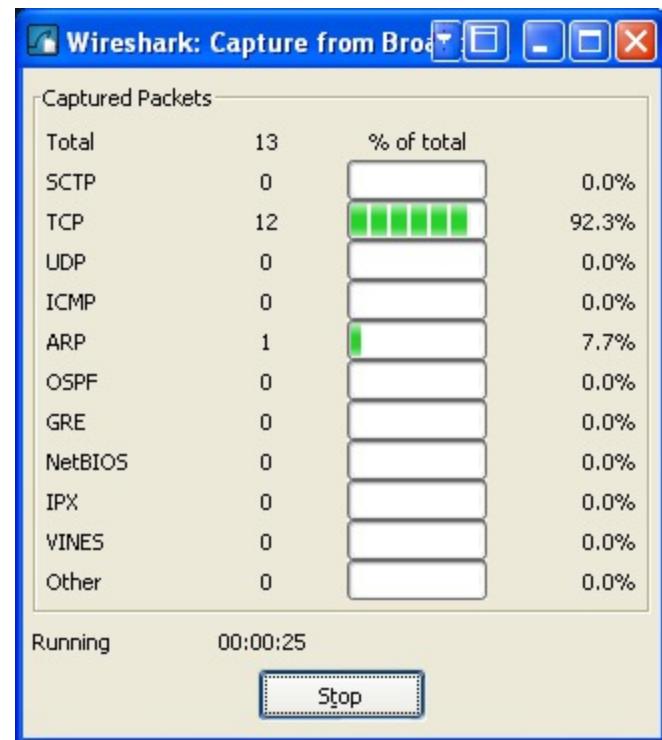
- In your browser open up the page
<http://csci571.com/Special/HTTP/proxy.html>
- In Wireshark
 - Click on Expression on the right-hand side of the third row
 - Select HTTP and click on OK
 - Click on Apply on the third row at the right-hand end
- In Wireshark
 - Click on Capture | Interfaces and then click the Start button that is on the line related to your wired/wireless connection to the Internet
 - A window appears as shown on the next slide

Start up the web Traffic

- Return to the browser and paste the first test URL into the address field

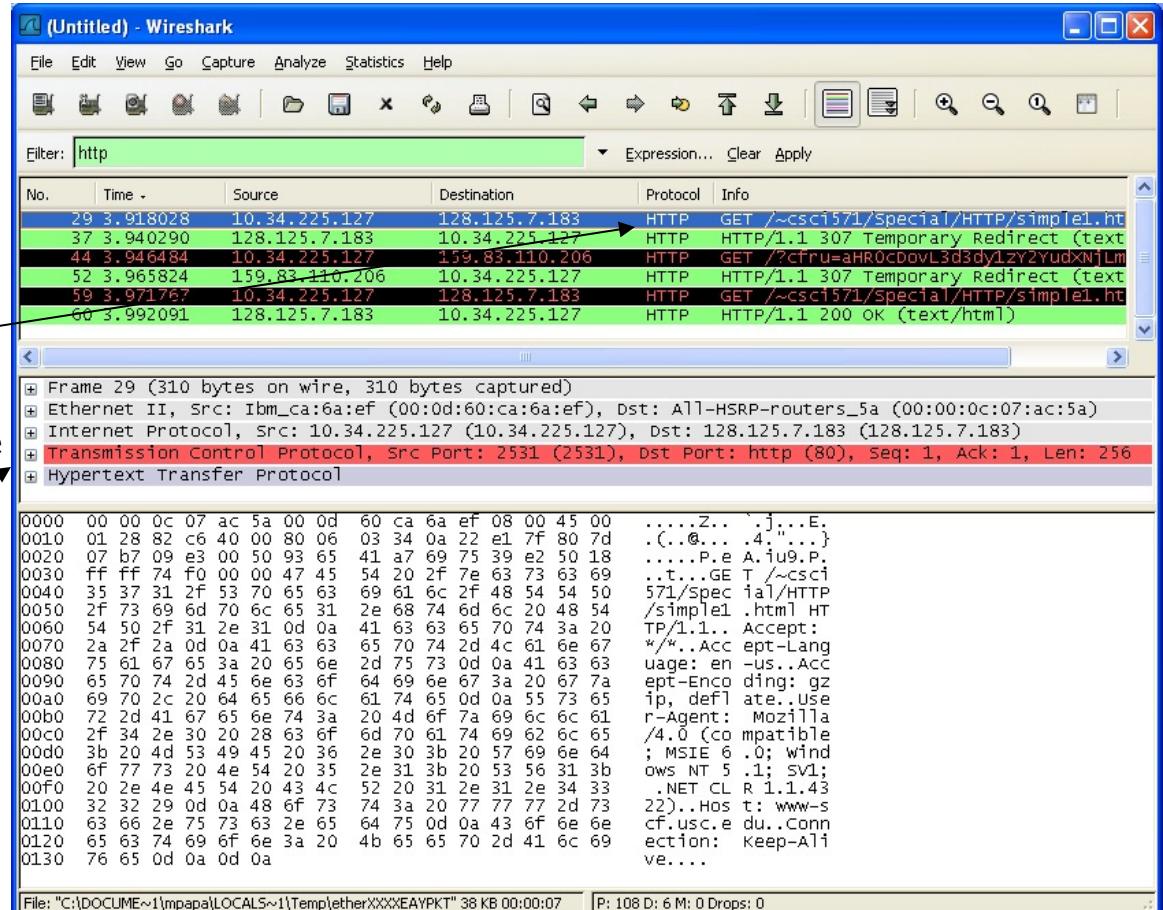
<http://csci571.com/Special/HTTP/simple1.html>

- Once the requested page appears in the browser return to the window that is shown on the right and click on the Stop button
- A new Wireshark screen appears as shown on the next slide



WireShark Output

- Scroll the green window until a line with HTTP GET appears;
- Click on the line;
- Then, in the middle window expand the line labeled Hypertext Transfer Protocol



Expansion of HTTP request

The screenshot shows the Wireshark interface with a captured session titled '(Untitled) - Wireshark'. A filter bar at the top is set to 'http'. The main pane displays three network frames. Frame 224 is selected and expanded, showing its detailed structure:

- Frame 224 (636 bytes on wire, 636 bytes captured)
- Ethernet II, Src: Intel_7a:c9:d8 (00:11:11:7a:c9:d8), Dst: LinksysG_e7:0d:e9 (00:06:25:e7:0d:e9)
- Internet Protocol, Src: 192.168.1.100 (192.168.1.100), Dst: 128.125.5.229 (128.125.5.229)
- Transmission Control Protocol, Src Port: 1782 (1782), Dst Port: http (80), Seq: 1, Ack: 1, Len: 582
- Hypertext Transfer Protocol
 - GET /~csci571/Special/HTTP/simple1.html HTTP/1.1\r\n
 - Host: www-scf.usc.edu\r\n
 - User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1) Gecko/20061010 Firefox/2.0\r\n
 - Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5,text/vnd.wap.wml;q=0.6\r\n
 - Accept-Language: en-us,en;q=0.5\r\n
 - Accept-Encoding: gzip,deflate\r\n
 - Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
 - Cookie: __utma=185748653.462061095.1156035922.1165639737.1165785580.10; MintUnique=1; __utmz=185748653.1156035922.1.1.utmccn=(direct)\r\n

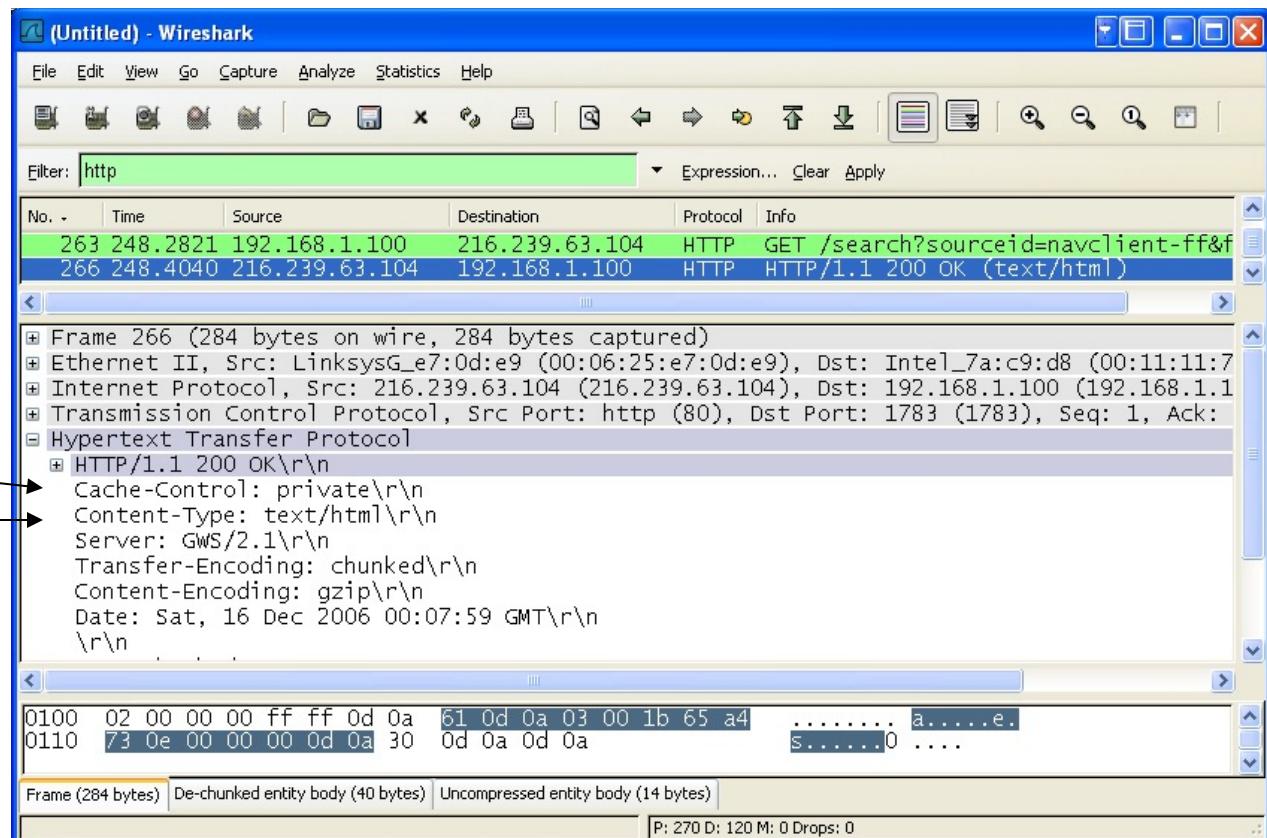
The bottom pane shows the raw hex and ASCII data of the selected frame.

Notice the:

- GET request
- Host
- User-Agent
- Accepts
- Cookie

Expansion of HTTP response

- Scroll down again in the upper window and click on the HTTP/1.1 200 OK line, as shown to the right
- Notice the additional lines that are sent including
 - Cache-Control
 - Content-Type
 - Server
 - Etc.



Repeat the Exercise

- Using the remaining links on
<http://csci571.com/Special/HTTP/proxy.html>
do the following:
 - In Wireshark, select the Capture Start menu
 - Click on the button “Continue without Saving”
 - Enter the new URL into the browser
 - Once the requested page appears click on Stop in the Wireshark Capture window
 - Examine the result in the main Wireshark window
 - Remember to note the HTTP commands, request, response and general commands

Postman

- Collaboration platform for API development
- Supports all HTTP commands. Download from:
<https://www.getpostman.com>
- Free to individuals and small teams
- Free license includes 1,000 API calls / month
- Desktop app available for MacOS, Windows and Linux

