CSC309 Programming on the Web
Summer 2015
About Me

Mashiyat Ahmed Shah

- 4th Year PhD student
- Worked in the industry for several years
  - In Start Up,
  - In mid-size company, and
  - In large consultancy firm
Teaching Assistants

Soheil Hassas Yeganeh
Email: soheil@cs.toronto.edu

Mohammad Hossein Danesh
Email: danesh@cs.toronto.edu

Sukwon Oh
Email: sukwon0709@gmail.com
Any One here Do not Like Web Programming?
What do you like about web programming?
Why I like Web

-- Massive reach! Facebook has 1.23B users (People have multiple accounts)

<table>
<thead>
<tr>
<th>Year</th>
<th>Facebook Monthly Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1m</td>
</tr>
<tr>
<td>2005</td>
<td>6m</td>
</tr>
<tr>
<td>2006</td>
<td>12m</td>
</tr>
<tr>
<td>2007</td>
<td>58m</td>
</tr>
<tr>
<td>2008</td>
<td>145m</td>
</tr>
<tr>
<td>2009</td>
<td>360m</td>
</tr>
<tr>
<td>2010</td>
<td>608m</td>
</tr>
<tr>
<td>2011</td>
<td>845m</td>
</tr>
<tr>
<td>2012</td>
<td>1,056m</td>
</tr>
<tr>
<td>2013</td>
<td>1,230m</td>
</tr>
</tbody>
</table>

SOURCE: FACEBOOK
Fast Scaling Power

1. AAPL, $479 billion
2. GOOG, $404 billion
3. MSFT, $316 billion
4. IBM, $195 billion
5. FB, $184 billion
6. ORCL, $175 billion
7. AMZN $171 billion
8. QCOM, $130 billion
9. INTC, $123 billion
10. CSCO, $116 billion

http://www.usatoday.com/ March 13, 2014
Fast Scaling Power

1. AAPL, $479 billion
2. GOOG, $404 billion
3. MSFT, $316 billion
4. IBM, $195 billion
5. FB, $184 billion
6. ORCL, $175 billion
7. AMZN $171 billion
8. QCOM, $130 billion
9. INTC, $123 billion
10. CSCO, $116 billion

→ Alibaba.com $168 Billion

http://www.usatoday.com/  March 13, 2014
What CSC309 is about?

This course provides an introduction to concepts and technologies of web development

- Static/dynamic client-side programming
- Static/dynamic server-side programming
- Web architectures
- Web development processes
- Security, Scalability, Usability on the Web
CSC309 Objectives and Outcomes

- **Enabling Knowledge**
  Ability to define and explain protocols, concepts and technologies that enable web applications

- **Critical Analysis and Problem Solving**
  Ability to analyse, design and implement web applications that accommodate specific requirements and constraints with regard to issues of usability, performance and security

- **Communication**
  Ability to work as part of a team or individually, explore relevant alternatives and make decision recommendations
How many of you have prior web programming experience?
Course Overview

- The Internet and the Web
- HTML Basics
- Cascading Style Sheets (CSS)
- Client-side Scripting (JavaScript)
- Server-Side Programming (NodeJS)
- E/R Model and Database Design
- Asynchronous Communication (Ajax) & JSON
- Web Architectures
- Web Security Issues
- Web Performance Issues
- Other Topics (Going live!, Monetization Strategies, …)
Self Study Topics

- Almost Everything! The purpose of the class is not to introduce all topics in detail

- We introduce concepts and technologies in some detail in class but is left to you to excel by
  - Reading online material
  - Practice, Practice, Practice
  - Incorporating them to your projects
Lecture Materials

Lecture Notes and material are mostly from,

- Eyal de Lara, Professor, DCS
- Manos Papagelis, Instructor of this course in some previous terms.
- Juan Gonzalez, Founder/CEO at Videogami
- And of course from Web
Course Information

- **Instructors:**
  - Ahmed Mashiyat (mashiyat@cs.toronto.edu)
  - Tutorial (BA1220): Tuesdays, 2pm--3pm
  - Lectures (BA1220): Thursdays, 2pm--4pm

- **Website:**

- **Discussion & Communication (register online):**
  Piazza: https://piazza.com/class/i9e3qpq0p144m6
Communication

- Office hours:
  - Mashiyat (BA3289): Thursdays, 12pm--2pm

- Email (I prefer communication through piazza):
  - Email must include your name
  - Subject must include “309”
  - Your Email should:
    - State your question clearly, with enough context
    - Have your sign (Name, Student number and CDF login are the most useful)
Course Textbooks


- Online Resources!
Course Prerequisites

- Make sure you have the prerequisites!
  - CSC209H1 - Software Tools and Systems Programming
  - CSC343H1 - Introduction to Databases (Automatically Waived)

*Note: Students that would like to attend CSC309 but miss one of the prerequisites would need to email the instructor asking for a course prerequisite waiver. (There is a CGPA prerequisite as well, which can only be waived by the Undergraduate Office.)*
Projects

- Assignment 1: Individual Project
  - Personal Website
  - More details to follow

- Assignment 2: Team Project (2 students)
  - Online Game development
  - Start looking for partners

- Assignment 3: Team Project (4 students)
  - Three Phases
Class Participation

You will be marked on:

- Class participation
  - It's not attendance, however highly encouraged
  - Ask interesting questions (in class or in Piazza)
  - Answer quiz that I will post in piazza
  - Most importantly help each other in finding answers to the questions posted.
  - Be a good team mate (Also counts towards group assignments)
Policies

- All deliverables will be submitted electronically using MarkUs
- Deliverables are due at 11:59 p.m. on the due date - check website for final due dates
- Late Work Policy: Accepted up to 2 days after the due date with a 10% penalty per day
- Demo day: to showcase assignment 3
Plagiarism

- “The work you submit must be your own, done without participation by others. It is an academic offense to hand in anything written by someone else without acknowledgement.”
- You are not helping your friend when you give him or her a copy of your work
- You are hurting your friend when you ask him or her to give you a copy of their work
What is Cheating?

- Cheating is
  - copying parts or all of another student’s assignment
  - including code from books, web sites, other courses without attribution
  - getting someone else to do substantial parts of your assignment
  - giving someone else your solution

- Cheating is not
  - helping to find a bug in a friend’s code (be careful)
  - helping each other understand example code
# Course Marking Scheme

<table>
<thead>
<tr>
<th>Work</th>
<th>Weight</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment 1</td>
<td>10%</td>
<td>Personal Website</td>
</tr>
<tr>
<td>Assignment 2</td>
<td>15%</td>
<td>Online Game</td>
</tr>
<tr>
<td>Assignment 3</td>
<td>30%</td>
<td>1&lt;sup&gt;st&lt;/sup&gt; Phase – 5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2&lt;sup&gt;nd&lt;/sup&gt; Phase – 5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3&lt;sup&gt;rd&lt;/sup&gt; Phase -20%</td>
</tr>
<tr>
<td>Final</td>
<td>35%</td>
<td>You must get &gt;=40% to pass the course</td>
</tr>
</tbody>
</table>
A Few Do’s and Don’ts

- Do
  - ask questions if you don’t understand something
  - work together to understand concepts/projects
  - use tutorials and office hours
  - read material online & practice

- Don’t
  - hand in other peoples’ work (it’s cheating)
  - harass others (see the University’s policies)
  - distract or disrupt the class (it’s immature)
Today’s Overview

- Course Administrivia
- Introduction
  - History of the Internet Video (~8min)
    http://www.youtube.com/watch?v=9hlQjrMHTv4
  - The Internet and the World Wide Web
- Simple Web Request
THE INTERNET AND THE WORLD WIDE WEB
What is the internet?

- A collection of computer networks that use a protocol to exchange data

- Is the World Wide Web (WWW) and the internet the same?
Brief history

- Began as a US Department of Defense network called ARPANET (1960s-70s)
- Packet switching (in the 60s)
- E-mail is born on 1971
- TCP/IP beginning on 1974 (Vinton Cerf)
- USENET (1979)
- By 1987: Internet includes nearly 30,000 hosts
Brief history (cont.)

- WWW created in 1989-91 by Tim Berners-Lee
- Popular web browsers released:
  - Netscape 1994
  - IE 1995
- Amazon.com opens in 1995
- Google January 1996
- Wikipedia launched in 2001
- MySpace opens in 2003
- Facebook February 2004
The future of the internet?
Key aspects of the internet

- Sub-networks are independent
- Computers can dynamically join and leave the network
- Built on open standards
- Lack of centralized control (mostly)
- Everyone can use it with simple, commonly available software
People and organizations

- Internet Engineering Task Force (IETF): internet protocol standards
- Internet Corporation for Assigned Names and Numbers (ICANN): decides top-level domain names
- World Wide Web Consortium (W3C): web standards
Internet Protocol (IP)

- Simple protocol for data exchange between computers
- IP Addresses:
  - 32-bit for IPv5
  - 128-bit for IPv6
Transmission Control Protocol (TCP)

- Adds multiplexing and reliable delivery on top of IP
  - Multiplexing: multiple programs using the same IP address
  - Reliability: guaranteed, ordered and error-checked delivery
- Port: a number given to each program or service
  - port 80: web browser (port 443 for secure browsing)
  - port 25: email
  - port 22: ssh
- Some programs (games, streaming media programs) use simpler UDP protocol instead of TCP
Web Browser requests and parses documents from web servers

- Mozilla Firefox
- Microsoft Internet Explorer (IE)
- Apple Safari
- Google Chrome
- ...
Domain Name Server (DNS)

- Set of servers that map (translate) written names to IP addresses
  - Example: www.cs.toronto.edu → 128.100.3.40

- Many systems maintain a local cache called a hosts file
  - Windows: C:\Windows\system32\drivers\etc\hosts
  - Mac: /private/etc/hosts
  - Linux: /etc/hosts
Uniform Resource Locator (URL)

- Identifies the path to a document on the web server

Example URL:

Upon entering this URL into the browser, it would:
- ask the DNS server for the IP address of the URL
- connect to that IP address at port 80
- request the document from the server by sending `GET mashiyat/csc309/index.htm`
- parse and display the resulting page on the screen
Hypertext Transport Protocol (HTTP)

- Set of commands understood by a web server and sent from a browser

- Some HTTP commands (your browser sends these internally):
  - GET filename: download
  - POST filename: send a web form response
  - PUT filename: upload
HTTP Error Codes

- When something goes wrong, the web server returns a special "error code" number

- Common error codes:

<table>
<thead>
<tr>
<th>Number</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>301-303</td>
<td>page has moved (permanently or temporarily)</td>
</tr>
<tr>
<td>403</td>
<td>you are forbidden to access this page</td>
</tr>
<tr>
<td>404</td>
<td>page not found</td>
</tr>
<tr>
<td>500</td>
<td>internal server error</td>
</tr>
</tbody>
</table>
## Internet Media ("MIME") types

<table>
<thead>
<tr>
<th>MIME type</th>
<th>file extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/html</td>
<td>.html</td>
</tr>
<tr>
<td>text/plain</td>
<td>.txt</td>
</tr>
<tr>
<td>image/gif</td>
<td>.gif</td>
</tr>
<tr>
<td>image/jpeg</td>
<td>.jpg</td>
</tr>
<tr>
<td>video/quicktime</td>
<td>.mov</td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>.exe</td>
</tr>
</tbody>
</table>
Enabling Web Technologies

- **Hypertext Markup Language (HTML)**
  Used for writing web pages

- **Cascading Style Sheets (CSS)**
  Stylistic info for web pages

- **JavaScript**
  Interactive web pages (on the client)

- **Server side Programming language** (Python, Node.js, etc.)
  Dynamic and data-driven pages (on the web server)
Enabling Web Technologies (cont.)

- eXtensible Markup Language (XML):
  Markup language for describing and exchanging data

- Asynchronous JavaScript and XML (Ajax)
  Creating asynchronous Web applications

- ...
SIMPLE WEB REQUEST
Simple Web Request

Thanks to Karen Reid for material in these slides
The request

- How do we tell the server what we want?
- How do we even find the server?
- How do the server and browser talk to each other?
HTTP Request

**Request**

GET / HTTP/1.1
Connection: Keep-Alive
Host: www.tkf.toronto.on.ca

**Reply**

HTTP/1.1 301 Moved Permanently
Date: Tues, 23 July 2002
Server: Apache/1.3.22(unix)
Content-Type: text/html
How do we find the server?

- Every computer on the Internet has an Internet address.
- Called an IP address (Internet Protocol)
- An IP address is 4 numbers separated by dots.

www.tkf.toronto.on.ca = 207.245.2.3
Domain Name Servers

- Browser
- Local name server
- Root name server
- "ca" name server
- "on" name server
- "toronto" name server

www.tkf.toronto.on.ca?
This is getting complicated!

Number of messages?

10-12
Actually, it’s worse than that

- The web page for www.tkf.toronto.on.ca doesn’t really live at 207.245.2.3

  request

  GET / HTTP/1.1
  Connection: Keep-Alive
  Host:www.tkf.toronto.on.ca

  reply

  HTTP/1.1 301 Moved Permanently
  Date: Tues, 23 July 2002
  Server: Apache/1.3.22(unix)
  Location: http://www.kites.org
  Content-Type: text/html

- So we need to go look up www.kites.org
Now what?

- Okay, we have the address.
- What do we do with it?
- Let’s look at how two computers communicate.
- HTTP is a high-level protocol
- HTTP is specific to the web.
- Computers communicate for other reasons.
Protocols

- Computers use several layers of general protocols to communicate.
- To understand why these layers are important, think about how a company sends you an invoice for a purchase.
Protocols

Invoice:
Customer: Karen Reid
Order No: 5379

<table>
<thead>
<tr>
<th>Qty</th>
<th>Unit</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Athalon</td>
<td>219.00</td>
<td>219.00</td>
</tr>
<tr>
<td>2</td>
<td>128 MB</td>
<td>149.95</td>
<td>299.90</td>
</tr>
<tr>
<td></td>
<td>Subtotal</td>
<td></td>
<td>518.90</td>
</tr>
<tr>
<td></td>
<td>Tax</td>
<td></td>
<td>77.84</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td></td>
<td>596.74</td>
</tr>
</tbody>
</table>

Payable to: CPUS are us $596.74
Five hundred ninety six 74/100

Karen Reid
Feb 18, 2001

We deliver!
## TCP/IP: Protocol Framework

<table>
<thead>
<tr>
<th>TCP/IP Layers</th>
<th>TCP/IP Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Layer</strong></td>
<td>HTTP</td>
</tr>
<tr>
<td></td>
<td>FTP</td>
</tr>
<tr>
<td></td>
<td>Telnet</td>
</tr>
<tr>
<td></td>
<td>SMTP</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
</tr>
<tr>
<td><strong>Transport Layer</strong></td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>UDP</td>
</tr>
<tr>
<td><strong>Network Layer</strong></td>
<td>IP</td>
</tr>
<tr>
<td></td>
<td>ARP</td>
</tr>
<tr>
<td></td>
<td>ICMP</td>
</tr>
<tr>
<td></td>
<td>IGMP</td>
</tr>
<tr>
<td><strong>Network Interface Layer</strong></td>
<td>Ethernet</td>
</tr>
<tr>
<td></td>
<td>Token Ring</td>
</tr>
<tr>
<td></td>
<td>Other Link-Layer Protocols</td>
</tr>
</tbody>
</table>
TCP/IP: Sending/Receiving Data

- **Application Layer**
  - Telnet, HTTP, FTP, SMTP

- **Transport Layer**
  - TCP, UDP

- **Network Layer**
  - IP

- **Physical Layer**
  - Ethernet, X.25, Token Ring
TCP/IP

- Transmission Control Protocol.
- Tells us how to package up the data.

Example analogy:
You can think that HTTP is what goes in the envelop, TCP is the envelop, and IP is the truck.
TCP Connection

3-way handshake
SYN

59
TCP: Three-way handshake

sequence number = J
socket
connect (blocks)
connect returns

client

server
sequence number = K
socket, bind, listen
accept (blocks)

SYN J
SYN K, ack J+1
ack K+1
accept returns
Each TCP packet is given a header:
- sequence number
- checksum

To make packets:

- Put in an IP envelope with another header.
When something goes wrong

- A packet might not arrive
  - traffic overload
  - bit corruption
- Receiver asks for missing packets to be resent. Want to send data as fast as possible.

- Strategies:
  - Send packets as fast as possible (too many lost)
  - Send packets at a certain rate (can go faster)
  - Wait for the ack (too slow)
  - Window-based (adaptive)
**TCP Congestion Control**

- **Window-based:**
  - Some number of packets allowed to be sent and not ack’d
  - As successful ack’s arrive, grow window
  - If packet loss is detected, cut window size
TCP Congestion Control

![Diagram showing how packet losses affect window size over time.](image)
Client-Server model: a client process wants to talk to a server process

Client must find server - **DNS lookup**

Client must find service on server - **ports**

Finally **establish a connection** so they can talk
Plus the redirect (20 hops)
At least 5 different cities

CSLab

U of T

U of T

Toronto, ON

Chicago

St. Louis, Missouri

Washington, DC

Scarborough, ON

Atlanta, Georgia

Chicago

??

??

??

??

??

??
Putting it together

```
www.kites.org
```

```
206.222.73.174
```

```
border.onet.on.ca
```

```
gate-gateway.gw.utoronto.ca
```

```
mcl-gpb.gw.utoronto.ca
```

```
“ca” name server
```

```
“toronto” name server
```

```
“on” name server
```

```
local name server
```

```
root name server
```

```
“org” name server
```

```
if-10-0.core1.Chicago3.Teleglobe.net
```

```
f-7-0.core1.Chicago3.Teleglobe.net
```

```
if-7-0.core1.Chicago3.Teleglobe.net
```

```
tbr1-p012201.n54ny.ip.att.net
```

```
tbr2-p013601.wswdc.ip.att.net
```

```
tbr1-p012201.n54ny.ip.att.net
```

```
tbr2-p013601.wswdc.ip.att.net
```

```
border.onet.on.ca
```

```
sf-cs27.gw.utoronto.ca
```

```
mcl-gpb.gw.utoronto.ca
```

```
gate-gateway.gw.utoronto.ca
```

```
if-3-2.core1.Toronto2.Teleglobe.net
```

```
if-3-2.core1.Toronto2.Teleglobe.net
```

```
if-10-0.core1.Scarborough.Teleglobe.net
```

```
f-7-0.core1.Chicago3.Teleglobe.net
```

```
12.122.10.29
```

```
12.122.10.74
```

```
12.124.65.22
```

```
12.122.10.29
```

```
gate-gateway.gw.utoronto.ca
```

```
mcl-gpb.gw.utoronto.ca
```

```
sf-cs27.gw.utoronto.ca
```

```
“org” name server
```

```
root name server
```

```
local name server
```

```
“ca” name server
```

```
“toronto” name server
```

```
“on” name server
```

```
“ca” name server
```

```
“on” name server
```

```
“toronto” name server
```

```
border.onet.on.ca
```

```
gate-gateway.gw.utoronto.ca
```

```
mcl-gpb.gw.utoronto.ca
```

```
sf-cs27.gw.utoronto.ca
```

```
sf-cs27.gw.utoronto.ca
```

```
“org” name server
```

```
root name server
```

```
local name server
```

```
“ca” name server
```

```
“toronto” name server
```

```
“on” name server
```

```
border.onet.on.ca
```

```
gate-gateway.gw.utoronto.ca
```

```
mcl-gpb.gw.utoronto.ca
```

```
sf-cs27.gw.utoronto.ca
```

```
sf-cs27.gw.utoronto.ca
```

```
“org” name server
```

```
root name server
```

```
local name server
```

```
“ca” name server
```

```
“toronto” name server
```

```
“on” name server
```
How many messages?

- It depends on the size of the web page we retrieve.
- If the web page is 100 Kbytes (small!) it will be broken up into ~80 IP packets.

10 (DNS) +
22 (Connect with toronto.on.ca) +
6 (DNS) +
+ 80 * 20 hops = 1638 messages!
Types of Connection (TCP/UDP)

- Connection oriented model
  - Like phone calls
  - Uses Transmission Control Protocol (TCP)
  - Defined ordering of messages and acks

- Connectionless model
  - Like sending letters via postal service
  - Uses User Datagram Protocol (UDP)
  - More efficient and good for sending broadcasts to many machines
CASCADING STYLE SHEETS (CSS)
Overview

- Basic CSS
- CSS Properties
- More CSS Syntax
Basic CSS
The good, the bad and the... ugly!

Tags such as `b`, `i`, `u`, and `font` are discouraged in strict XHTML.

- Why is this bad?
Cascading Style Sheets (CSS)

- Describes the **appearance**, **layout**, and **presentation** of information on a web page.
  - HTML describes **the content** of the page.
- Describes **how** information is to be displayed, **not** **what** is being displayed.
- Can be embedded in HTML document or placed into separate `.css` file.
Basic CSS rule syntax

- A CSS file consists of one or more **rules**
- Each rule starts with a **selector**
- A selector specifies an HTML element(s) and then applies style **properties** to them
  - a selector of * selects all elements
Aside: Favorites icon ("favicon")

```
<link href="filename" type="MIME type" rel="shortcut icon" />
```

```
<link href="yahoo.gif" type="image/gif" rel="shortcut icon" />
```

- The link tag, placed in the HTML page's head section, can specify an icon
  - this icon will be placed in the browser title bar and bookmark/favorite
Attaching a CSS file `<link>`

- A page can link to multiple style sheet files
  - In case of a conflict (two sheets define a style for the same HTML element), the latter sheet's properties will be used
Embedding style sheets: `<style>`

- CSS code can be embedded within the head of an HTML page.
- *Bad style* and should be avoided when possible (why?)
Inline styles: the style attribute

- Higher precedence than embedded or linked styles
- Used for one-time overrides and styling a particular element
- Bad style and should be avoided when possible (why?)
Cascading Style Sheets

- Properties of an element cascade together in this order (precedence):
  - browser's default styles
  - external style sheet files (in a `<link>` tag)
  - embedded/internal style sheets (inside a `<style>` tag in the page's header)
  - inline style (the style attribute of the HTML element)
W3C CSS Validator

CSS output

- jigsaw.w3.org/css-validator/
- checks your CSS to make sure it meets the official CSS specifications
CSS Properties
CSS properties for colors

```css
p {
  color: red;
  background-color: yellow;
}
```

This paragraph uses the style above

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>color of the element's text</td>
</tr>
<tr>
<td>background-color</td>
<td>color that will appear behind the element</td>
</tr>
</tbody>
</table>
Specifying colors

CSS

```css
p { color: red; }
h2 { color: rgb(128, 0, 196); }
h4 { color: #FF8800; }
```

This paragraph uses the first style above

This h2 uses the second style above.

This h4 uses the third style above.

- **color names**: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white (white), yellow
- **RGB codes**: red, green, and blue values from 0 (none) to 255 (full)
- **hex codes**: RGB values in base-16 from 00 (0, none) to FF (255, full)
Grouping styles

```css
p, h1, h2 {
  color: green;
}

h2 {
  background-color: yellow;
}
```

This paragraph uses the above style.

**This h2 uses the above styles.**

- A style can select multiple elements separated by commas
- The individual elements can also have their own styles
CSS comments /*...*/

/* This is a comment. It can span many lines in the CSS file. */
p {
color: red; background-color: aqua;
}

- CSS (like HTML) is usually not commented as rigorously as programming languages such as Java
- The // single-line comment style is NOT supported in CSS
- The <!-- ... --> HTML comment style is also NOT supported in CSS
CSS properties for fonts

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>font-family</td>
<td>which font will be used</td>
</tr>
<tr>
<td>font-size</td>
<td>how large the letters will be drawn</td>
</tr>
<tr>
<td>font-style</td>
<td>used to enable/disable italic style</td>
</tr>
<tr>
<td>font-weight</td>
<td>used to enable/disable bold style</td>
</tr>
</tbody>
</table>

Complete list of font properties

[Complete list of font properties](http://www.w3schools.com/css/css_font.asp)
font-family

```css
p {
  font-family: Georgia;
}

h2 {
  font-family: "Courier New";
}
```

- Enclose multi-word font names in quotes
More about font-family

```css
p {
  font-family: Garamond, "Times New Roman", serif;
}
```

output

- We can specify multiple fonts from highest to lowest priority
- Generic font names:
  - serif, sans-serif, cursive, fantasy, monospace
- If the first font is not found on the user's computer, the next is tried
- Placing a generic font name at the end of your font-family value, ensures that every computer will use a valid font
font-size

```css
p {
  font-size: 24pt;
}
```

units: pixels (px) vs. point (pt) vs. m-size (em)

16px, 16pt, 1.16em

vague font sizes: xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger

percentage font sizes, e.g.: 90%, 120%
font-size

```css
p {
    font-size: 24pt;
}
```

output

- **pt** specifies a number of points, where a point is 1/72 of an inch onscreen.
- **px** specifies a number of pixels on the screen.
- **em** specifies number of m-widths, where 1 em is equal to the font's current size.
font-weight, font-style

Either of the above can be set to normal to turn them off
# CSS properties for text

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text-align</td>
<td>alignment of text within its element</td>
</tr>
<tr>
<td>text-decoration</td>
<td>decorations such as underlining</td>
</tr>
<tr>
<td>line-height,</td>
<td>gaps between the various portions of the text</td>
</tr>
<tr>
<td>word-spacing,</td>
<td></td>
</tr>
<tr>
<td>letter-spacing</td>
<td></td>
</tr>
<tr>
<td>text-indent</td>
<td>indents the first letter of each paragraph</td>
</tr>
</tbody>
</table>

[Complete list of text properties](http://www.w3schools.com/css/css_text.asp)
text-align

blockquote { text-align: justify; }  
h2 { text-align: center; }

CSS

output

text-align can be left, right, center, or justify
text-decoration

```css
p {
  text-decoration: underline;
}
```

- can also be **overline**, **line-through**, **blink**, or **none**
- effects can be combined:

```css
text-decoration: overline underline;
```
The list-style-type property

```css
ol { list-style-type: lower-roman; }
```

- Possible values:
  1. none: No marker
  2. disc (default), circle, square
  3. Decimal: 1, 2, 3, etc.
  4. decimal-leading-zero: 01, 02, 03, etc.
  5. lower-roman: i, ii, iii, iv, v, etc.
  6. upper-roman: I, II, III, IV, V, etc.
  7. lower-alpha: a, b, c, d, e, etc.
  8. upper-alpha: A, B, C, D, E, etc.
  9. lower-greek: alpha, beta, gamma, etc.
  10. others: hebrew, armenian, georgian, cjk-ideographic, hiragana...
Body styles

- Applies a style to the entire body of your page
- Saves you from manually applying a style to each element

```css
body {
    font-size: 16px;
}
```
Inheriting styles

- A bulleted list

- when multiple styles apply to an element, they are inherited
- a more tightly matching rule can override a more general inherited rule

This is a heading

A styled paragraph. Previous slides are available on the website.
when two styles set **conflicting** values for the same property, the latter style takes precedence
# CSS properties for backgrounds

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-color</td>
<td>color to fill background</td>
</tr>
<tr>
<td>background-image</td>
<td>image to place in background</td>
</tr>
<tr>
<td>background-position</td>
<td>placement of bg image within element</td>
</tr>
<tr>
<td>background-repeat</td>
<td>whether/how bg image should be repeated</td>
</tr>
<tr>
<td>background-attachment</td>
<td>whether bg image scrolls with page</td>
</tr>
<tr>
<td>background</td>
<td>shorthand to set all background properties</td>
</tr>
</tbody>
</table>
background-image

```css
body {
  background-image: url("images/draft.jpg");
}
```

- background image/color fills the element's content area
background-repeat

- can be repeat (default), repeat-x, repeat-y, or no-repeat
value consists of two tokens, each of which can be top, left, right, bottom, center, a percentage, or a length value in px, pt, etc.

value can be negative to shift left/up by a given amount