CISC 1600 Midterm Review

Topics:

Lec 1.1: Introduction, HTML5

Lab 1.1: HTML5

Lec 1.2: CSS

Lab 1.2: CSS

Lec 1.3-4: Web design

Lec 1.5: Internet and WWW

Lec 2.1: Intro to Processing

Lab 2.1: Intro to Processing

CISC 1600 Lecture 1.1
Introduction

Topics:

Multimedia computing
Course information
HTML5
Markup languages

Multimedia Computing at Brooklyn College

Brooklyn College, The City University of New York

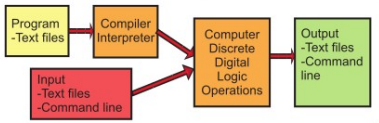
Dept of Computer and Information Science
http:/ / www.sci.brooklyn.cuny.edu/ cis

BS in Multimedia Computing
Professor Elizabeth Sklar, Director
sklar@sci.brooklyn.cuny.edu

Introductory Programming Sequence

(CISC 1110 + CISC 3110)

Introductory programming courses tend to limit input and output to text files and command line data. This is intentional as it allows students and instructors to focus on the fundamentals of computer programming.



Multimedia Computing Courses

CISC 1600 Intro to Multimedia Computing
(web design, interface design, animation, game design, data visualization, simulations.)

CISC 3610 Intro to Multimedia Programming
(in-depth multimedia programming and authoring using Flash and Actionscript.)

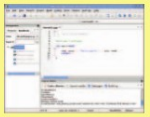
CISC 3630 Multimedia Computing
(multimedia hardware and software platforms, media types and formats, compression techniques.)

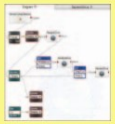
CISC 4610 Multimedia Databases
(storage and organization of multimedia data.)

Multimedia Computing

Programming

Advanced IDE - IDE (Integrated development environments) programs facilitate creation of programs through use of graphical tools.





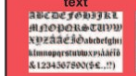
Visual Programming Language (VPL) - VPLs let users create programs by manipulating program components (objects, functions, variables) graphically rather than by specifying them with text.

Many VPLs are targeted towards beginner programmers who have only a basic understanding of concepts like variables and logic. However, VPLs are not limited to novices. VPLs may appeal to more advanced programmers for rapid prototyping or code development. VPLs are also often well suited to programming within a variety of concurrent or distributed processing scenarios. VPLs thus appeal to a wide audience of users from students to professionals.

Input

Multimedia computing incorporates the rich array of possible computer input and moves beyond text and command-line limitations.

text



audio

2D/3D data constructs

devices

images

video

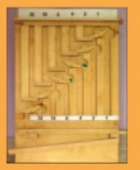
analog and digital signals

Multimedia computing recognizes that input media...
-can be **TRANSFORMED**; a scanned page may become text, which is then rendered as audio output for the visually impaired.
-can result from the **FUSION** of many types of input; a game level may be composed of text, images, sounds, video clips, and other data.

Computer

As both research and art project students and scientists have been exploring alternatives to the traditional silicon computer:

Analog Computers (RICE)
Water Computers (MIT)
Mechanical Computers
(Digi-Comp) (Marble Adder)



Output

Multimedia computing allows for output in forms that...
-facilitate **CLARITY** (Presentations)
-improve **COMPREHENSION** (Visualizations)
-help manage **COMPLEXITY** (Simulations)
-directly **CONTROL** objects in the environment

text

audio

2D/3D data constructs

images

video

simulations

visualizations

webpages

interactive computing:
adding interactivity ("scripting" languages)

affordances: output that informs
users about input possibilities ("Click Here")

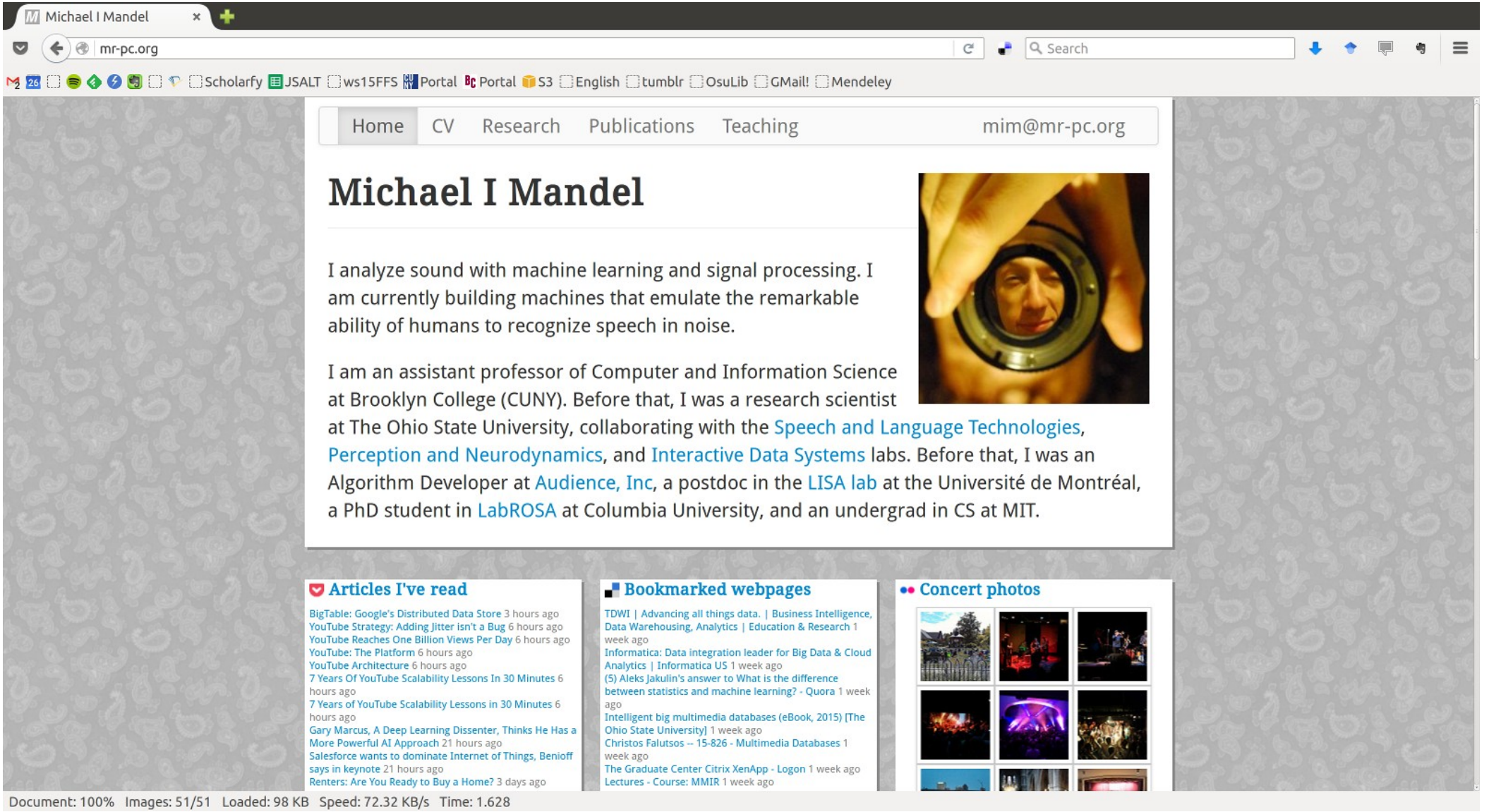
analog & digital signals

Devices
Simple Machines, Robotics, Biological Interfaces

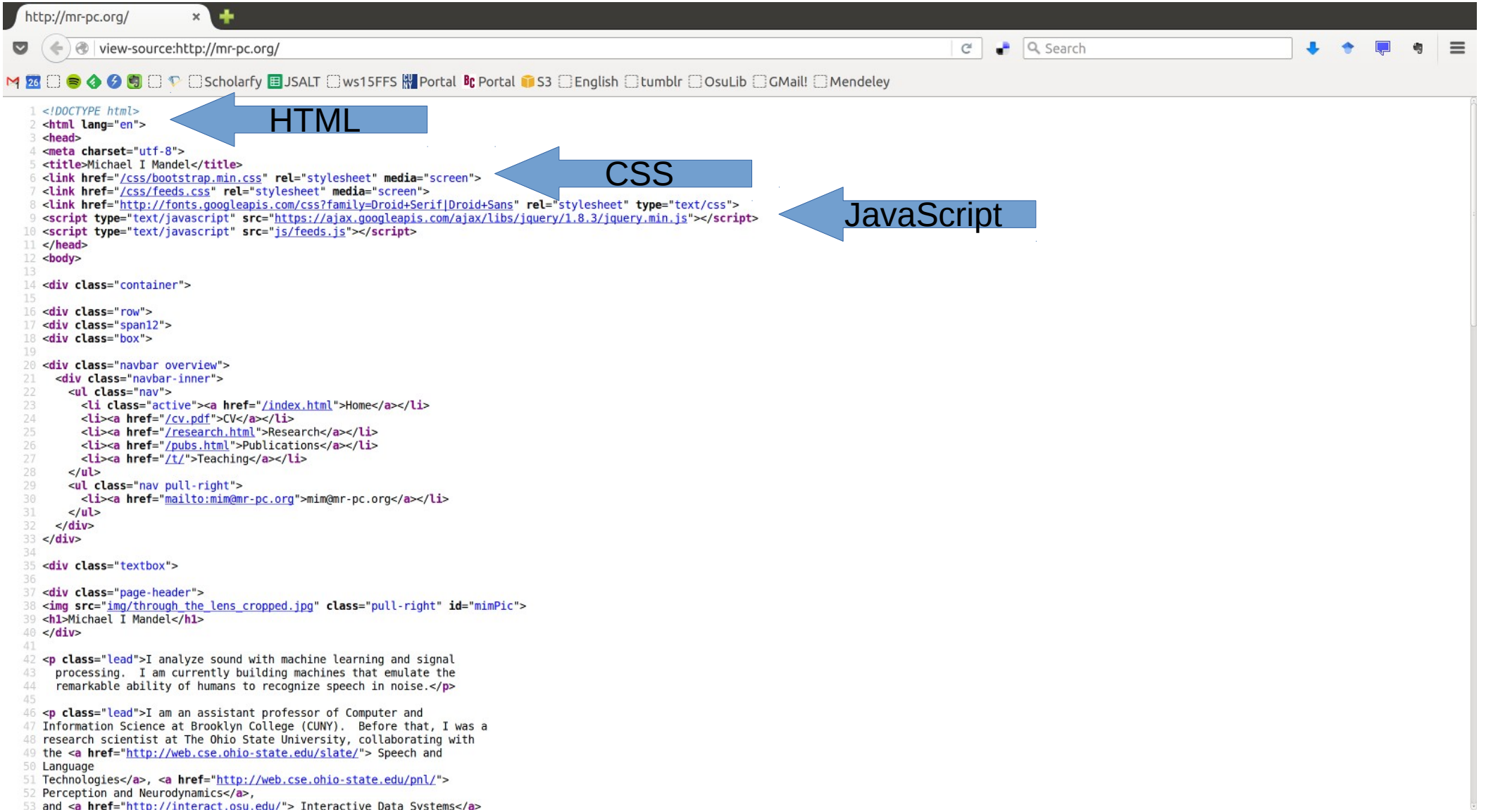
HTML5

- Latest web standard for HyperText Markup Language (HTML)
  - Supported on all major platforms: computers, tablets, phones, watches, etc.
  - Native browser support for multimedia data
- Three main components
  - HTML: content, structure
  - CSS: appearance
  - JavaScript: interactivity

Example HTML5 website



Example HTML5 website




```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Michael I Mandel</title>
<link href="/css/bootstrap.min.css" rel="stylesheet" media="screen">
<link href="/css/feeds.css" rel="stylesheet" media="screen">
<link href="http://fonts.googleapis.com/css?family=Droid+Serif|Droid+Sans" rel="stylesheet" type="text/css">
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<script type="text/javascript" src="js/feeds.js"></script>
</head>
<body>

<div class="container">

<div class="row">
<div class="span12">
<div class="box">

<div class="navbar overview">
  <div class="navbar-inner">
    <ul class="nav">
      <li class="active"><a href="/index.html">Home</a></li>
      <li><a href="/cv.pdf">CV</a></li>
      <li><a href="/research.html">Research</a></li>
      <li><a href="/pubs.html">Publications</a></li>
      <li><a href="/t/">Teaching</a></li>
    </ul>
    <ul class="nav pull-right">
      <li><a href="mailto:mim@mr-pc.org">mim@mr-pc.org</a></li>
    </ul>
  </div>
</div>

<div class="textbox">

<div class="page-header">
<img src="img/through_the_lens_cropped.jpg" class="pull-right" id="mimPic">
<h1>Michael I Mandel</h1>
</div>

<p class="lead">I analyze sound with machine learning and signal
  processing.  I am currently building machines that emulate the
  remarkable ability of humans to recognize speech in noise.</p>

<p class="lead">I am an assistant professor of Computer and
Information Science at Brooklyn College (CUNY).  Before that, I was a
research scientist at The Ohio State University, collaborating with
the <a href="http://web.cse.ohio-state.edu/slate/"> Speech and
Language
Technologies</a>, <a href="http://web.cse.ohio-state.edu/pnl/">
Perception and Neurodynamics</a>,
and <a href="http://interact.osu.edu/"> Interactive Data Systems</a>
```

HTML Key Terminology

Tag: A markup that constitutes an instruction to an interpreting program, and is not part of the text being marked up.

Element: If a document can be converted into a “tree”-like representation, as HTML can, then an element is a “node” in the “tree”.

Attribute: A markup signifying a property of an element.

HTML Key Terminology: Tag

Tag: A markup that constitutes an instruction to an interpreting program, and is not part of the text being marked up.

- In HTML, tags begin with "<" and end with ">"
- They come in three flavors:
  - start-tags, for example `<p>`
  - end-tags, for example `</p>`
  - and empty-element tags, for example `<img/>`

HTML Key Terminology: Element

Element: If a document can be converted into a “tree”-like representation, as HTML can, then an element is a “node” in the “tree”.

- In HTML, an element begins with a start-tag and ends with a matching end-tag
  - or consists only of an empty-element tag
- Characters between the start- and end-tags are the element’s content
  - May include other elements, which are called child elements
- An example of an element is `<p>Hello, world.</p>`
- Another is `<br/>`

HTML Key Terminology: Attribute

Attribute: A markup signifying a property of an element.

- In HTML, attributes consist of a name/value pair within a start-tag or empty-element tag

```
<img src="button.jpg" alt='button'/>
```

- Here the element `img` has two attributes, `src` and `alt`

HTML5 Skeleton

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Prof Mandel's App</title>
</head>
<body>
    <h1>Hello world!</h1>
    <p>This is the first paragraph</p>
</body>
</html>
```

CISC 1600 Lecture 1.2
Introduction

Topics:

Markup

HTML box model

Document object model (DOM)

Cascading style sheets (CSS)

Taking a step back:

- What's a language?
  - Verbal languages
  - Written languages
  - Visual languages
  - Programming languages
  - Markup languages
- Simple Answer: "A medium for sharing / exchanging information"

Markup Languages (defined).

From: http://en.wikipedia.org/wiki/Markup_language

“A markup language is a system for annotating a document in a way that is syntactically distinguishable from the text. The idea and terminology evolved from the "marking up" of paper manuscripts, i.e., the revision instructions by editors, traditionally written with a blue pencil on authors’ manuscripts.”

Types of Markup (HTML does all three)

- **Presentational markup**: Used by traditional word-processing systems, to create a WYSIWYG effect. Examples: add a line break, bold a word, change font style or color.
- **Procedural markup**: Provides instructions for programs that are to process the text. Examples: add an image, video, or link to a document.
- **Semantic markup**: Used to label parts of a document and attach additional meaning to those sections. Examples: define the title of a document or declaring that a section of text is an address.

Back to HTML and CSS

Every HTML element is a box

Hello world!

This is the first paragraph

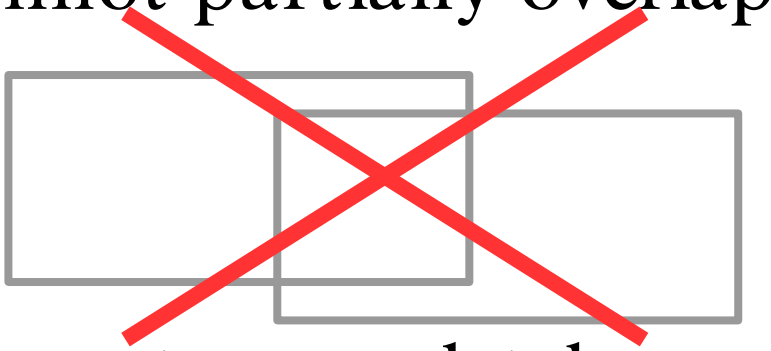
Every HTML element is a box

Hello world!

This is the first paragraph

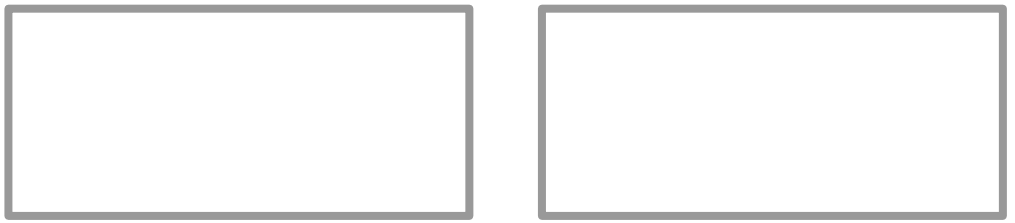
Boxes cannot partially overlap

- Two boxes cannot partially overlap with each other



- They either have to completely overlap

- Or be completely separate



Boxes can be organized into a tree

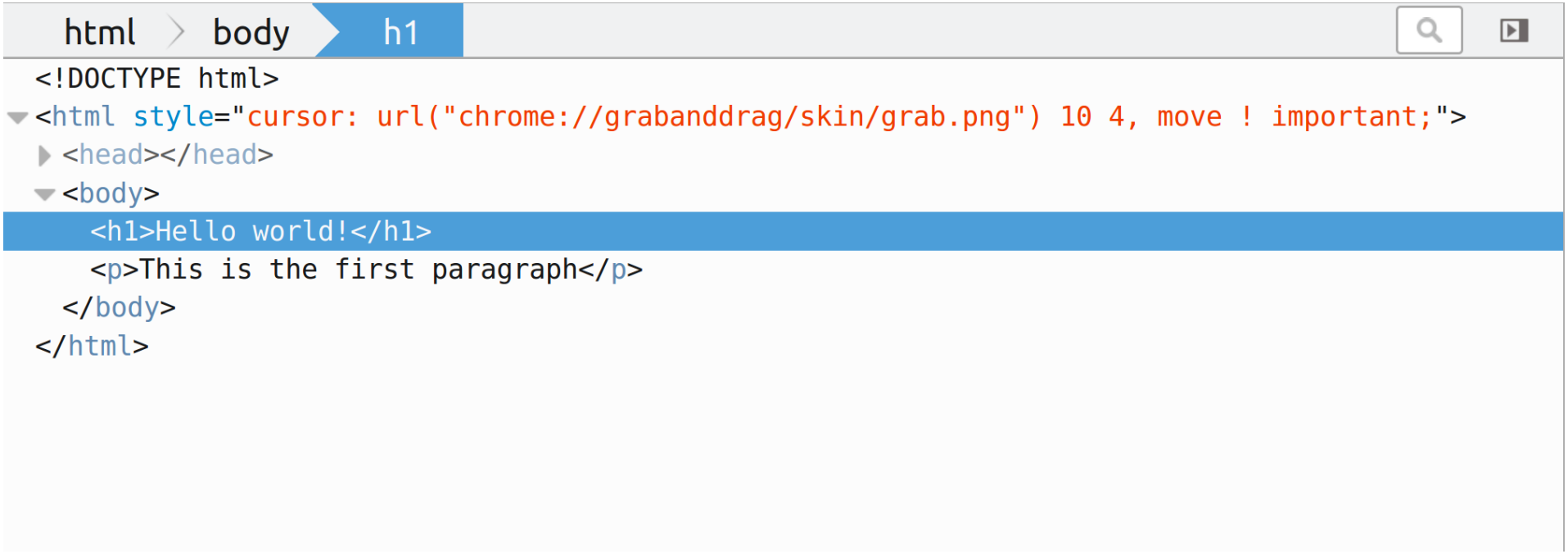
Hello world!

This is the first paragraph

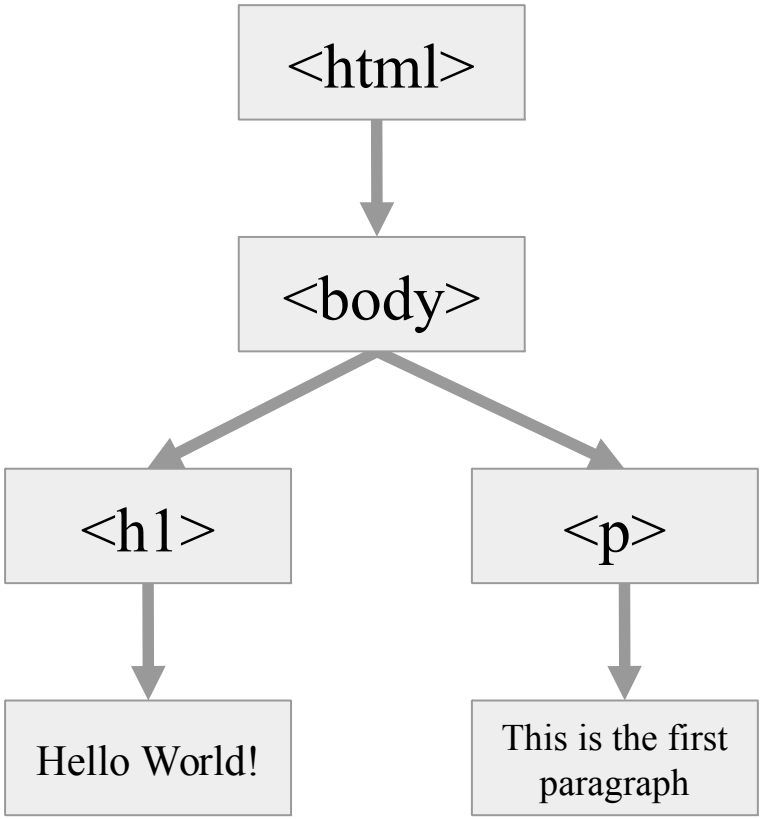
Document Object Model (DOM)

- Because the boxes (elements) can't overlap, they can be put into a hierarchy
  - The hierarchy can be represented as a "tree"
  - The DOM tree
- When building a page, consider this structure first, i.e., the DOM
- DOM tree can be manipulated by CSS and javascript after it is built

Example DOM


html
body
h1
<!DOCTYPE html>
<html style="cursor: url("chrome://grabanddrag/skin/grab.png") 10 4, move ! important;">
<head></head>
<body>
<h1>Hello world!</h1>
This is the first paragraph
</body>
</html>

Example DOM



Cascading style sheets (CSS)

- CSS provides a way to style DOM elements
  - Different syntax from HTML
- “Cascading” because rules cascade down the DOM tree from root to leaf
- Controls both styling and positioning

CSS Syntax

- General form of a declaration:

```
selector { property: value; property: value; }
```

- Selector is usually the name of a tag
- Braces are curly { }
- Properties are separated from values by colons :
- Property-value pairs are terminated with semicolons ;
- White space is ignored

CSS example declarations

- Simple:

```
p { margin: 10pt; }
```

- Multiple properties in one block:

```
h1 {
     font-family: Verdana, sans-serif;
     color: red;
     font-size: 20px;
  }
```

- Multiple selectors, relative sizing:

```
p, div, h2 { color: #00ddff ; width: 80%; }
```

There are three ways to include CSS in HTML

- Include an external CSS file
- Include CSS declarations directly in the head of your document
- Include CSS declarations in style attribute of individual elements
- Can use any combination
  - the “closest” definition is used

Include an external file

- Use a link tag in the head of your HTML document:

```
<link rel="stylesheet" type="text/css" href="mystyles.css" />
```

  - “rel” stands for ‘RELationship’
  - type shows that it’s a text file acting as a CSS stylesheet

- You can use this tag multiple times in the same document
  - To link multiple stylesheets to the page
  - e.g., one file for fonts, another for margins and spacing

Include CSS declarations directly in the head of your document

- Put a style element into the head of your file

```
<style type="text/css">
    p { font-weight: bold; color: gray; }
    h1 { color: black; }
</style>
```

- The type attribute again tells the browser to interpret this as CSS
- Called "inline style block"

Include CSS declarations in style attribute of individual elements

- Put a style attribute in the opening tag:

```
<p style="color: blue; font-family: Ariel"> … </p>
```

- Only applies within that element (& children)
- Note the lack of curly braces here, but the colons and semicolons are still necessary
- You shouldn't need to use this

Don't repeat yourself

- I like to be efficient (and I'm a little lazy)
- I don't want to have to solve a problem that I have solved before
- I want to **reuse** my previous solution
  - The one that I've already debugged
- The great thing about computer science is that you can do exactly that

Don't repeat yourself When styling a website

- Cascading style sheets allow us to define styles once per website
  - And then reuse those definitions on all of its pages
- Changes to the centralized style will automatically be propagated to all pages
- The CSS file can be downloaded once by the browser and saved to speed up page loading

Separation of concerns: presentation vs content

- Content is the “what” of the document:
  - text, media (e.g., images), structure
- Presentation is the “how”
  - How should the page be displayed to the user?
- They are only loosely coupled to each other
- Different people could be responsible for each
- This is a general example of the concept of a “separation of concerns”

Classes and IDs

- How would you style different instances of the same element differently?
  - Paragraphs in the main article vs in the sidebar
- Or how would you style different non-nested tags the same?
- Use the “class” and “id” attribute for any tag

Classes

- In the HTML file, classes are attributes:

```
<p class=”caution”> … </p>
```

- In the CSS file, classes are selected by preceding them with a dot:

```
.caution { font-size: 200%; color: red; }
```

- Multiple tags in the same page can have the same class

IDs

- IDs are almost the same as classes
- But there can be at most one element in a single HTML page with a given ID
- An ID is a unique identifier of that element
- In the HTML:

```
<ul>
      <li id=”item1”> … </li>
      <li id=”item2”> … </li>
</ul>
```

- In the CSS:

```
#item1 { background: gray; }
```

Span and div tags

- With classes and IDs as selectors, we could just use the same tag for everything with different classes:

```
<div class="big-red-box">
    <span class="caution"> Warning:</span> wrong password
</div>
```

- `<div>` acts like the boxes we saw earlier
- `<span>` lets you apply a class/id to running text
- While possible, it's better to use new HTML semantic tags, which are standardized and give the browser more information

CSS files can be validated

- Can check validity of CSS files and blocks

http://jigsaw.w3.org/css-validator

- Will flag:
  - Syntax errors
  - Invalid properties
  - Invalid values
  - Etc.

CISC 1600, Lecture 1.3
Design

Topics:

User-centered design

Usability and visual design

Accessibility

Responsive web design

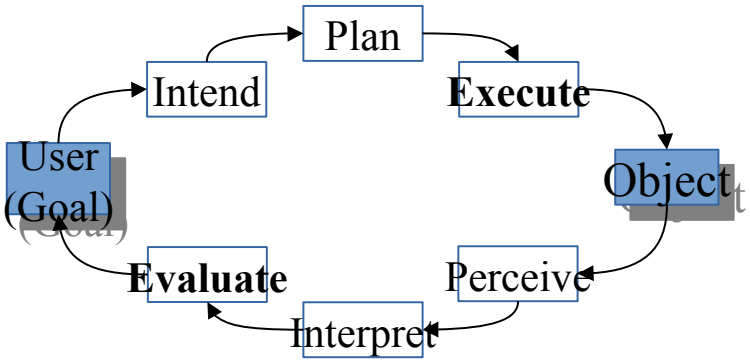
Design review

User-centered design

The user wants to do something

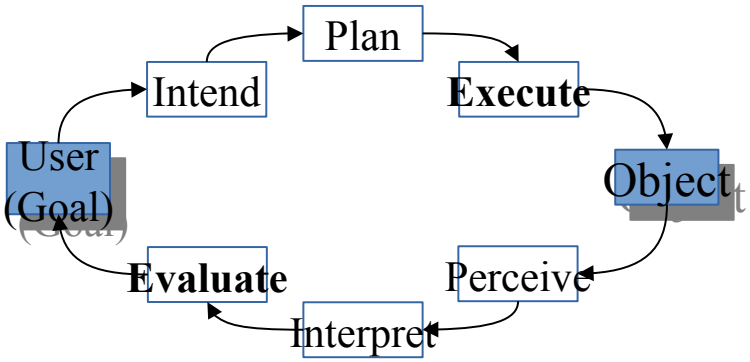
- The user has a **goal**, something they want to make happen in the world
- Interfaces are the means by which they can
  - Actually, the means by which they **must**
- Good interfaces make it easy
- Building good interfaces is the domain of interaction designers & user experience designers
- This applies equally to objects and web pages

Do what you want, know that you have



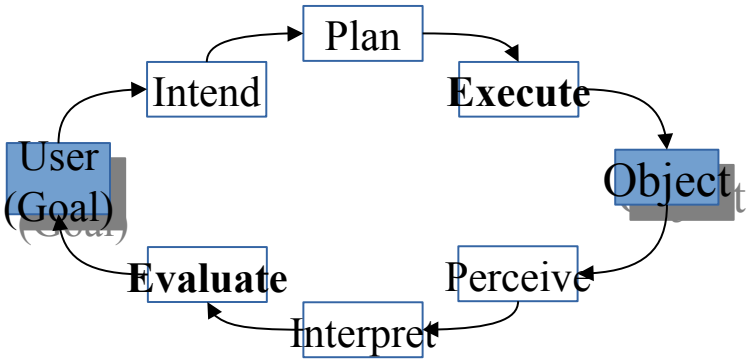
- Goal: What the user wants to do
- Intend: Determine possible ways of satisfying goal
- Plan: Map from intention to action
- Execute: Perform the action
- Perceive: Determine the current state of the system
- Interpret: Make sense of the state of the system
- Evaluate: Determine if that is the desired state

Example: make a nice breakfast



- Goal: What the user wants to do
- Intend: Determine possible ways of satisfying goal
- Plan: Map from intention to action
- Execute: Perform the action
- Perceive: Determine the current state of the system
- Interpret: Make sense of the state of the system
- Evaluate: Determine if that is the desired state

Example: make a nice breakfast



- Goal: Make a nice breakfast
- Intend: Cook a fancy omelet
- Plan: beat eggs, chop vegetables, defrost sausage
- Execute: actually do those things
- Perceive: how does it look, smell, taste?
- Interpret: does it seem like a fancy omelet?
- Evaluate: was that a nice meal?

Leads to Norman's principles of good design

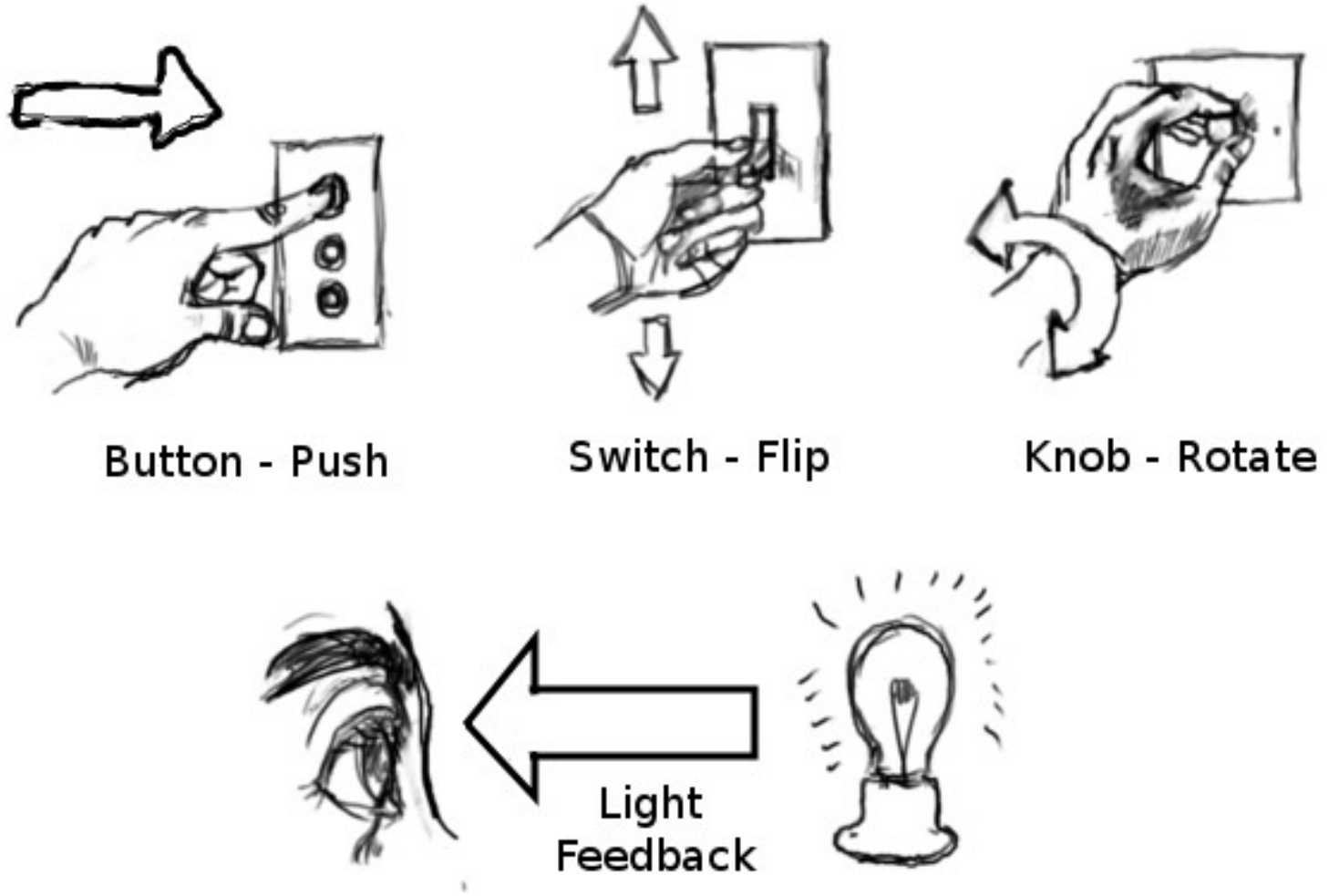
- Well-defined **interfaces** facilitate this process
- Visibility: easy to identify possibilities for action ("affordances")
- Conceptual model: easy to relate to known experiences / objects / metaphors
- Good mappings: easy to determine mapping from actions to results, controls to effects
- Feedback: easy to set the object's state

Affordances for door interfaces

From: http://en.3cbang.com/view/19925.html

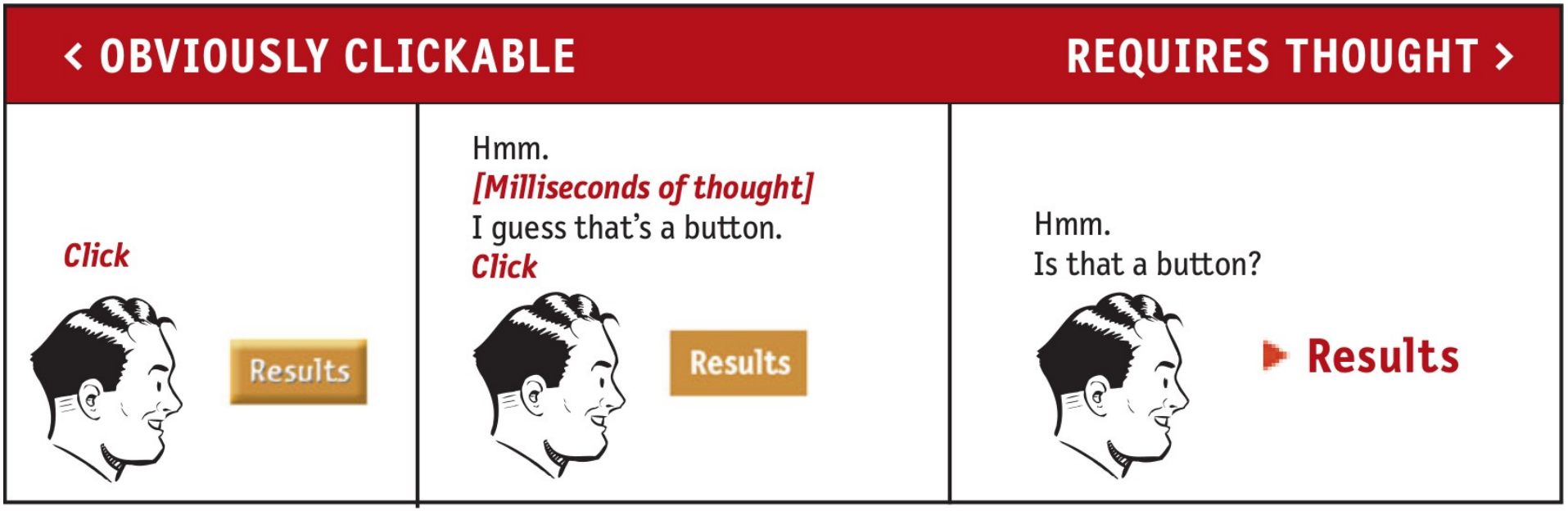
Affordances for car interfaces

Affordances for lighting interfaces



From: http://paaralan.blogspot.com/2010/09/affordance-and-educational-games.html

Affordances on the web



From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Other digital affordances

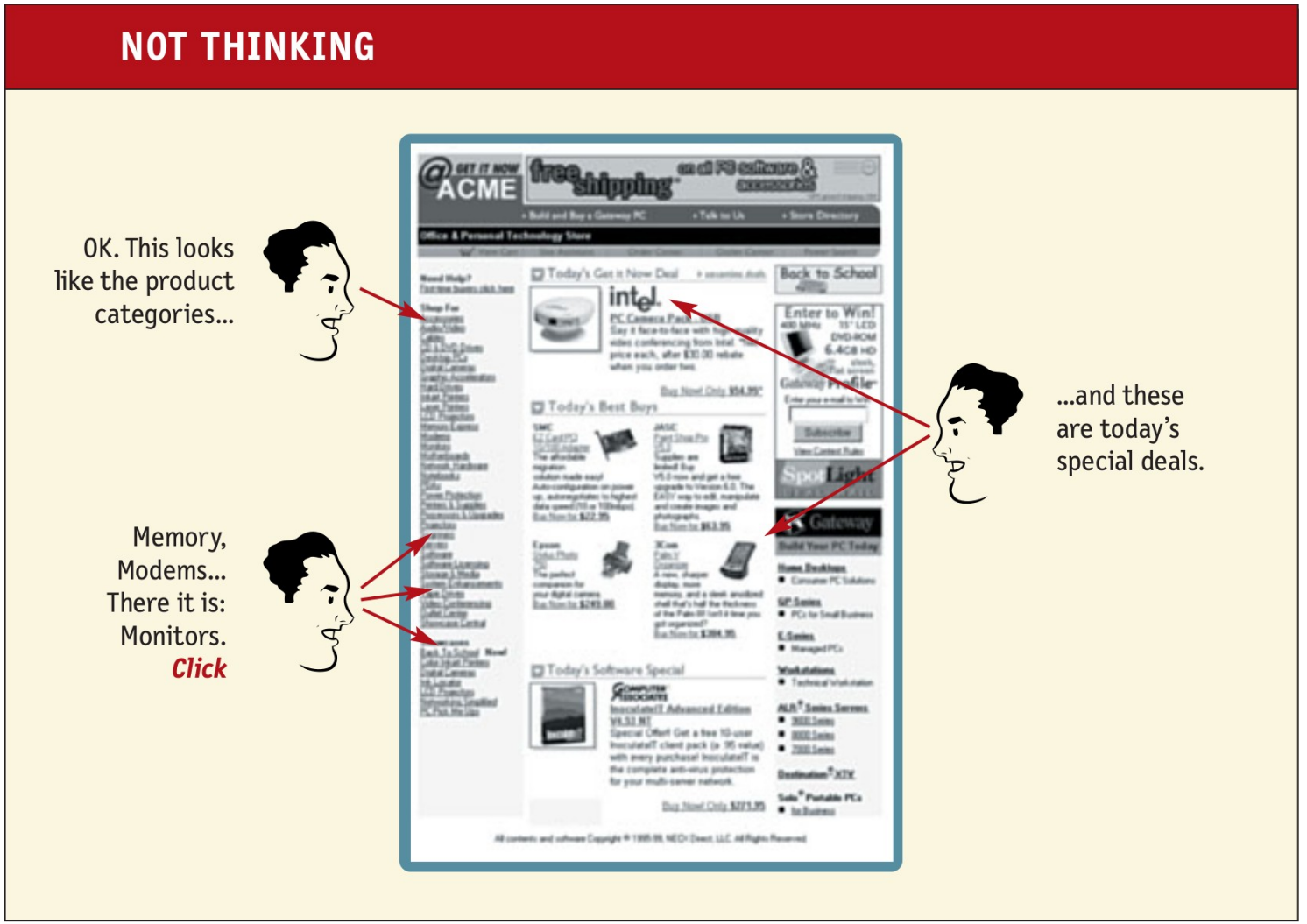
- User-interface elements
  - Arrows indicate drop-down menus, scrolling, etc
  - Standard icons indicate standard functions
- Changing cursor indicates clickable, draggable
- Half a picture (the “fold”) indicates scrollable

Usability and visual design

- The easiest interface to use is the one the user expects: consistency
- Visual hierarchy indicates what is important
- Consistency reinforces a sense of “place”
- Aesthetics convey information about the site and the viewer
  - E.g., color can convey emotion

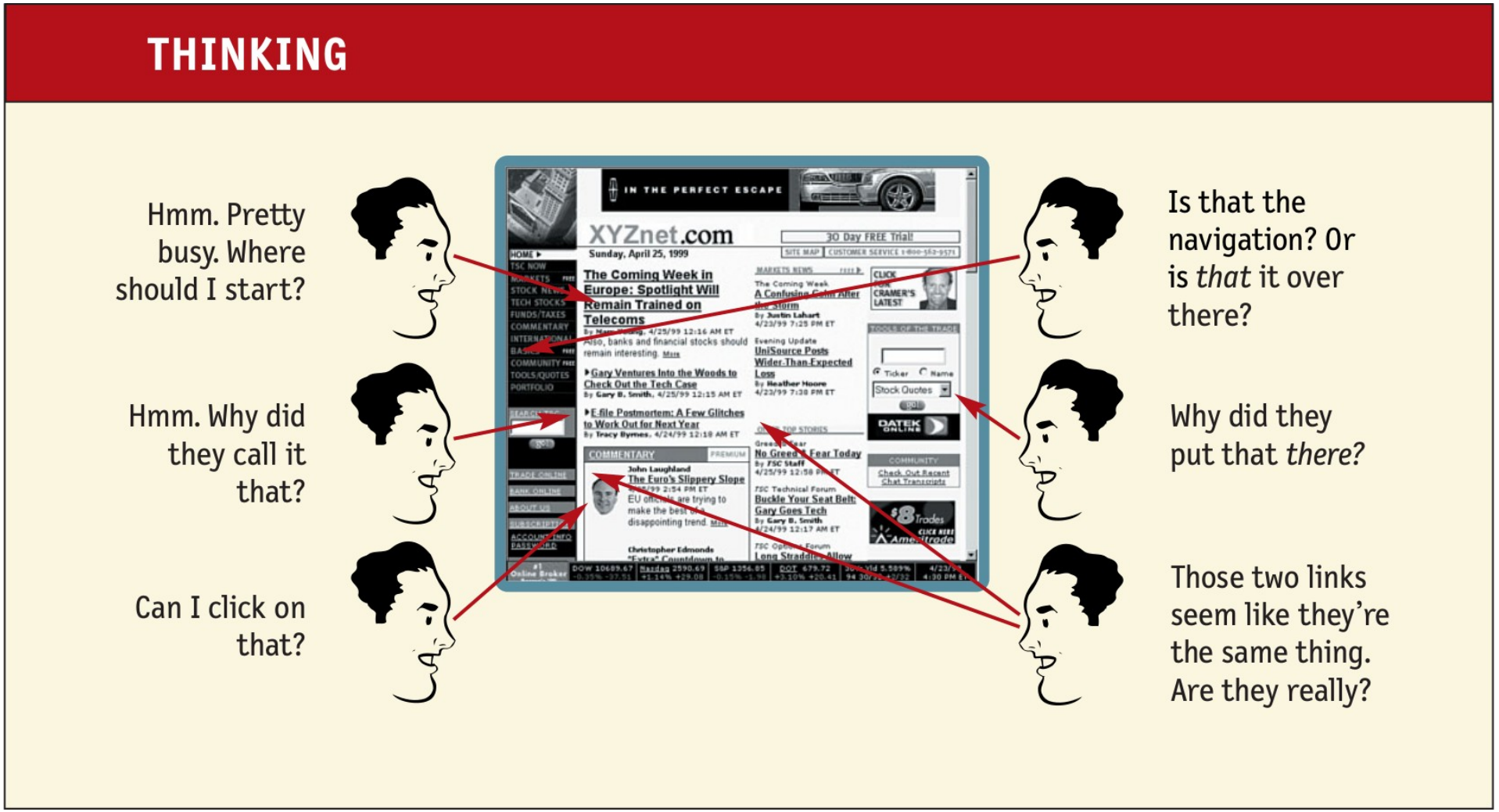
Usability and visual design

Doesn't make you think: good



From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Makes you think: bad



From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Conventional names don't make you think

From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Conventional visual cues don’t make you think (affordances)

< OBVIOUSLY CLICKABLE | REQUIRES THOUGHT >

Click

Results

Hmm.
[Milliseconds of thought]
I guess that’s a button.
Click

Results

Hmm.
Is that a button?

Results

From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Visual hierarchies help us scan a newspaper

The headline spanning these three columns makes it obvious that they're all part of the same story.

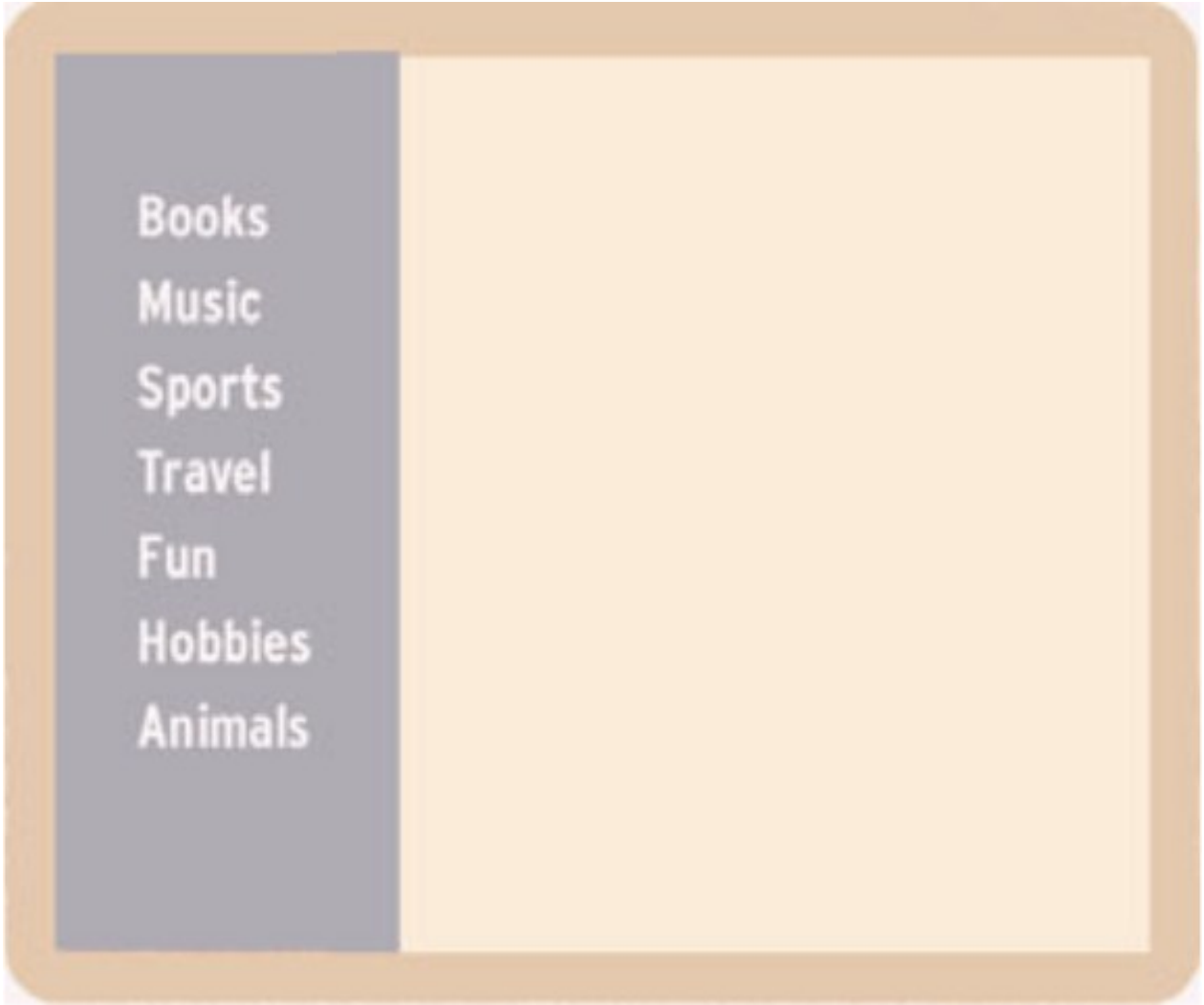
The size of this headline makes it clear at a glance that this is the most important story.

From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

The more important something is the more prominent it should be

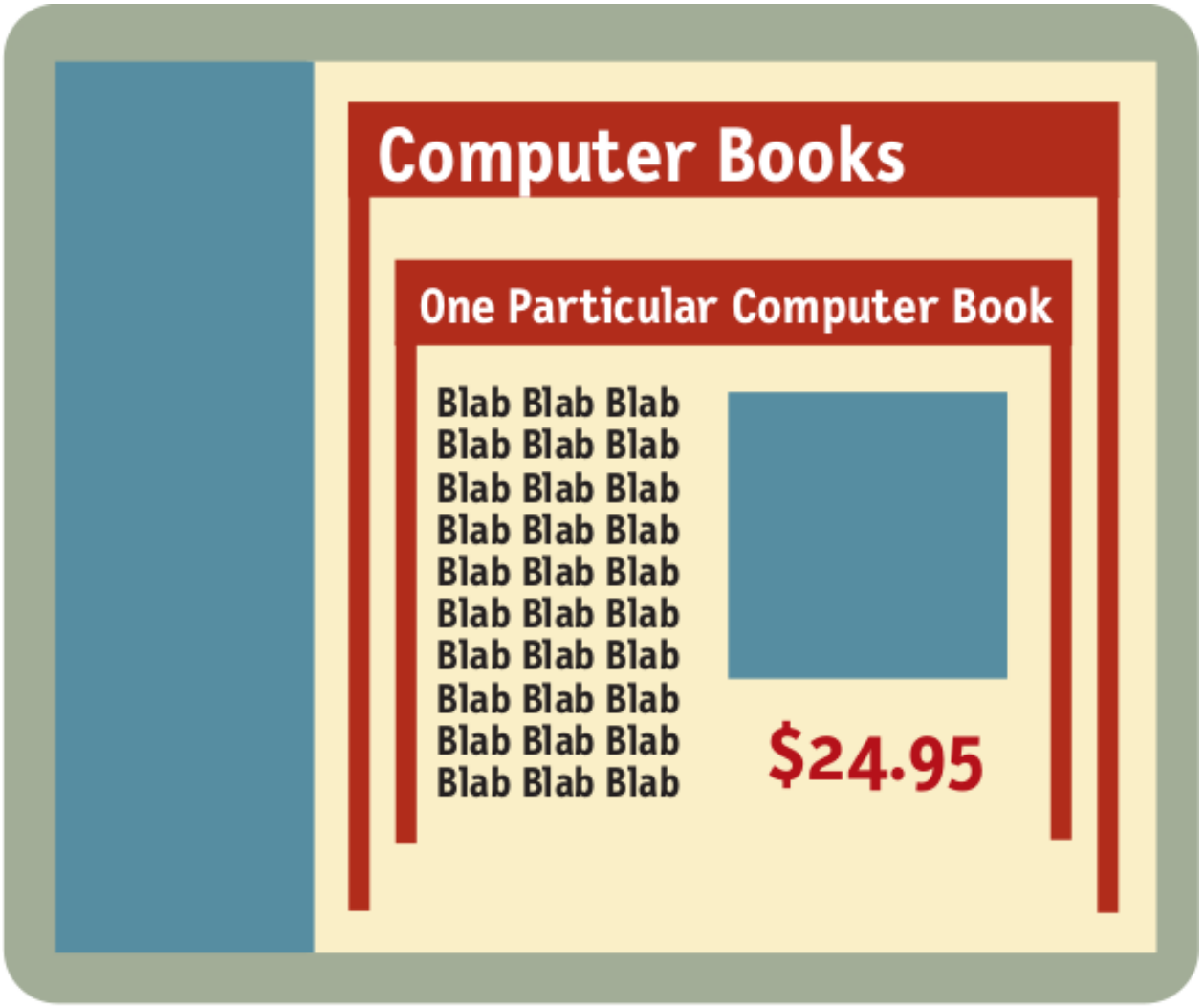
From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Things that are related logically should also be related visually



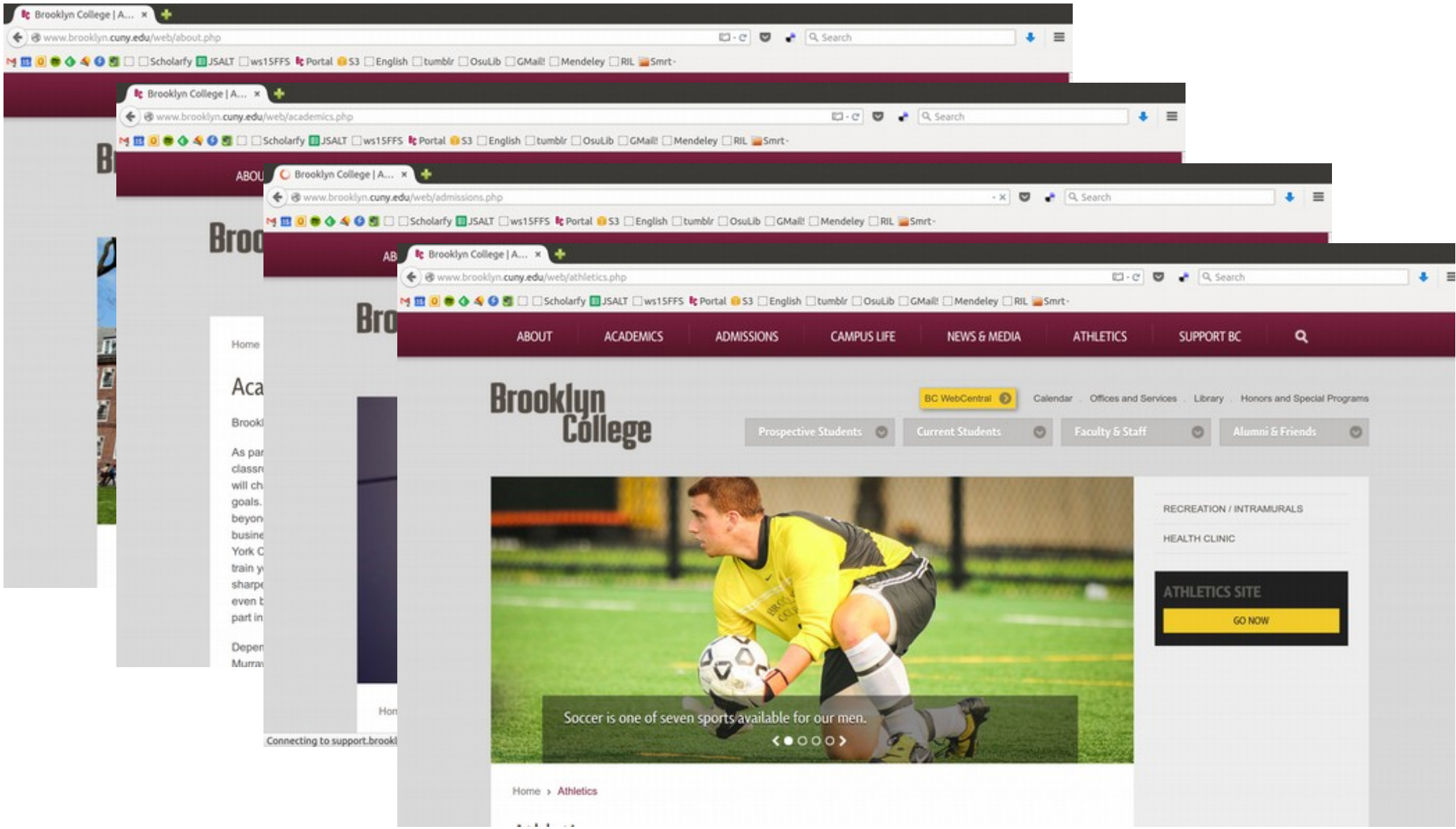
From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Things should be nested visually to show parts of wholes



From: Krug, Steve. Don't make me think! A common sense approach to web usability, 2nd ed. 2006. New Rider Press.

Consistency reinforces a sense of “place”



Color can convey emotion

From: http://www.color-wheel-pro.com/color-meaning.html

Accessibility

- Design for users with special needs
- Might include you at some point
  - You still want/need to use your favorite websites
- Many issues easy to solve with a little care
  - Standard, best-practice solutions exist

Custom formatting

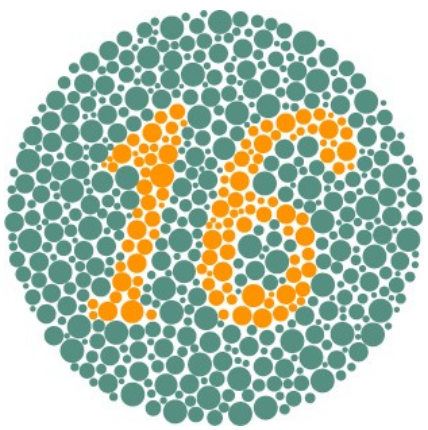
- Visually impaired users may prefer
  - Increased font sizes
  - Increased contrast
- Accessibility considerations
  - Avoid text in images (can't be resized)
  - Use CSS, separate presentation from content
- How to check in Firefox
  - Text size: control-plus, control-minus
  - Contrast: Preferences → Content → Fonts & Colors
    - Or set high contrast mode on your OS

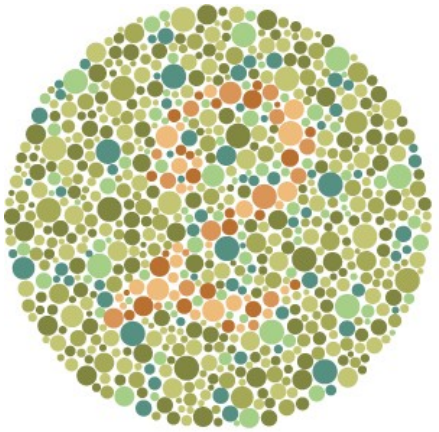
Screen reader

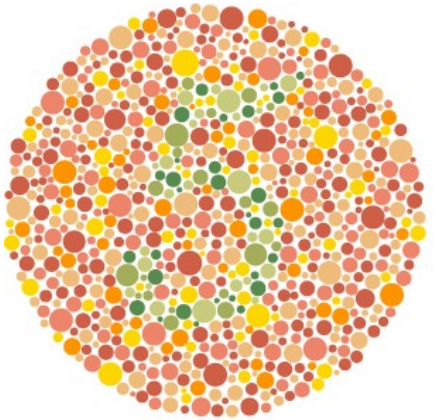
- Visually impaired users can use text-to-speech programs to read interfaces to them
- Generally turn the speaking rate up very fast
- Accessibility considerations
  - Put hidden link to skip navigation at top of page
  - Make sure important words are first in menu items
- How to check in Firefox
  - Use developer tools to browse through DOM in order

Red-green colorblind

- ~9% of population, mostly male, cannot distinguish certain colors, mainly red vs green
- Accessibility considerations
  - Do not encode important information as red-green contrasts in color
  - e.g., links that have been visited or not







From: http://colorvisiontesting.com/ishihara.htm

JavaScript disabled

- People disable JavaScript for various reasons
  - Not supported on their (mobile) device
  - Low bandwidth
  - Avoid ads, trackers
- Accessibility considerations
  - Make sure the main functionality of your site is still available without javascript
  - Gmail's solution: provide a simple html alternative
- How to check in Firefox
  - Go to about:config page, set javascript.enable=false

Keyboard only

- Some people cannot or choose not to use a mouse or pointing device
  - Users with motor impairments
  - Although it's usually faster to use the keyboard (no need to move hands away and back)
- Accessibility considerations
  - Put hidden link to skip navigation at top of page
  - Consider and indicate the "focus" of the cursor
  - Maybe provide keyboard shortcuts for power users
- How to check in Firefox
  - Don't use your mouse (try tab key to move focus)

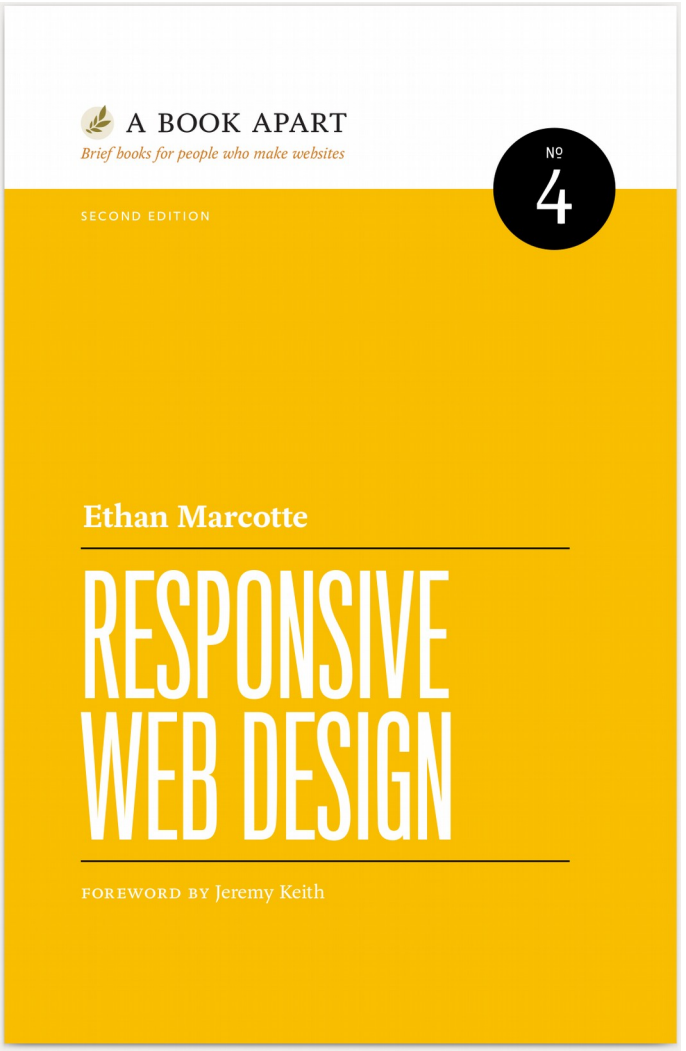
CISC 1600 Lecture 1.4
Design, part 2

Topics:

Responsive web design

Example design review

Responsive web design



From: Marcotte, Ethan. Responsive Web Design, 2nd ed. 2014. A Book Apart.

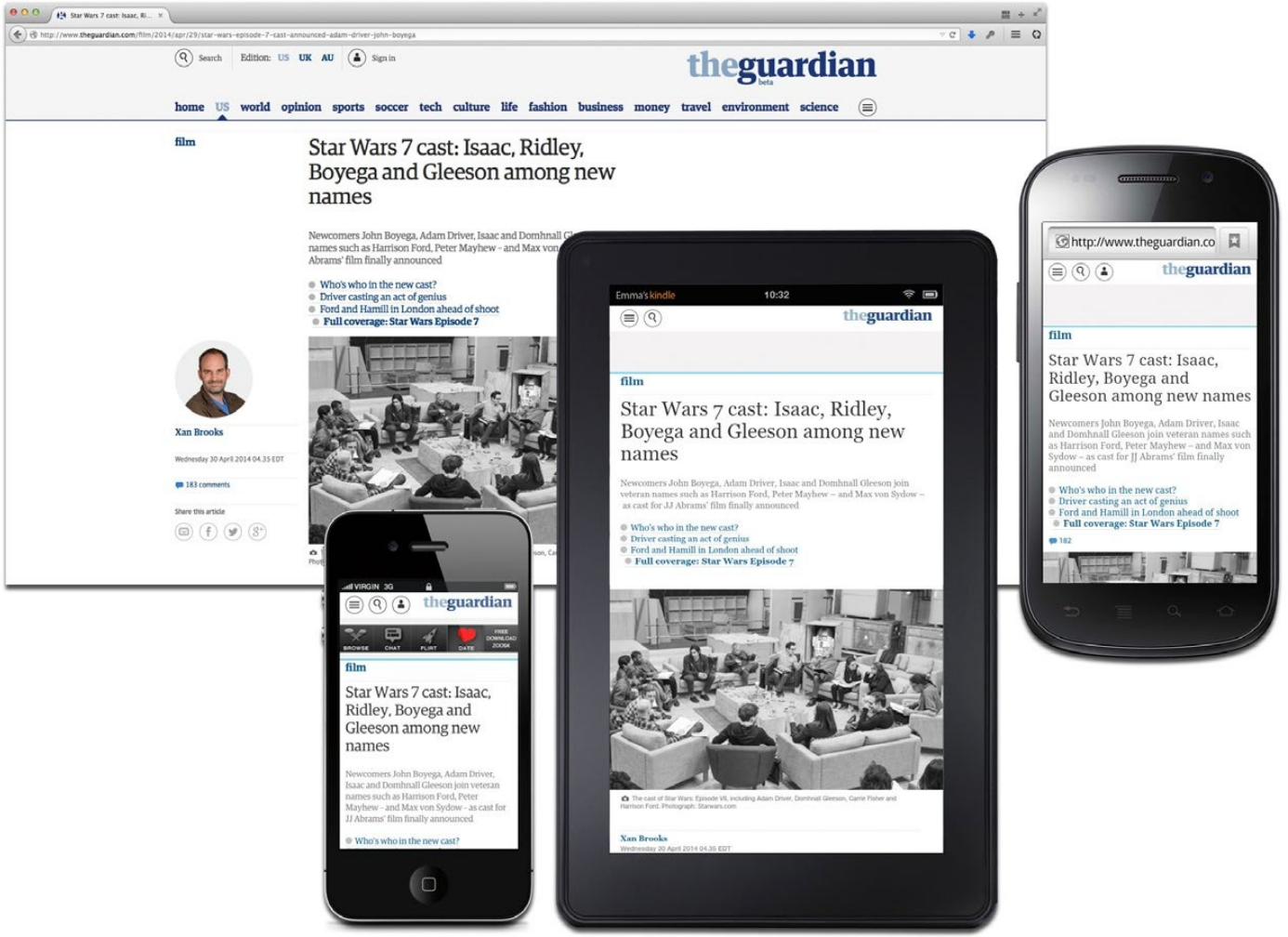
Responsive web design goals

- Use the same HTML and CSS for many different devices
  - Don’t repeat yourself
  - Don’t build a separate “mobile” web page
  - Ease of maintenance, modularity
- Build it in such a way that it is flexible to changes in screen size, input methods
  - TV, desktop, laptop, tablet, smart phone, feature phone

Respond to small changes in size

From: Marcotte, Ethan. Responsive Web Design, 2nd ed. 2014. A Book Apart.

Respond to large changes in size



From: Marcotte, Ethan. Responsive Web Design, 2nd ed. 2014. A Book Apart.

Responsive web design combines three elements

- Flexible grid-based layout
  - What: Adjusts to small changes in display size
  - How: Avoid specifying sizes in pixels
- Media queries in CSS
  - What: Adjusts to large changes in display size
  - How: CSS rules that only apply to certain displays
- Flexible images
  - What: Allow images to resize or crop as needed
  - How: CSS rules applied to image or containers

Noticeable as three effects when page changes size

- Flexible grid-based layout
  - Text containers change shape/size
  - Text reflows within them
- Media queries in CSS
  - Elements change relative/absolute position
  - (many other possibilities as well)
- Flexible images
  - Images change size, change bounding box

Implementing a responsive design: Flexible grid-based layout

- What: Text containers change shape/size
  - Text reflows within them
- How: Use percentages, not absolute sizes
  - Compute from size of element relative to container
  - Better yet, use a library like twitter bootstrap

Implementing a responsive design: Media queries in CSS

- What: Elements change position/size based on browser window properties
- How
  - Enable by putting this in you HTML <head>:
    ```
    <meta name="viewport" content=" initial-scale=1.0, width=device-width" />
    ```
  - Use media selectors in your CSS for conditional formatting:
    ```
    @media screen and (max-width: 768px) {
      /* this will affect pages up to 768px wide */
    }
    ```
  - Example of changing background color based on width

Implementing a responsive design: Media queries in CSS

FEATURE NAME	DEFINITION	HAS min- AND max- PREFIXES
width	The width of the display area.	✔
height	The height of the display area.	✔
device-width	The width of the device's rendering surface.	✔
device-height	The height of the device's rendering surface.	✔
orientation	Accepts portrait or landscape values.	✘
aspect-ratio	Ratio of the display area's width over its height. For example: on a desktop, you'd be able to query if the browser window is at a 16:9 aspect ratio.	✔
device-aspect-ratio	Ratio of the device's rendering surface width over its height. For example: on a desktop, you'd be able to query if the screen is at a 16:9 aspect ratio.	✔
color	The number of bits per color component of the device. For example, an 8-bit color device would successfully pass a query of (color: 8). Non-color devices should return a value of 0.	✔
color-index	The number of entries in the color lookup table of the output device. For example, @media screen and (min-color-index: 256).	✔

FEATURE NAME	DEFINITION	HAS min- AND max- PREFIXES
monochrome	Similar to color, the monochrome feature lets us test the number of bits per pixel in a monochrome device.	✔
resolution	Tests the density of the pixels in the device, such as screen and (resolution: 72dpi) or screen and (max-resolution: 300dpi).	✔
scan	For tv-based browsing, measures whether the scanning process is either progressive or scan.	✘
scan	Tests whether the device is a grid-based display, like feature phones with one fixed-width font. Can be expressed simply as (grid).	✘

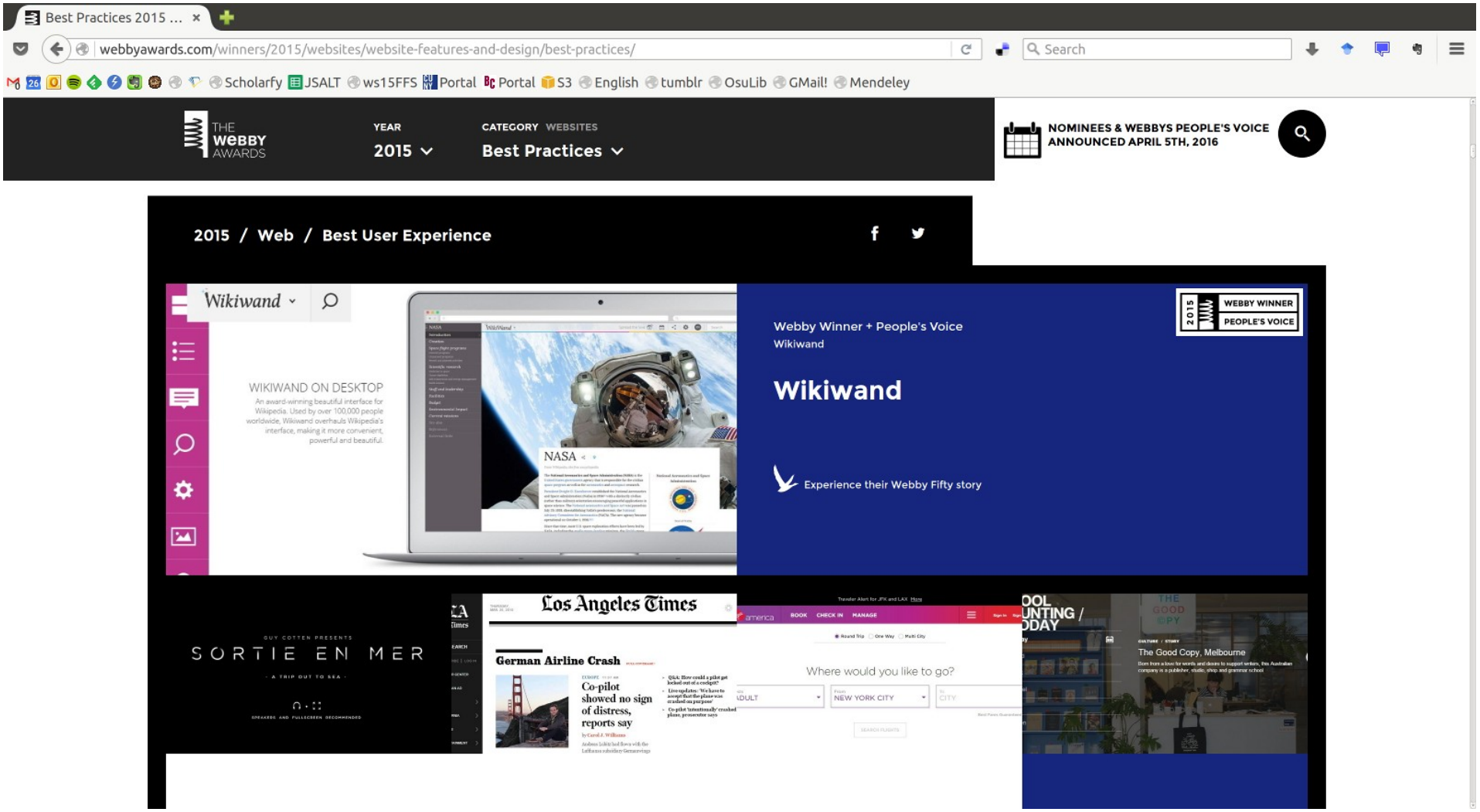
From: Marcotte, Ethan. Responsive Web Design, 2nd ed. 2014.

Implementing a responsive design: Flexible images

- What: Images change size, cropping
- How: CSS for image or its containing element
  - Stretch & squeeze to fit: “#myimg { width: 100%; }”
  - Squeeze to fit: “#myimg { max-width: 100%; } ”
  - Crop: “#mycont { overflow: hidden; } ”
  - Examples of different kinds of flexible images

Design review

Webby awards highlight good sites



Review: Responsiveness

- What changes shape?
- What changes size?
- What changes position?

Review: Accessibility

- Can the formatting be customized?
- How would a screen reader read this?
- Is any important information conveyed by red-green contrast?
- How does the site behave without javascript?
- Can you use it with only the keyboard?

Review: Visual design

- Is there a clear visual hierarchy?
  - What is most important? Second most?
- Is the page consistent with itself? The site?
- What do the aesthetics say about the site?
- What do the aesthetics say about the target audience?
- What feelings do the colors convey?

Review: Usability

- Does it make you think about what to do?
- What parts use conventional visual design?
- What parts have a conventional structure?
- What actions are possible?
- Where does this page fit in the overall site?

CISC 1600 Lecture 1.5
The internet and World Wide Web

Topics:

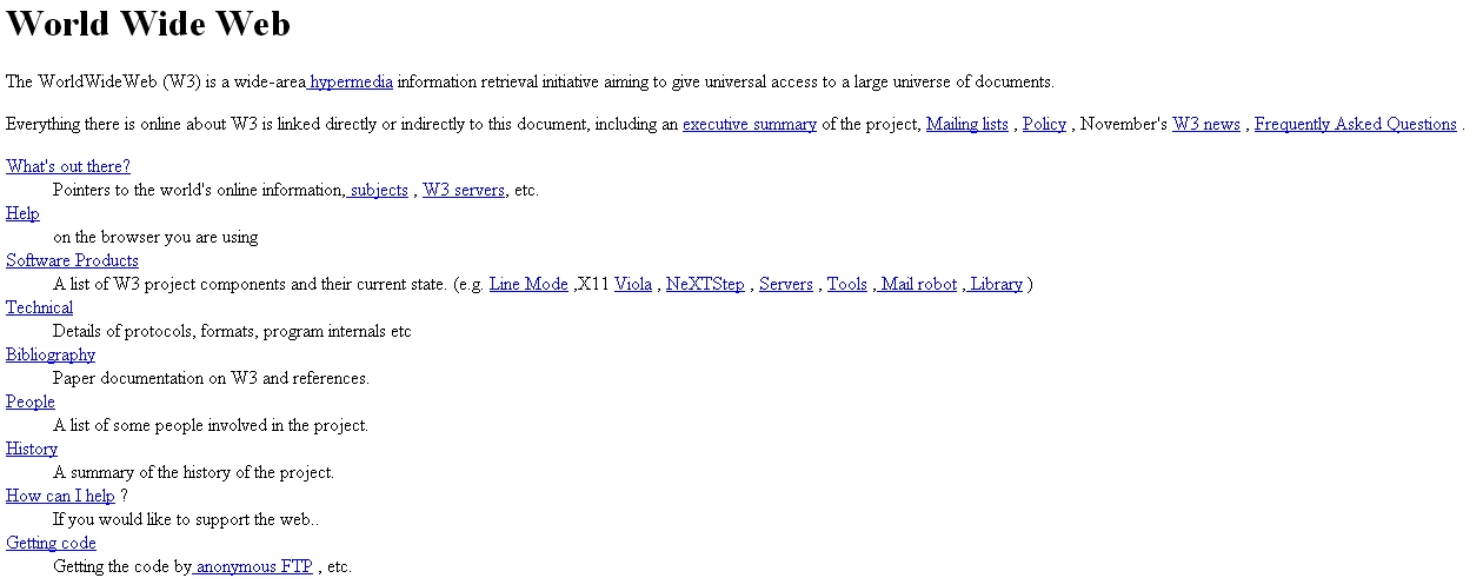
Internet overview
Web request overview
URLs and DNS
Network layers
Web request detail

The Internet is a global network of computer networks

- The US Department of Defense wanted a secure, reliable, efficient network robust to nuclear attack
- Built ARPANET (1969-1985)
  - Connected military and university networks
  - Precursor to the internet
- Based on packet switching (like mailing a letter)
- As opposed to circuit switching (like a phone call)

The World Wide Web is one service transported on the Internet

- Invented by Tim Berners-Lee at CERN in 1989
- Documents identified by URLs
- Linked by hyperlinks
- Accessed by the Internet



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

From: http://www.digitalgov.gov/2013/05/02/internet-time-travel/first-www-page-3/

Universal resource locator (URL)
Human-readable name for computer

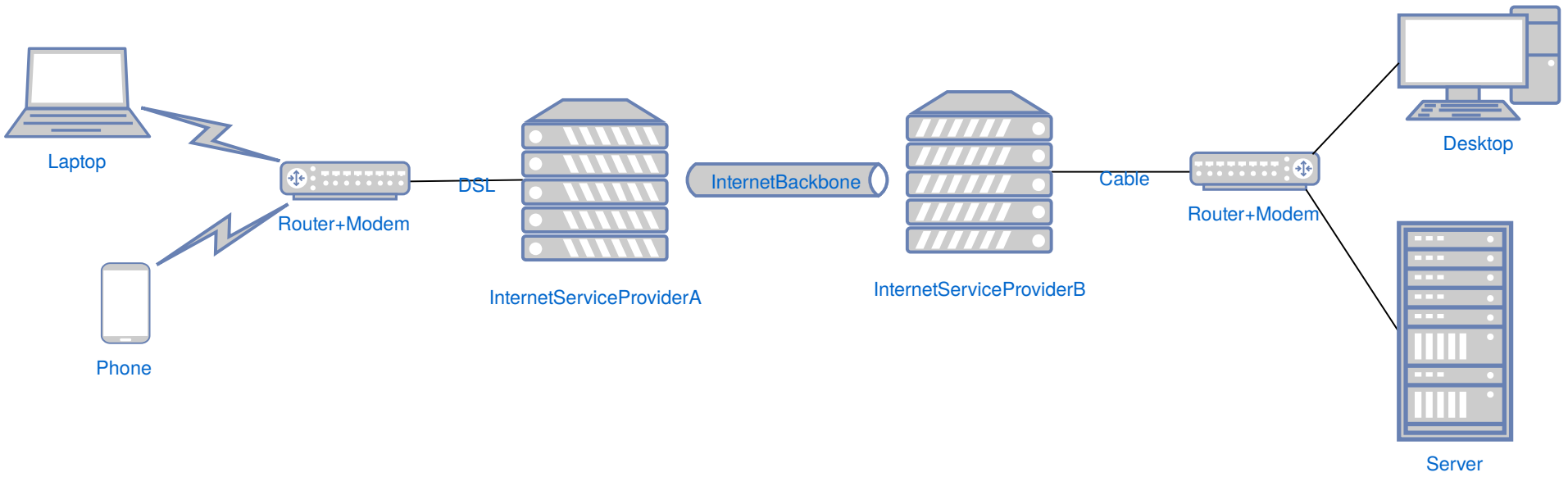
http://mr-pc.org/t/cisc1600/index.html

- http: Protocol, used by client to talk to server
- mr-pc.org: domain name, used by client to find server
- /t/cisc1600/: directory on server
- index.html: file in directory on server
  - index.html is the default file name, so optional here

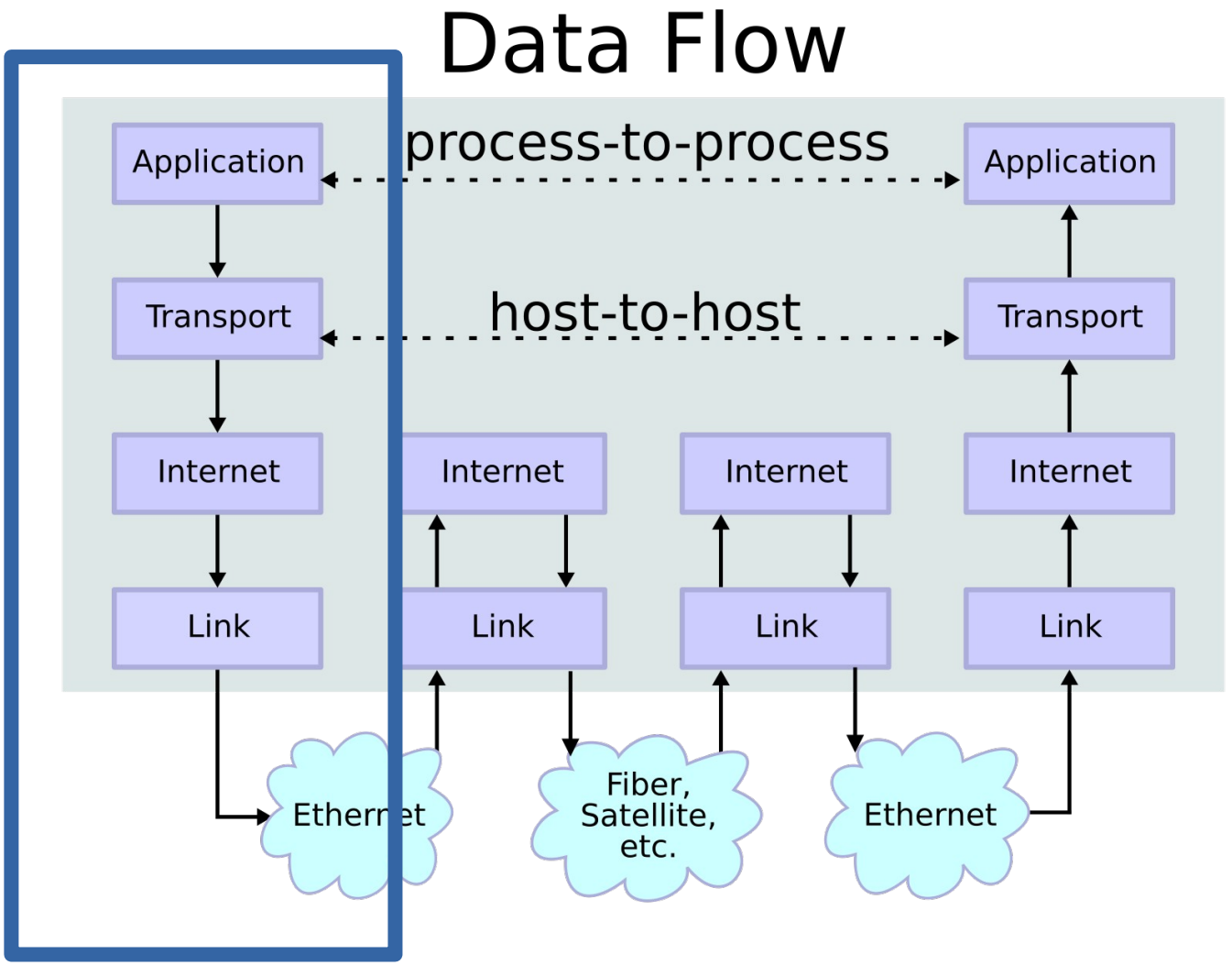
Domain Name System (DNS) finds computer-readable address

- Maps human-readable names (www.google.com) to numerical IP addresses (74.125.29.104)
- Actually a multi-stage lookup
  - DNS server for . knows IP address of .com.
  - DNS server for .com. knows IP address of google.com.
  - Which knows IP of www.google.com
- That's a lot of communication!
- Caching at many levels speeds up lookups
  - Browser, operating system, router, ISP, DNS servers

Computers are connected through several networking components



Establish temporary connection

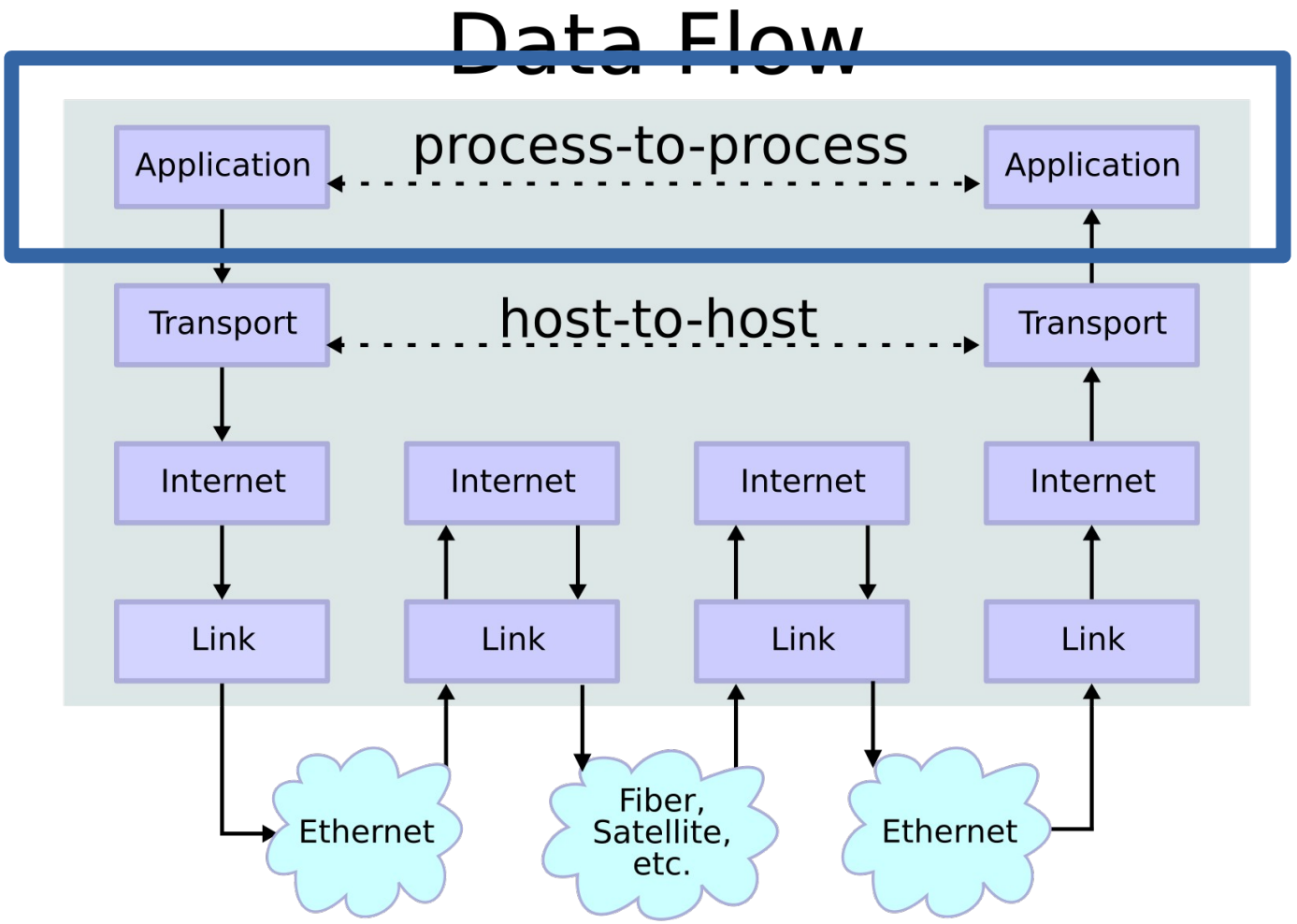


From: https://commons.wikimedia.org/wiki/File:IP_stack_connections.svg

This is a great example of a separation of concerns

- Separation of concerns between the layers
- Each layer has a job to do
- Each layer talks to the layers above and below it
  - Expose communication **interfaces** to each other
  - And can completely ignore the other layers

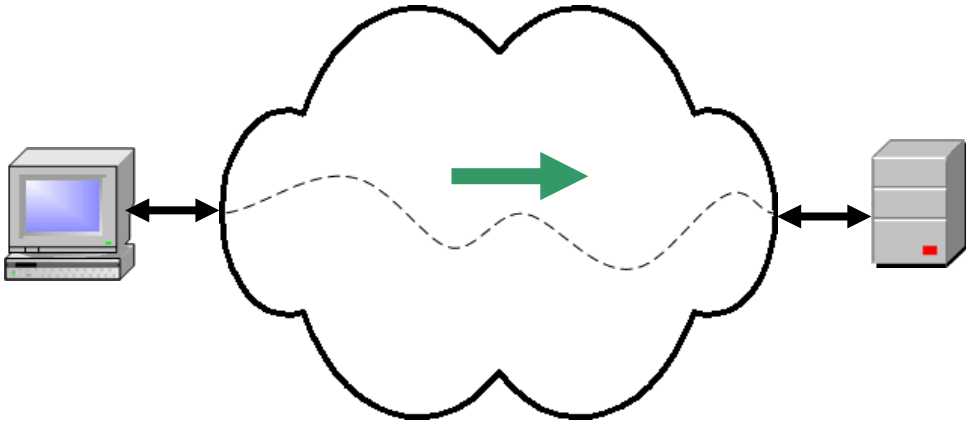
The applications are connected. Now they need to talk to each other



Browser sends an HTTP request

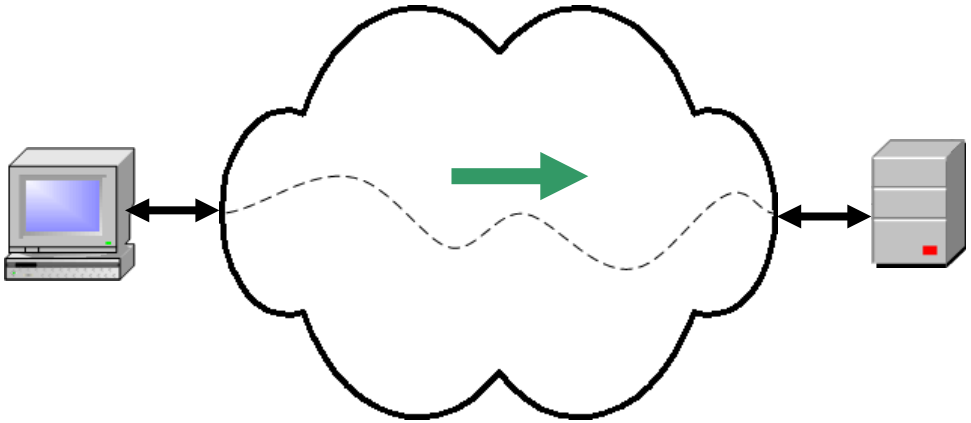
- HTTP is just (structured) text going back & forth

```
GET http://mr-pc.org/teaching/cisc1600/index.html
Host: mr-pc.org
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:41.0) Gecko/20100101 Firefox/41.0
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: __utma=121487444.1831872924.1418399252.1443726520.1444160561.55;
__utmz=121487444.1434478428.33.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); __utmb=121487444;
__utmc=121487444
Connection: keep-alive
If-Modified-Since: Wed, 30 Sep 2015 21:58:08 GMT
...
```



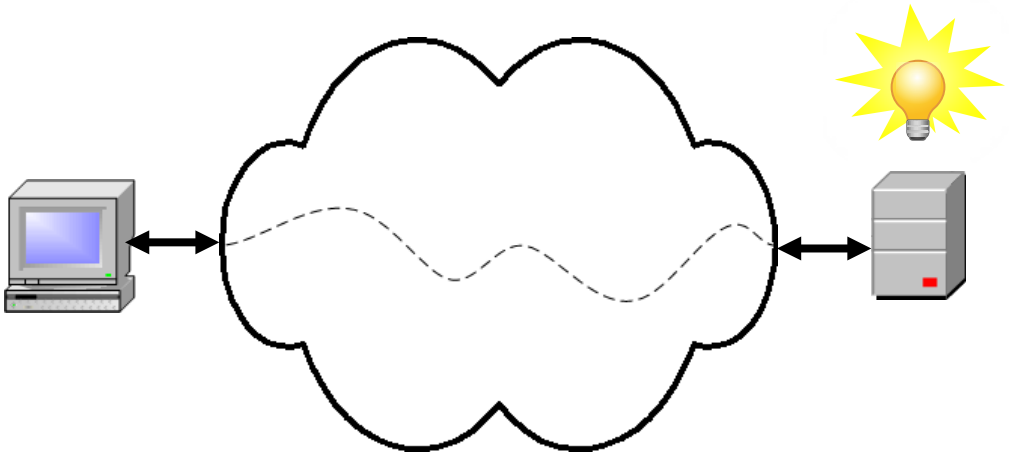
Browser sends an HTTP request

- `GET` command requests a particular page
- Browser identifies itself using `User-Agent` header
- Tells server what types of responses it will `Accept`
  - Language, character encoding, compression, etc
- Sends `Cookies` that the server has asked it to save
- Requests the server keep the `Connection` open



Server handles request

- Web server handles communication with client
- May talk to other servers to complete request
  - Read from and/or write to Database or storage server
  - Read from and write to Application server
  - Caching servers and load balancers in between
- Finally assembles response

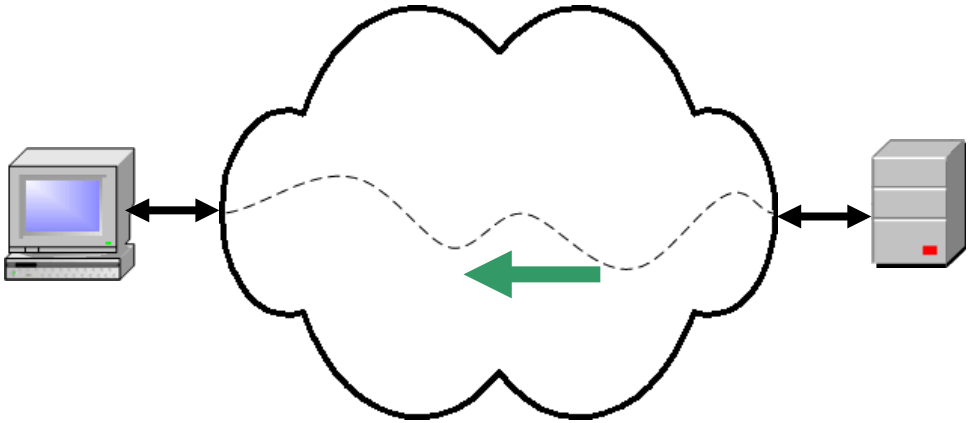


Server sends HTML response

- Sends headers as plain text, may compress body

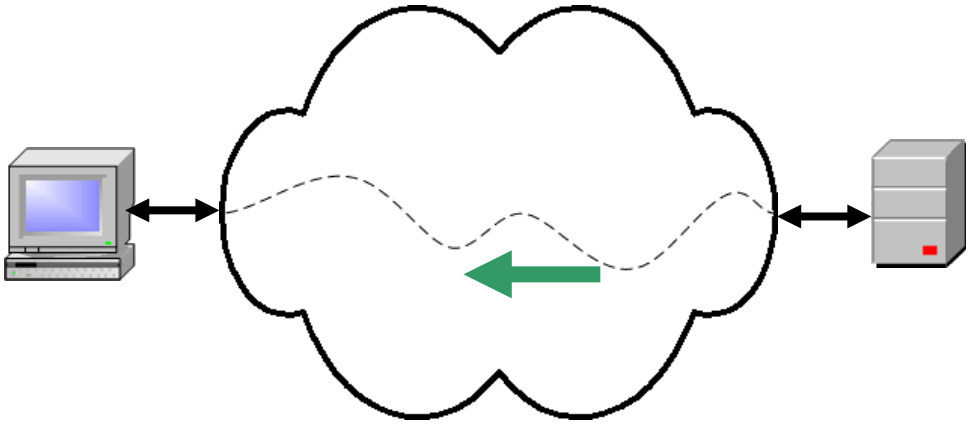
```
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 1575
Content-Type: text/html
Date: Tue, 06 Oct 2015 20:01:21 GMT
Etag: "168c-520fe069ff501-gzip"
Keep-Alive: timeout=5, max=100
Last-Modified: Wed, 30 Sep 2015 21:58:08 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding

...
```

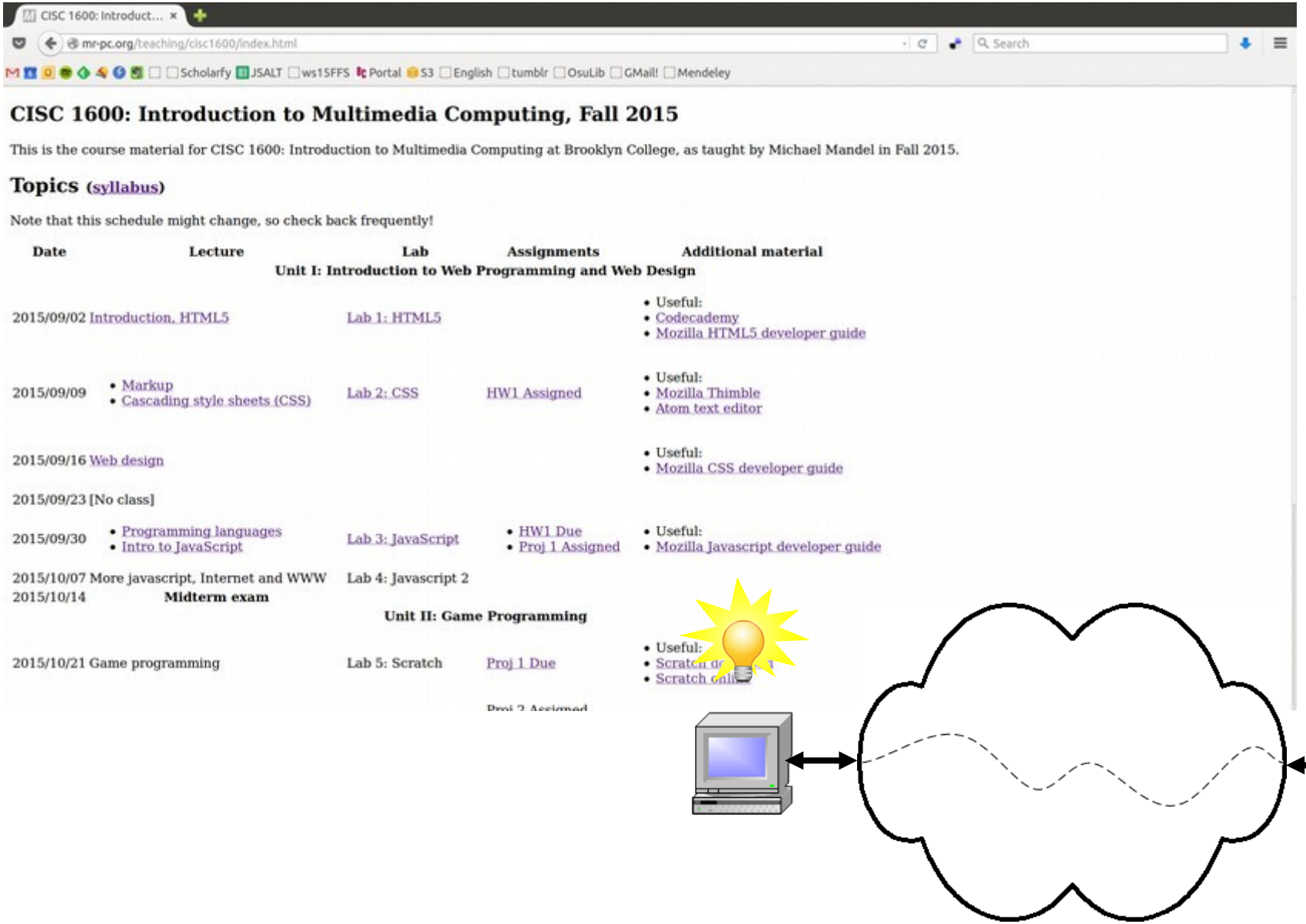


Server sends HTML response

- Information about `Server`
- `Content-Encoding` – content is compressed
- `Content-Type` – this is html
- How long the content should be cached for

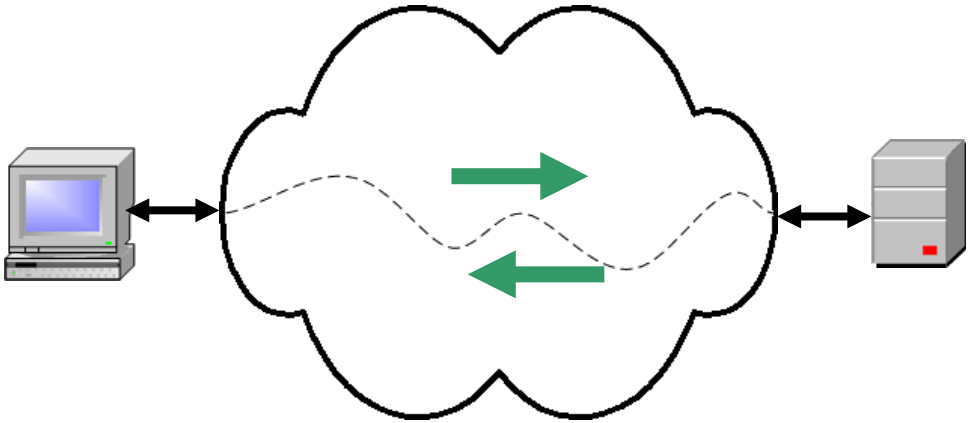


Browser starts to render the page



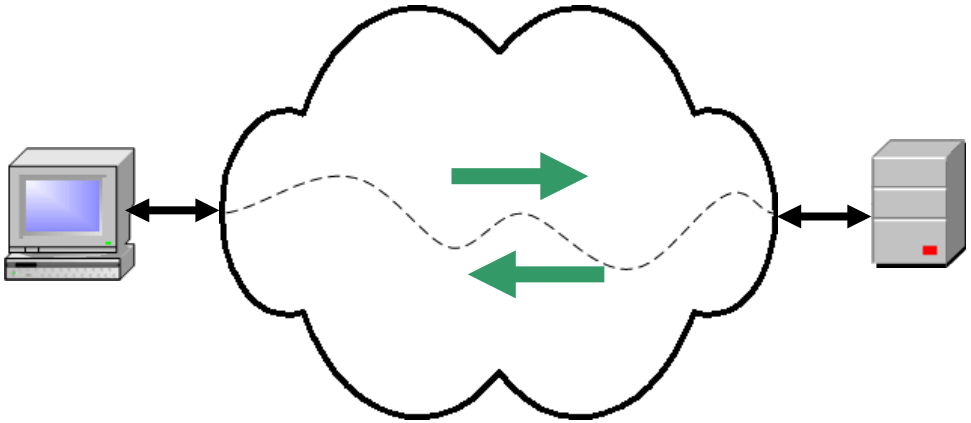
Browser requests objects embedded in HTML

- Can use cached versions if unchanged
- Otherwise, need separate request for each
  - Image
  - CSS file
  - Javascript file



Browser sends further asynchronous (AJAX) requests

- Page is fully displayed on client
- Loads any additional content
  - Secondary/optional pieces that can be deferred
  - Pieces of page loaded on-demand
  - Requires JavaScript to work (accessibility)



CISC 1600 Lecture 2.1
Introduction to Processing

Topics:

Example sketches
Drawing functions in Processing
Colors in Processing
General Processing syntax

Processing is for sketching

- Designed to allow artists to “sketch” with code
- Does a lot behind the scenes to make this possible
- Simplified version of Java
- Allows imperative, event-driven, procedural, and object-oriented programming paradigms
- Large collection of powerful libraries
  - 3D, sound, data visualization, text analysis, etc.

Your first Processing sketch

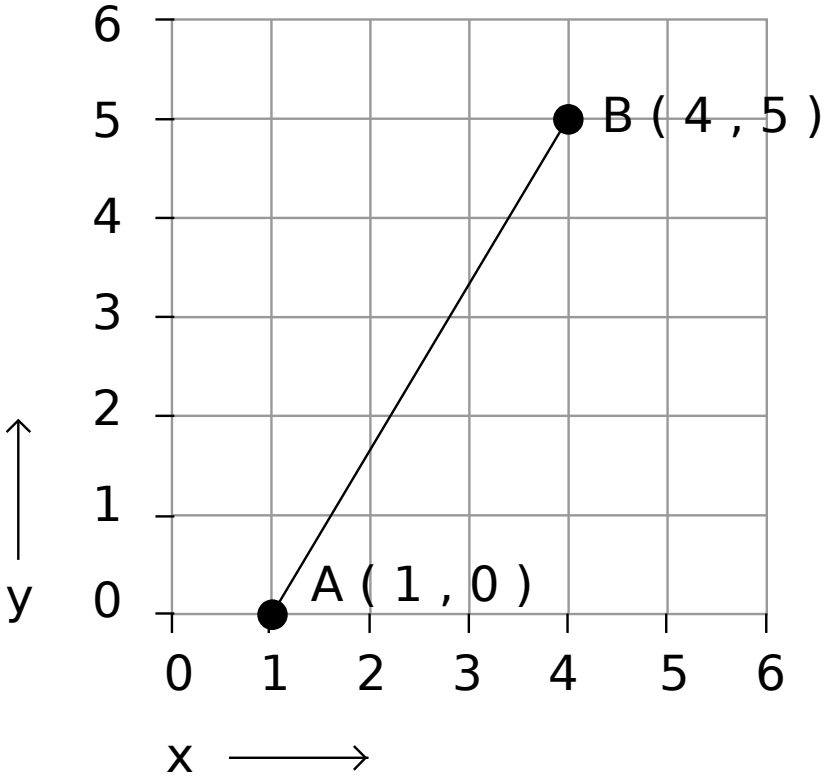
- Go to http://cisc1600.sketchpad.cc
- Create a new sketch
- Replace the existing code with

```
void setup() {
    text("Hello world", 10, 10);
}
```

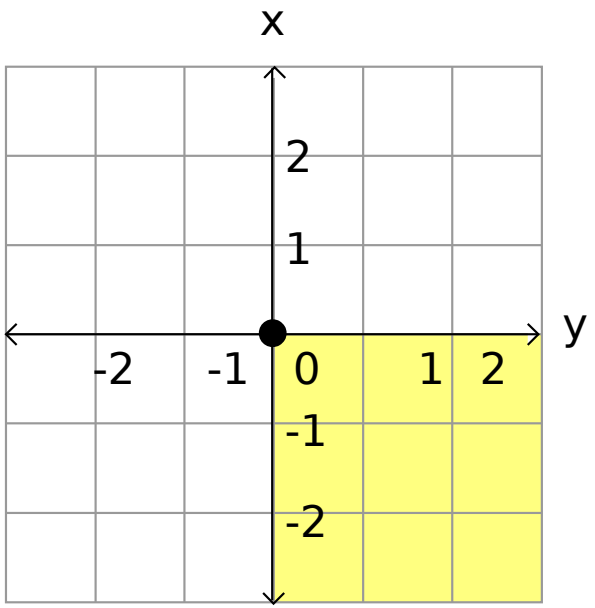
- Writes the text “Hello world” in white on the “canvas”

Drawing in Processing is like drawing on graph paper

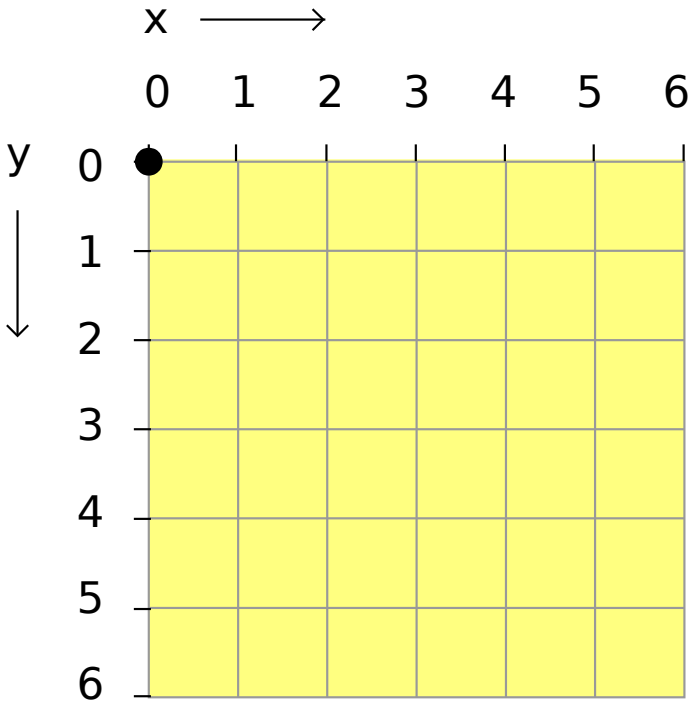
- Remember graphs from 8th grade?
- Drawing in processing is like telling a friend how to make a drawing on graph paper
- How would you describe this line to your friend?



One difference: the way the coordinates are setup

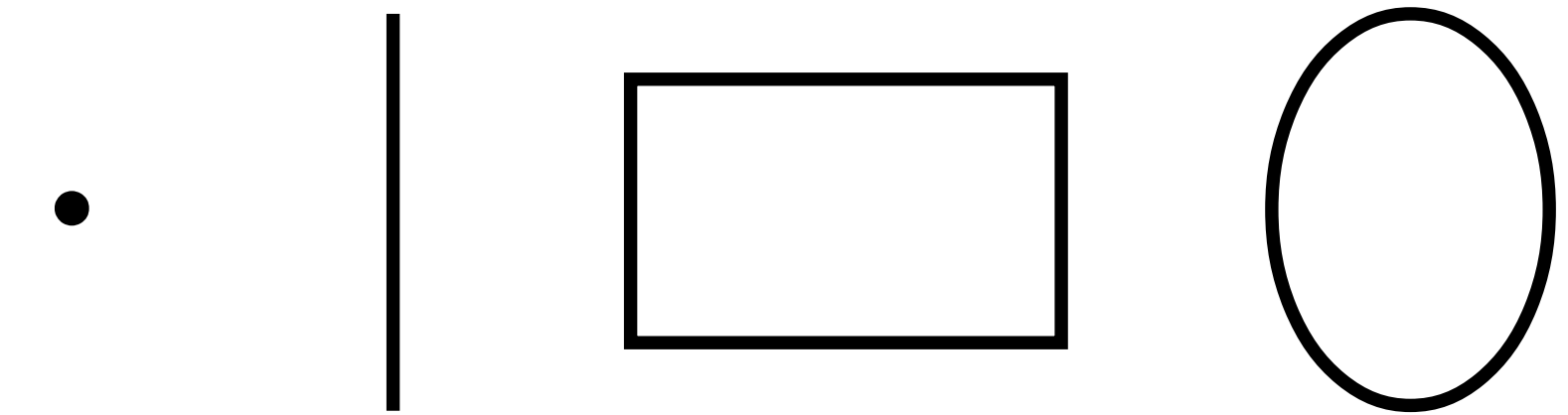


Eight Grade



Computer

Processing gives you functions to draw various shapes



Point Line Rectangle Ellipse

Functions allow you to use someone else's code

- Processing has a large library of pre-defined **functions**
- Functions take input **arguments** & return outputs

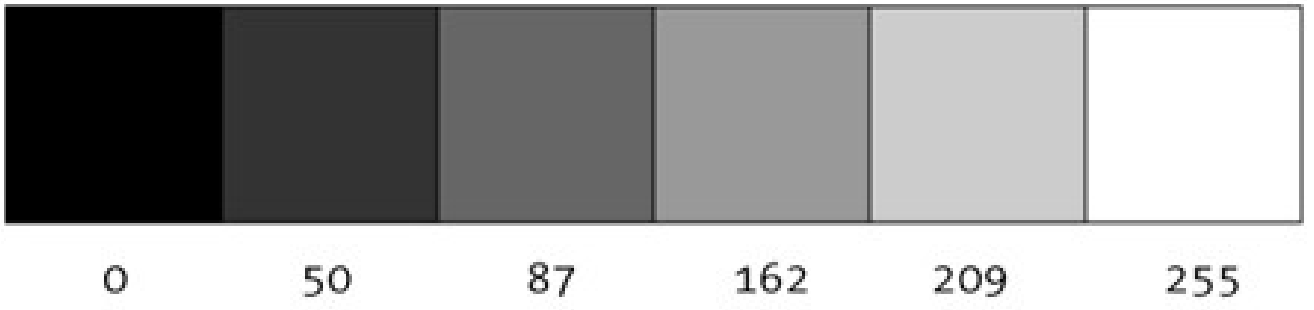
```
// Add two integers together
int add ( int x , int y ) {
    return x + y;
}
```

- The most important part of the function is the first line (and the comment)
- This function is called "add" (the word before the parentheses)
- It takes two input arguments, x and y (the things in the parens)
  - x and y are both of **type** int: an integer (0, 1, 2, -1, -2, ...)
- It returns an integer (the "int" before the name of the function)
  - Return type "void" means it doesn't return anything

Processing gives you functions to draw various shapes

- `point( x, y )`
  - draws one point (looks like a dot...)
- `line( x1, y1, x2, y2 )`
  - connects two points
- `rect( x, y, width, height )`
  - origin + extent; square if width=height
- `ellipse( x, y, width, height )`
  - origin: center of ellipse's; circle if width=height
- `triangle( x1, y1, x2, y2, x3, y3 )`
  - connects three points
- `quad( x1, y1, x2, y2, x3, y3, x4, y4 )`
  - connects four points

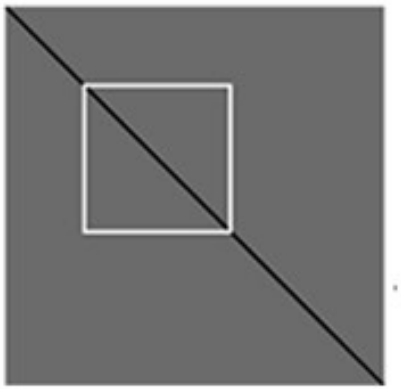
Processing color functions use grayscale or RGB



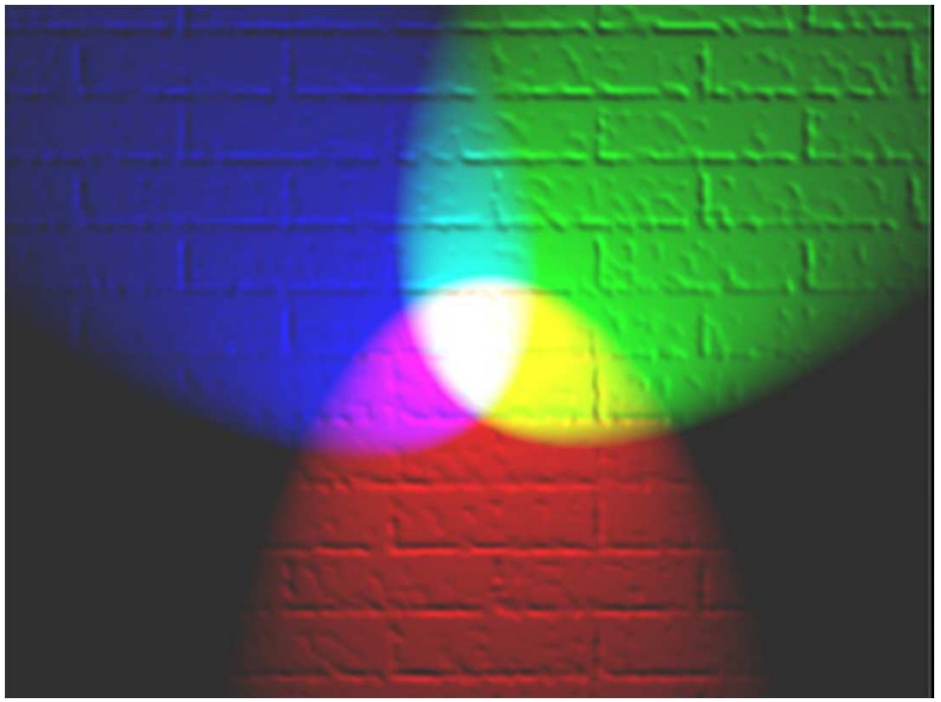
Processing color functions use grayscale or RGB

- Functions: `stroke(), fill()`
  - Change color of all subsequent shapes drawn
  - Like setting the current color in a drawing program
- Have two versions: `stroke(gray), stroke(r,g,b)`

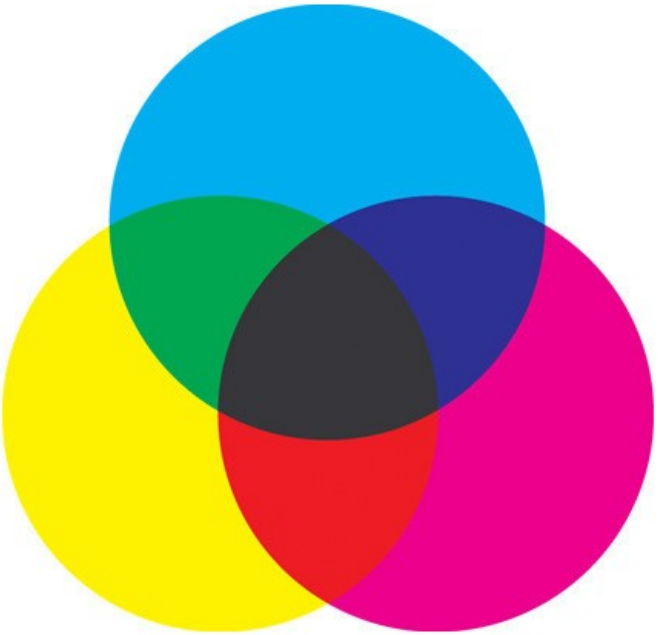
```
background(150);
stroke(0);
line(0,0,100,100);
stroke(255);
noFill();
rect(25,25,50,50);
```



Colors mix differently on screen than they do in art class



Red, green, and blue:
Primary colors for mixing light



Cyan, Magenta, and Yellow:
Primary colors for mixing paint

General Processing syntax

```
/* This is a multi-line comment.
   The computer will ignore it.
   It is just for people to read */

// This is a one-line comment

// Declare some variables to hold data for later
int x, y;

// Setup function is called once before everything
void setup() {
    // Do things that need to be done once
}

// Draw function is called once for every frame
void draw() {
    // Draw to canvas
}
```

Variables hold values

- Variables are named locations for storing data
- Assign a value to a variable to refer to it later
  - Values can be simple: number, string, boolean
  - Or complex: various predefined and user-defined data structures
- Create a variable

```
int count;
```

- Assign a value to it:

```
count = 0;
```

- Use it:

```
text("count = " + count, 10, 10);
```

The `if` statement lets our program make a choice

- Processing implements the imperative paradigm
- The syntax of its `if` statement is identical to JavaScript's

```
if (mouseX < 150 && mouseY >= 150) {
    // mouseX is less than 150
    // AND mouseY is greater than or equal to 150
} else if (mouseX < 150) {
    // What can we say about mouseY here?
} else {
    // What can we say about mouseX here?
}
```

The `for` loop lets our program repeat operations on new data

- `for` loops are good when the number of iterations is known in advance

```
for (int i=0; i<5; i++) {
    rect(0, 60*i, 40, 40);
}
```

- The three statements in the parentheses are
  - Initialization: run once before loop
  - Continuation: loop continues until this is not true
  - Increment: update counter, run before each iteration
- You can think of `draw()` as being called in a loop

The `while` loop lets our program repeat operations on new data

- `while` loops are good when you don't know the number of iterations in advance
- Repeat a set of statements while a condition holds

```
while (y < height - 60) {
    rect(0, y, 40, 40);
    y = y + 60;
}
```