Lecture 11: Interfaces on the Web

Web interface origins

- Origins of the web interfaces lie in *hypermedia* and *hypertext*
- Early beginnings...
  - Vannevar Bush (Roosevelt science advisor, 1945)
    - memex tool: microfilm with encyclopedias of information and associative trails
    - just stare at short text and it would be "amplified"
  - Ted Nelson (1960s)
    - coined term "hypertext"
    - along with "docuverse" and "stretch text"
    - "computopian hopes" !!!

Web interface origins

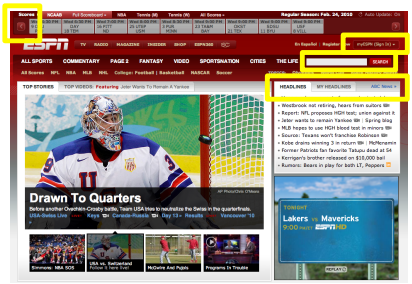
- Development & implementation...
  - Douglas Englebart (1960s) - remember him?
    - Human Augmentation system: point-and-click, expanding outlines, etc.
  - Andries van Dam
    - earliest electronic books
    - exploited new technologies, especially graphics and animation (2d & 3d)
- By mid-1980s, hypertext was mainstream
  - primarily as a publication tool — presenting information with "convenient jumps"
  - Apple HyperCard (Bill Atkinson, 1987)

Hypertext

- Writing & reading hypertext is different than writing/reading normal text
- Three Golden Rules (Shneiderman):
  - 1. There is a large body of information organized into numerous fragments.
  - 2. The fragments relate to one another.
  - 3. The user needs only a small fraction of the fragments at any one time.
- What's not (easily) amenable to hypertext? (according to Shneiderman…)
  - novels, poems
  - reference books?
  - news articles??

Hypertext to web pages

- Hypertext was a necessary condition for web pages, but not a sufficient one
- What else do we need?
  - layout
  - images/icons
  - styles
  - GUI elements
  - animation?
  - scripted objs?



Categorizing web sites

- Categorizing by site goals
  - Sell products
    - e.g., book sellers, eBay
  - Advertise products or services
    - e.g., real estate agents, auto dealers
  - Inform and announce
    - e.g., universities, governments
  - Provide access
    - e.g., libraries, newspapers
  - Create discussions
    - e.g., bboards, chat rooms
  - Nurture communities
    - e.g., professional orgs, political orgs

Categorizing web sites

- Categorizing by size/genre
  - 1 - 10 pages
    - personal site, project summary
  - 5 - 50 pages
    - conference program, organization overview
  - 50 - 500 pages
    - city guide, product catalog
  - 500 - 5,000 pages
    - technical reports, film database
  - 5,000 - 50,000 pages
    - university guide, newspaper site
  - 50,000 - 500,000 pages
    - directories/indices, airline schedules
  - 500,000 - 5,000,000 pages
    - congressional digest
  - > 5,000,000 pages
    - Library of Congress, NASA archives

Is web design different from GUI design?

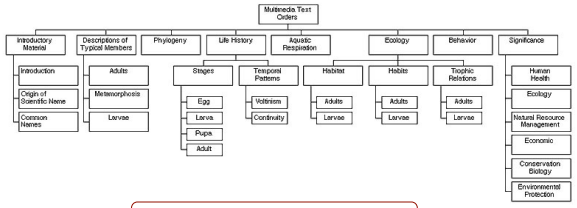
- Hmm. Experts don't agree on this one.

Is web design different from GUI design?

- "It's very different"…
  - you only design part of the interface; you don't control one big part: the *browser*
  - user can manipulate many aspects of interaction
    - e.g., resizing windows, changing fonts, navigating back and forth, bookmarking, etc.
  - some things are out of both your & users' control
    - e.g., download times, security
  - your pages are a tiny part of the web space
    - thus, seen by a small fraction of people, or (likely) a self-selected group of people
  - scale of particular sites
    - can large sites really be designed?

Web site design

- Site prototyping
  - storyboards are very useful
  - flowcharts / hierarchies provide nice overviews of entire sites (or parts thereof)
    - e.g., site for "Aquatic Entomology" course



Next bunch of slides derived from information at http://www.edtech.vt.edu/edtech/id/interface/

Web site design

- Navigation types
  - menu-tree navigation
    - menu hierarchy is visible at all times, user can go down then backtrack up

Web site design

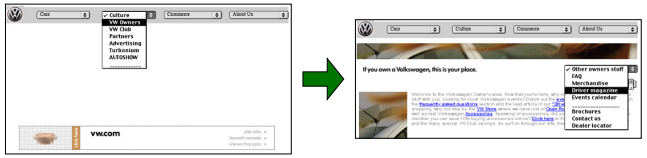
- Navigation types
  - tab-stop navigation
    - like menu-tree, but uses "tab" physical metaphor

Web site design

- Navigation types
  - index navigation
    - (almost) all information visible/accessible from top level

Web site design

- Navigation types
  - pull-down menu navigation
    - uses Javascript to select next page
    - display current selection??



Web site design

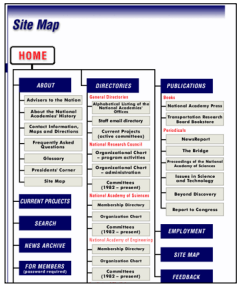
- Navigation types
  - iconic navigation
    - find information by picture
    - (perhaps) less ordered than textual menus (??)

Web site design

- Navigation types
  - page-turning navigation
    - natural for sequential information, not as useful for large and/or complex sites

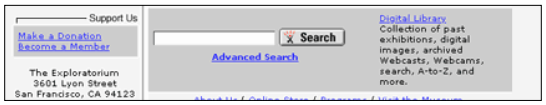
Web site design

- Navigation types
  - navigation by site map or table of contents
    - site map has grouping+linking, table of contents doesn't



Web site design

- Navigation types
  - navigation by search
    - essentially, builds an index on the fly based on given search terms

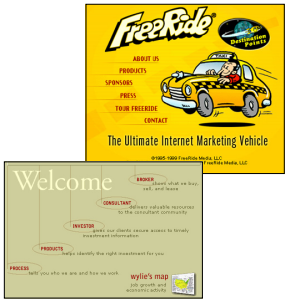


Web site design

- Navigation types
  - mixing navigation types...

Web site design

- Screen layout
  - balance is an essential component, as it is for any window



Web site design

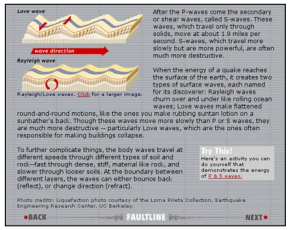
- Screen layout
  - focal point guides viewer's eye to desired places

Web site design

- Screen layout
  - high-density layouts are difficult to navigate, especially without strong visual grouping

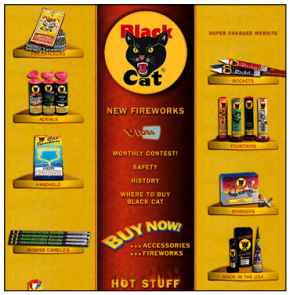
Web site design

- Screen layout
  - consistency (as always) is key from page to page



Web site design

- Screen layout
  - metaphors can evoke mental models of well-known objects and improve recognition



Web GUIs

- Two ways to think about web GUIs...
  - (1) The web page itself is a GUI.
    - links ≈ buttons
    - radio buttons & check boxes
    - input fields ≈ text fields
    - pull-down menus
    - layout
    - keyboard shortcuts (browser)
  - (2) The web page can contain "sub-GUIs."
    - i.e., applets!
    - Swing & Java give us the tools to make applets

Exercise

- Can we design a Java IDE for the web?
  - provide editing, compiling, running, debugging
  - what would it look like?
  - what functions would you expect from it?
  - how does this compare to "on-your-machine"?
    - interaction feel?
    - other issues?
- Note: Not too far-fetched! Could be good for a new programming language.

Applets

- Ok, let's return to Web GUIs as applets.
- Applet methods

```
public class Simple extends Applet {
    . . .
    public void init() { . . . }
    public void start() { . . . }
    public void stop() { . . . }
    public void destroy() { . . . }
    . . .
}
```

  - init() : fast one-time initialization
    - should contain code normally in the constructor
  - start() : performs some work or starts threads
  - stop() : pauses work when applet not visible
  - destroy() : final cleanup

Applets

- Event handling, Part I
  - drawing by painting + handling events manually

```
class Simple extends Applet {
    public void paint(Graphics g) { . . . }
    . . .
}
...

public boolean mouseDown (Event event, int x, int y) {
    addItem("click!... ");
    return true;
}
```

- Event handling, Part II
  - treat applet like a normal window
    - JApplet is a top-level container, like JFrame & JDialog

Applets

- Example: HelloSwingApplet


```
public class HelloSwingApplet extends JApplet {

    public HelloSwingApplet() {
        << "hack" to avoid error in 1.1
    }

    public void init() {
        JLabel label = new JLabel(
                "You are successfully running a Swing applet!");
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setBorder(BorderFactory.createMatteBorder
                        (1,1,2,2,Color.black));
        getContentPane().add(label, BorderLayout.CENTER);
    }
}
```


Applets

- Example: AppletDemo


```
public class AppletDemo extends JApplet
                        implements ActionListener {
    protected JButton b1, b2, b3;

    protected static final String DISABLE = "disable";
    protected static final String ENABLE = "enable";

    protected String leftButtonFilename = "images/right.gif";
    protected String middleButtonFilename = "images/middle.gif";
    protected String rightButtonFilename = "images/left.gif";

    private boolean inAnApplet = true;
    URL codeBase; //used for applet version only

    //Hack to avoid ugly message about system event access check.
    public AppletDemo() {
        this(true);
    }
```


Applets

- Example: AppletDemo

```
    public AppletDemo(boolean inAnApplet) {
        ...
    }

    public void init() {
        setContentPane(makeContentPane());
    }

    public Container makeContentPane() {
        ...
        b1 = new JButton("Disable middle button", leftButtonIcon);
        b1.setVerticalTextPosition(AbstractButton.CENTER);
        b1.setHorizontalTextPosition(AbstractButton.LEFT);
        b1.setMnemonic(KeyEvent.VK_D);
        b1.setActionCommand(DISABLE);
        << set up other buttons >>

        JPanel pane = new JPanel();
        pane.add(b1);
        << add other buttons >>
        pane.setBorder(BorderFactory.createMatteBorder(1,1,2,2,Color.black));

        return pane;
    }
```


Applets

- Example: AppletDemo

```
    public void actionPerformed(ActionEvent e) {
        << handle enabling/disabling >>
    }

    ...

    // Following code only for running code as an application!

    public static void main(String[] args) {
        JFrame frame = new JFrame("Application version: AppletDemo");

        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

        AppletDemo applet = new AppletDemo(false);
        frame.setContentPane(applet.makeContentPane());
        frame.pack();
        frame.setVisible(true);
    }
}
```

Applets

- Including applets on a web page
  - a simple way...

```
<APPLET CODE=AppletSubclass.class WIDTH=anInt HEIGHT=anInt>
</APPLET>
```

  - with parameters and alternate text...

```
<APPLET CODE="Animator.class" WIDTH=460 HEIGHT=160
 ALT="If you could run this applet, you'd see some animation">
<PARAM NAME="imageSource" VALUE="images/Beans">
<PARAM NAME="backgroundColor" VALUE="0xc0c0c0">
<PARAM NAME="endImage" VALUE=10>
<PARAM NAME="soundSource" VALUE="audio">
<PARAM NAME="soundtrack" VALUE="spacemusic.au">
<PARAM NAME="sounds"
       VALUE="1.au|2.au|3.au|4.au|5.au|6.au|7.au|8au|9.au|0.au">
<PARAM NAME="pause" VALUE=200>
Your browser is completely ignoring the &lt;APPLET&gt; tag!
</APPLET>
```


Applets

- Applet security
  - What applets typically *cannot* do (subject to particular browser)
    - read or write files on the host that's executing it
    - make network connections except to the host that it came from
    - start any program on the host that's executing it
    - read certain system properties
    - control certain aspects of window appearance

Applets

- Applet security
  - What applets typically *can* do (subject to particular browser)
    - make network connections to the host they came from.
    - cause HTML documents to be displayed.
    - invoke public methods of other applets on the same page
    - keep running after you leave their page (though this is not recommended/desired for most applets)
  - Note: Applets loaded from the local file system have fewer restrictions than network applets

Web 2.0

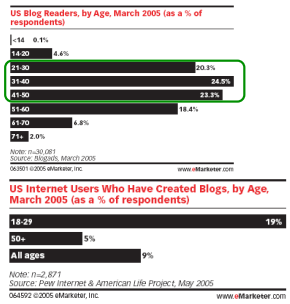
- What is "Web 2.0"?

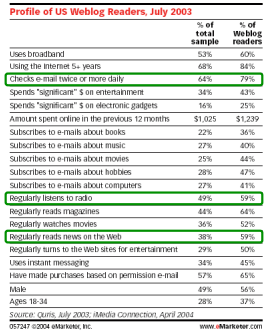
Web 1.0 (1993-2003)		Web 2.0 (2003- beyond)
Pretty much HTML pages viewed through a browser		Web pages, plus a lot of other "content" shared over the web, with more interactivity; more like an application than a "page"
"Read"	Mode	"Write" & Contribute
"Page"	Primary Unit of content	"Post / record"
"static"	State	"dynamic"
Web browser	Viewed through...	Browsers, RSS Readers, anything
"Client Server"	Architecture	"Web Services"
Web Coders	Content Created by...	Everyone
"geeks"	Domain of...	"mass amateurization"

Next few slides derived from material by Jim Cuene

Web 2.0

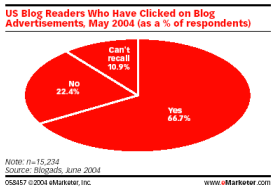
- Who uses Web 2.0 sites?
  - If you believe that bloggers are roughly like Web 2.0 users…

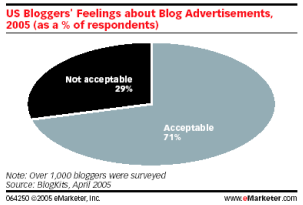




Web 2.0

- Surprisingly, users find advertising acceptable
  - Web 2.0 could open up marketing options?





Web 2.0

- What changes might we expect from marketers?
  - More users are connecting to each other and content through networked, peer-driven activities & content
    - Linkedin now has service referrals as part of their package
  - API's and Content syndication will lead to more machine generated connections
    - "Non-compliant" content won't fit into the flow as readily
  - Web 2.0 is truly two-way
    - Marketers need to be very willing to "listen" and receive more than broadcast
  - User-generated content may be more valuable to users than content generated by companies/organizations
  - Adoption will drive investments in online advertising

Engineering Models of User Behavior

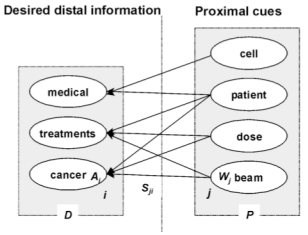
- We've talked recently about…
  - engineering models to predict user behavior in certain situations
  - user model frameworks (e.g., KLM-GOMS) that predict time-on-task
  - computational user models (e.g., production systems) as used for intelligent interfaces and model tracing
- The basic ideas also apply to the web domain… but we can do better by focusing on specifics of this domain
  - #1: SNIF-ACT model [Fu & Pirolli]
  - #2: Bloodhound system [Chi et al.]

SNIF-ACT

- Goal: Encapsulate web-browsing behavior in a computational model
  - ... to better understand behavior
  - ... to predict behavior (as we will see)
- Approach: Model based on...
  - Information Foraging
  - the ACT-R cognitive architecture - a production system framework

SNIF-ACT

- Key component: Information Scent
  - distal info = desired info a few clicks away
  - proximal cues = info right now (e.g., link names)



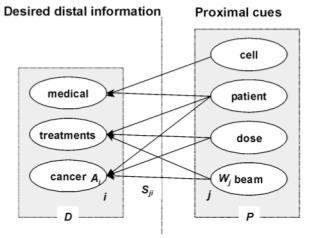
SNIF-ACT

- Key component: Information Scent

Try finding information about majoring in Sociology at Drexel... www.drexel.edu

SNIF-ACT

- Key component: Information Scent



Activation

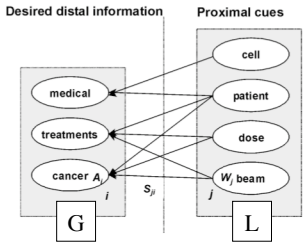
Strengths (log-prior odds)

$$A_i = B_i + \sum_{j \in L} W_j S_{ji}.$$

Base-level activation

SNIF-ACT

- Key component: Information Scent
  - for G = information goal (what user is seeking) and L = link to that information...



$$A_i = B_i + \sum_{j \in L} W_j S_{ji}.$$

Information Scent

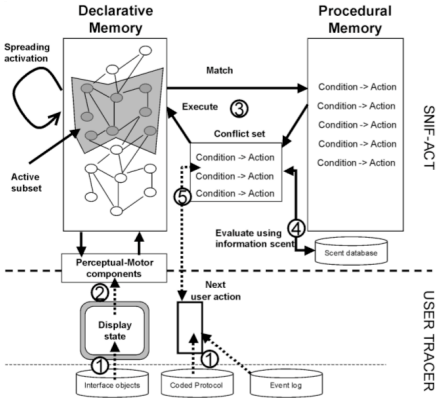
$$IS(G,L) = \sum_{i \in G} A_i = \sum_{i \in G} (B_i + \sum_{j \in L} W_j S_{ji})$$

SNIF-ACT

- Key component: Information Scent
  - Where do we get Sji?
    - can construct activation networks from online text corpora and calculate Sji for different words and information goals
    - and base-rate frequencies Bi of all words and pairwise co-occurrence frequencies of words can also be computed

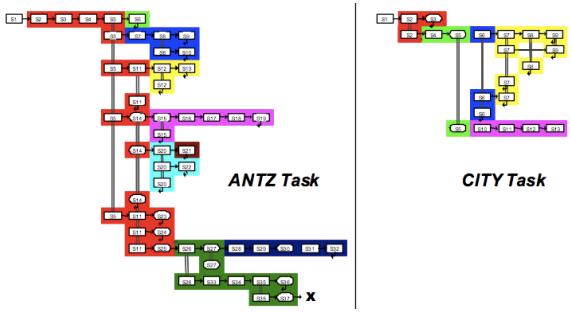
SNIF-ACT

- Declarative Memory
  - perception puts info in memory: link names, etc
- Procedural Memory
  - knowledge of using the browser
  - stored as condition-action production rules
  - e.g., attend-to-link, click-link



SNIF-ACT

- Web Behavior Graphs for two domains...
  - (different colors = different web sites)



SNIF-ACT

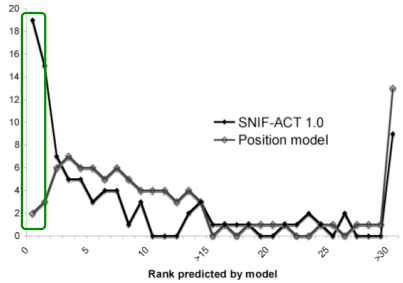
- Model predicts...
  - (1) which links user will click on
  - (2) when people decide to leave a site
- Testing
  - these two actions extracted from log files
  - actions compared to model predictions

SNIF-ACT

- Results
  - histogram showing predicted model rank for clicked links...

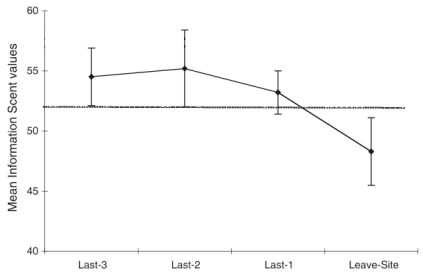
(bias to the left better)

(Position model simply ranks by position, top & left better)



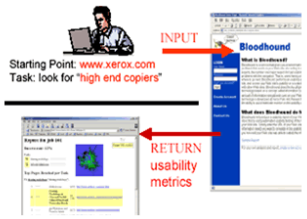
SNIF-ACT

- Results
  - scent values before and right when leaving site
    - values decrease below overall mean (dotted line)



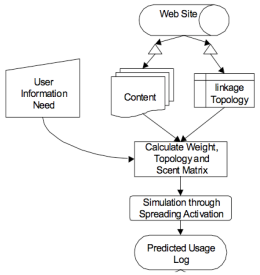
Bloodhound

- So SNIF-ACT, Information Foraging, etc. provide reasonable predictions of behavior
- How can we instantiate this into a real tool?
- Bloodhound is a system that analyzes the information cues on a web site
  - strives for a simple application of theory
  - enter a web site + search words, get a usability report



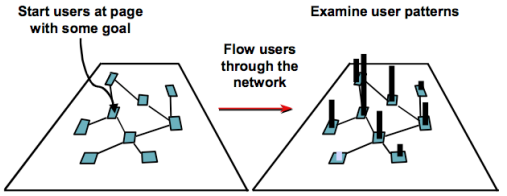
Bloodhound

- Overview of algorithm (Web User Flow by Information Scent) and spreading activation



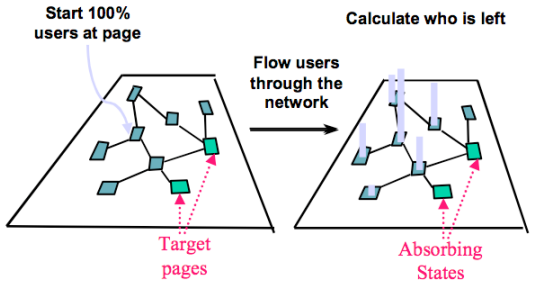
Bloodhound

- Overview of algorithm (Web User Flow by Information Scent) and spreading activation



Bloodhound

- Overview of algorithm (Web User Flow by Information Scent) and spreading activation



Bloodhound

- User testing
  - 244 users, 1386 user sessions
- Domains
  - help.yahoo.com (Yahoo! help system section)
  - www.rei.com (a camping/outdoor online store)
  - hivin-site.ucsf.edu (AIDS and HIV medical site)
  - parcweb.parc.com (company intranet)

Bloodhound

- **Tasks**

Help.yahoo.com (7484 documents):

• You want Yahoo! to add your site to the Yahoo! Directory. Find some guidelines for writing a description of your site.
• When is the playing season for Fantasy Football?
• You want to get driving directions to the airport, but you don't know the street address. How else can you get accurate directions there?

REI (36422 documents):
• You are planning a week-long hiking trip for this summer, and you're on a budget. Find a single person tent for less than $120.
• Find the location of the REI store nearest you.
• Find yourself some warm, fairly heavy long underwear for the upcoming ski season.

etc.

Bloodhound

- **Results** (yellow good, green not so much)

Corr. Coeff.	Yahoo	REI	HivIn-Site	Parc-web
task 1a	0.7528	0.4701	0.6811	0.7394
task 1b	0.7218	0.4763	0.7885	0.8756
task 2a	0.7489	0.9892	0.6671	0.8930
task 2b	0.8840	0.7073	0.6880	0.8573
task 3a	0.7768	0.7321	0.8835	0.7197
task 3b	0.6973	0.6979	0.5660	0.7123
task 4a	0.9022	0.9415	0.8407	0.8340
task 4b	0.9052	0.7600	0.4634	0.9344

Table 2: correlation coefficients for frequency distribution comparisons between Bloodhound generated frequency vector versus user study data.

Bloodhound

- **Summary**
  - Information Foraging + SNIF-ACT provides the theory.
  - Bloodhound provides the usable system.
    - part theory, part database (for associations), part usable interface

- **What about Web 2.0 applications? They're working on it...**

Developing Web Interfaces

- **So far, we've focused on Java Swing, which is useful for standard & Web applications.**
- **Most "applications" on the Web, however, are not Swing applets.**
- **What else is used?**
  - So many different things, we can't cover them in 1 lecture, or even 10.
  - But let's check out the major ones.

Design Guidelines

- The points we've seen earlier with design guidelines all apply to Web design.
- The dominant way of achieving consistency in Web design: CSS.
- CSS = Cascading Style Sheets
  - Defines the style for HTML elements
  - Defines classes with particular styles
  - Basically: defines the look and feel of the web site, all in (sort of) one place

Design Guidelines

- CSS example

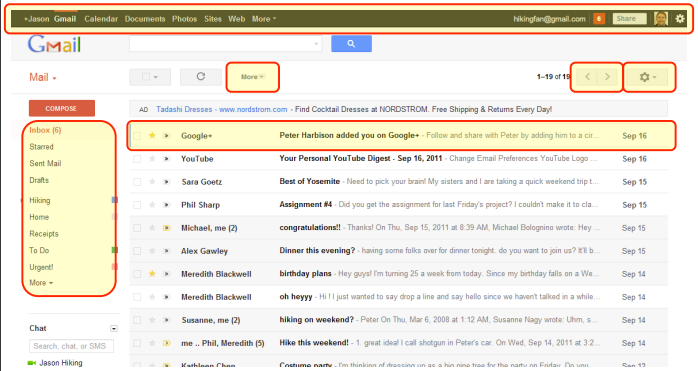
```
body {
        background-color: #dfa;
        font-family: "Times New Roman";
        font-size: 11pt;
}

h1 {
        font-size: 16pt;
        color: blue;
        text-align: center;
        margin-top: 12px;
}

h2 {
        font-size: 14pt;
        margin-top: 6px;
}
```


A Continuum: Application <--> Web Site

- A decade ago…
  - Application: on your computer, local data, many associated functions
  - Web site: delivered from elsewhere, remote data, a few targeted functions
- Nowadays…
  - Applications are connected, can use remote data
  - Web sites are looking more and more like applications
    - Fewer discrete (complete) state transitions
    - Dynamic feel, smooth animations
    - Mixture of local and remote data

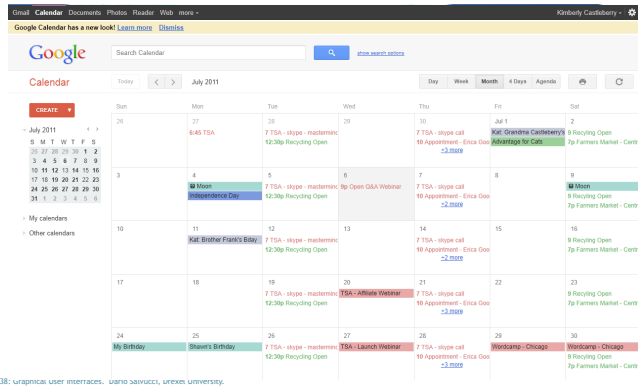
A Continuum: Application <--> Web Site

- Example: Google Mail (Gmail)



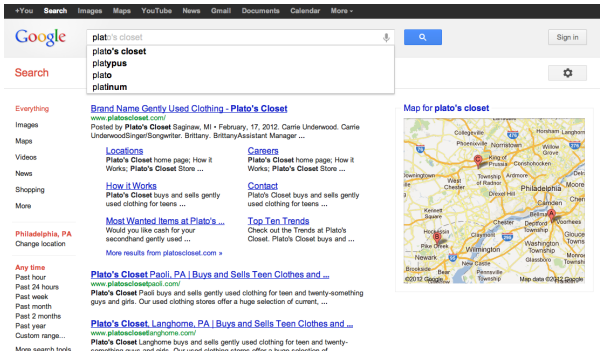
A Continuum: Application <--> Web Site

- Example: Google Calendar



A Continuum: Application <--> Web Site

- Mini-Example: Google suggester (w/ geo loc)



A Continuum: Application <--> Web Site

- The big difference now is the method of thinking about and handling events.
- Basic web page / core HTML
  - Basically one event: click on a link, go to the link
  - i.e., a discrete state change of the entire page
- Dynamic web page
  - Many types of events
  - Javascript coding to handle them
  - JavaScript ≠ Java!

JavaScript

- Dates back to 1995 (part of Netscape)
- Adopted by all major browsers today
- Interpreted
  - no compilation of code
- Weakly typed
  - code doesn't specify types of variables, arguments, etc.
- First-class functions
  - treats functions as values
  - store them, pass them as arguments, etc.

JavaScript

- Language tidbits
  - Many things look the same as Java
    - Operators

```
+, *, &&, ||, ==, …
```

    - Conditionals and loops

```
if, switch, for, while, …
```


JavaScript

- Language tidbits
  - Some things are a little different
    - Variable declaration

```
var a = 2;
```

    - Arrays and associative arrays

```
var myArray = [“hello”, “there”];
alert (myArray[0]);

var myAssoc = new Array();
myAssoc[“name”] = “John”;
alert (myAssoc[“name”]);
```


JavaScript

- Language tidbits
  - A straightforward recursive function

```
function factorial(n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
}
```


JavaScript

- Language tidbits
  - A function with *closure*

```
function displayClosure() {
    var count = 0;
    return function () {
        return ++count;
    };
}
var inc = displayClosure();
inc();
inc();
inc();
```

[from http://en.wikipedia.org/wiki/JavaScript]

JavaScript

- Including JavaScript in a web page

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head><title>simple page</title></head>
  <body>
    <h1 id="header">This is JavaScript</h1>

    <script type="text/javascript">
      document.write('Hello World!');
      var h1 = document.getElementById("header");
      h1 = document.getElementByTagName("h1")[0];
    </script>

    <noscript>
         <p>Your browser either does not support JavaScript,
            or has JavaScript turned off.</p>
    </noscript>
  </body>
</html>
```

[from http://en.wikipedia.org/wiki/JavaScript]

JavaScript

- Handling events
  - Embedding the handler in HTML

```
<button id="mybutton" onclick="doSomething()">
My Button
</button>
```

  - Specifying the handler in JavaScript

```
button.onclick = "doSomething()";
OR
button.onclick = doSomething;
OR
button.onclick = function() { … };
```

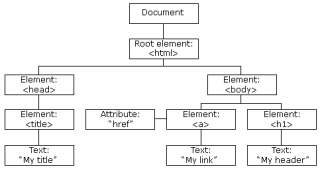

JavaScript

- What events can be handled?
- How do these events map onto Java Swing?
- A sampling…

JavaScript	Java Swing
onClick	ActionListener
onKeypress	KeyListener
onChange	ChangeListener
onMouseOver, onMouseOut	MouseListener
onMouseMove	MouseMotionListener
onFocus, onBlur	FocusListener
onSubmit, onLoad	<no real equivalent>

JavaScript

- What work is done inside the handler?
  - In Swing, we update the model & view
  - Same in JavaScript
- The model/view is the HTML page, or more accurately, the DOM = Document Object Model



[from www.w3schools.com]

JavaScript

- A handler function will typically…
  - Find DOM elements to update

```
var el = document.getElementById ("myelement");
```

  - Change some property of these elements

```
el.style.color = "blue";
el.style.fontSize = "11pt";
```

- [Note: Many use packages like jQuery as an easier way to find and change elements]

```
$("#myelement").css ('fontSize', '11pt');
$("div.myclass").css ('color', 'blue');
```


JavaScript

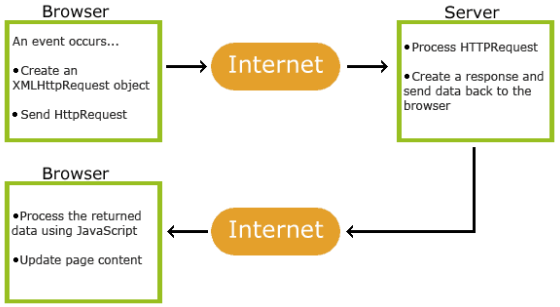
- Examples & demos
  - http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onclick
  - http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_iframe_contentdocument
  - http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onchange
  - http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onkeypress
  - http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onmouseover

AJAX

- So far, all the examples used data that are available locally
- What about remote data?
  - Suggester: needs a database of suggestions
  - Next-page button: needs the next page of results
  - Moving email to a folder: need to actually move the email on the server side, not just show it
- AJAX = Asynchronous JavaScript and XML
  - A way of transferring information between client (JavaScript) and server — all the while staying on the same web page

AJAX

- Core process / flow of information:



(from www.w3schools.com)

AJAX: Client Side

- XMLHttpRequest
  - The JavaScript object used for performing requests back to the server
  - (only back to the original server, for security purposes)
  - Can issue different kinds of requests
    - (we'll see this in a bit)

AJAX: Client Side

- XMLHttpRequest
  - Example: assume server response = "hello"

```
var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    document.getElementById("ml").innerHTML = xmlhttp.responseText;
}

xmlhttp.open("GET","server.php",true);
xmlhttp.send();
```

(from www.w3schools.com)

AJAX: Client Side

- XMLHttpRequest
  - Example: assumes XML response = <items><item>hi</item><item>hello</item><items>

```
var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {
  if (xmlhttp.readyState==4 && xmlhttp.status==200) {
    var items =
xmlhttp.responseXML.documentElement.getElementsByTagName("item");
    …
  }
}

xmlhttp.open("GET","server.php",true);
xmlhttp.send();
```

(from www.w3schools.com)

AJAX: Client Side

- Notes
  - XML isn't the only possible data format; could return JSON, or HTML, or other?
    - JSON = JavaScript object notation

```
[ {"text": "hello"}, {"text": "hi"} ]
```

  - jQuery provides convenient functions for AJAX

```
$.get ('server.php', {datum:37}, function (response) {
        alert (response);
      });
```

    - Note the asynchrony: function called only when response is received

AJAX: Server Side

- The AJAX client (JavaScript) gets its data from the server (web service)
- How is the server implemented?
  - Largely independent of the client side code
  - Could use…
    - PHP, commonly used with SQL databases
    - Java: servlets and JSP (JavaServer Pages)
    - Ruby on Rails, Django, etc.
  - … or anything really — as long as it delivers XML, or JSON, or whatever the client is expecting

AJAX: Server Side

- One method of handling data: REST
- REST = Representational State Transfer
  - GET: retrieve information, no side effects

```
http://www.myco.com/api/search?query=hello
```

  - POST (or PUT): create entry or update entry

```
http://www.myco.com/api/add
+ POST data (e.g., a file)
```

  - DELETE: delete entry
- This would be called a RESTful web service

Web UIs

- Layout
  - CSS
- Event handling
  - Javascript for event handling and page updates
  - AJAX for communication
    - Client side: GET, POST, etc. comments to web service
    - Server side: any dynamic generation of responses
- There are many other ways to do all this!
    - E.g., jQuery UI provides Swing-like widgets — again, different code/language/specifics, same idea