ICOM 6034
Website Engineering

Dr. Roy Ho
Department of Computer Science, HKU

Session 2: Enabling standards and technologies (Part 2)

Session objectives

- Web standardization
- HTML/XHTML
- CSS
- JavaScript
- Importance of separation between {document structure (HTML), presentation details (CSS), and behavior (JS)}

Covered in the last session (Session 1)

The four low-level, client-side technologies of the web

- Introduction to the Document Object Model (DOM) for naming/updating X/HTML elements
- The login page of Facebook.com as an integrated example of HTML + CSS + JavaScript + DOM
- Introduction to HTML5 and CSS3
- Lab 1B: JavaScript, HTML5 and CSS3

This Session

Quick review:

4. JavaScript and HTML DOM

JavaScript and DOM

- JavaScript relies on the Document Object Model (DOM) to access and manipulate individual elements in an HTML document

- The DOM is used for:
  - Capturing and handling events (e.g., mouse clicks, keystrokes, etc.)
  - Naming/updating individual elements in a web page (e.g., a particular paragraph <p>, a division , etc.)

- The basic DOM is a W3C standard and is the same across browsers
  - But similar to CSS/JS, some DOM features may not be supported in some browsers
  - So, developers prefer to use JavaScript frameworks/libraries (instead of using “raw” JS/DOM) which hide the differences of browsers.
    - E.g., jQuery, Bootstrap, etc.
    - More on this in Session 4

Events

- Most elements in a web page respond to user actions (e.g., mouse clicks, keystrokes, etc.) by creating events
  - Different elements produce different events

- You can attach event handlers to HTML elements (e.g., a form), which would be executed when the events occur

- A simple event handler:
  -
```
<form method="post" action="">
  <input type="button"
         id="myButton"
         value="Click me"
         onclick="alert('You clicked the button!');">
</form>
```

  - If the event is not triggered, the handler would do nothing

Common events

- Most HTML elements produce the following events:
  - onclick – the element is clicked
  - ondblclick – the element is double-clicked
  - onmousedown -- the mouse button is pressed while over the element
  - onmouseover -- the mouse is moved over the element
  - onmouseout -- the mouse is moved away from the element
  - onmouseup -- the mouse button is released while over the element
  - onmousemove -- the mouse is moved

- A special event: onload, which would be triggered when the page is fully loaded, e.g., you may show a welcome message to the user once the page is loaded.

- Traditionally, people like to use capital letters in naming HTML events (e.g., “onClick” rather than “onclick”), but lower case should be used for HTML5
  - Note: the use of lower case is enforced in XHTML

This is called “camelCase”.

Example: mouse-over

- The following code will make the text Hello **red** when the mouse moves over it, and **blue** when the mouse moves away

```
<h1 onmouseover="style.color='red';"
    onmouseout="style.color='blue';">Hello</h1>
```

These events and attributes are defined by DOM

- Image rollovers:

```
<img src="../Images/pic1.gif"
    onmouseover="src='../Images/pic2.gif';"
    onmouseout="src='../Images/pic1.gif';">
```

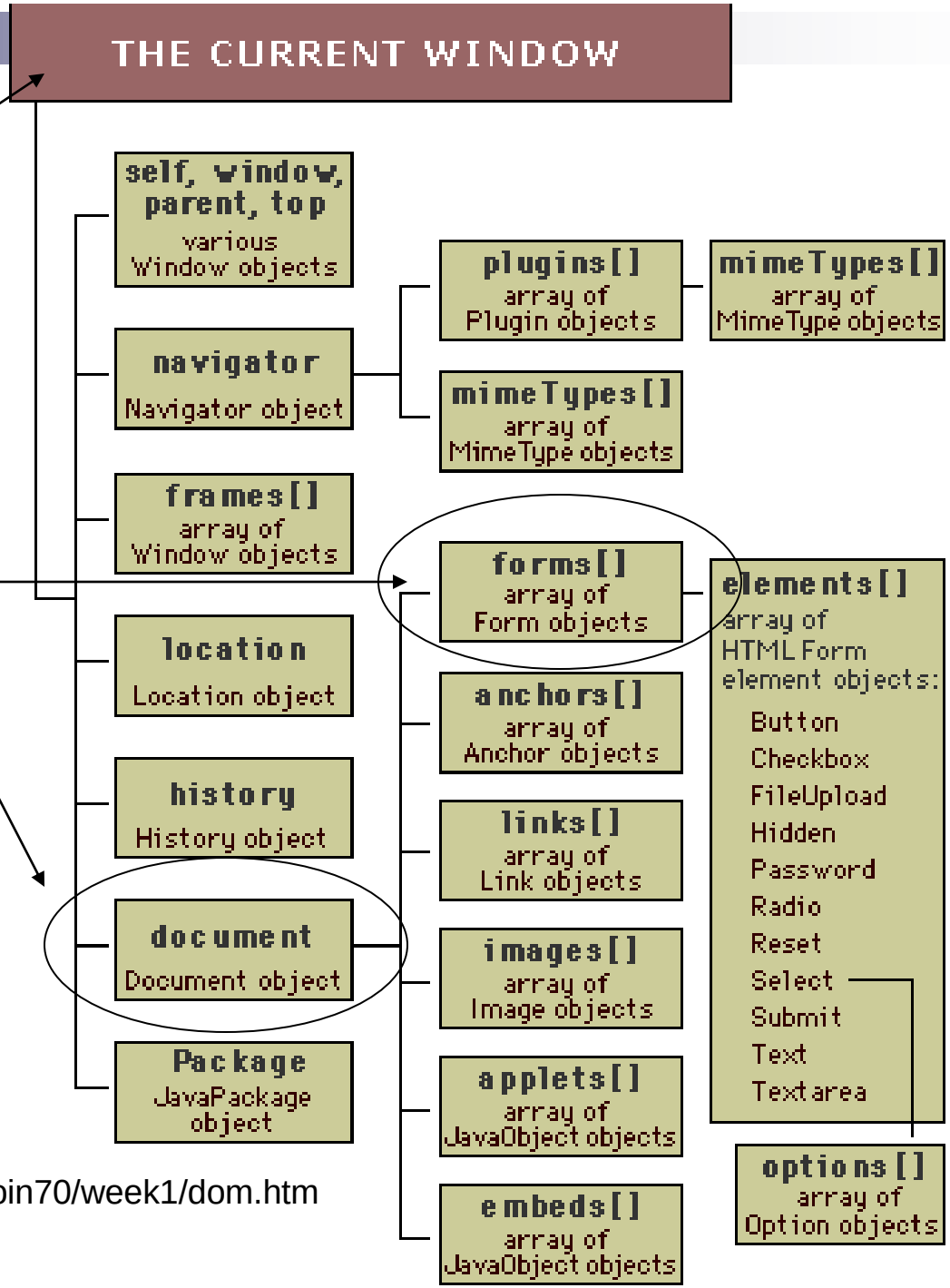
Naming: the window and document objects

- In HTML DOM, the *window* (i.e., the browser window/tab) is the "highest-level" element
  - All other elements can be reached by traversing down from there
  - The most important window property is document (the current page), which stores all HTML elements in the current page
  - E.g., window.document.myForm.myButton
  - It is assumed that all variables are properties of the "window" object, so the use of "window." can be omitted:
  - E.g., document.myForm.myButton

The DOM hierarchy

DOM defines that:

- The entire document is a document node
- Every HTML tag is an element node



Source: http://sislands.com/coin70/week1/dom.htm

Properties of “window”

(for reference only)

- parent
  - If in a frame, the immediately enclosing window.
- top
  - If in a frame, the outermost enclosing window.
- frames[ ]
  - An array of frames (if any) within the current window. Frames are themselves windows.
- length
  - The number of frames contained in this window.
- document
  - The HTML document being displayed in this window.
- location
  - The URL of the document being displayed in this window. Setting this property to a new URL and calling location.reload() will refresh the window.
- navigator
  - A reference to the Navigator (browser) object. Some properties of Navigator are:
    - userAgent -- the name of the browser, such as “Safari"
    - platform -- the OS running the browser, such as "Win64"
- status
  - A read/write string displayed in the status area of the browser window. Can be changed with a simple assignment statement.

Methods of “window”

(for reference only)

- alert(string)
  - Displays an alert dialog box containing the string and an OK button.
- confirm(string)
  - Displays a confirmation box containing the string along with Cancel and OK buttons. Returns true if OK is pressed, false if Cancel is pressed.
- prompt(string)
  - Displays a confirmation box containing the string, a text field, and Cancel and OK buttons. Returns the string entered by the user if OK is pressed, null if Cancel is pressed.
- open(URL)
  - Opens a new window and load the web page hosted at URL.

Properties of “document” (1/2)

(for reference only)

- anchors[ ]
  - An array of Anchor objects (i.e., links with <a name=...> tags)
- applets[ ]
  - An array of Applet objects
    - The properties are the public fields defined in the applet
    - The methods are the public methods of the applet
    - Cautions:
      - You must supply values of the correct types for the fields and method parameters
      - Changes and method calls are done in a separate thread
- forms[ ]
  - An array of Form objects
    - If the document contains only one form, it is forms[0]
- images[ ]
  - An array of Image objects
    - To change the image, assign a new URL to the src property

Properties of “document” (2/2)

(for reference only)

- links[ ]
  - An array of Link objects linking to external documents (e.g., CSS, JS, etc.)
- bgColor
  - The background color of the document
    - May be changed at any time
- title
  - A read-only string containing the title of the document
- URL
  - A read-only string containing the URL of the document

Properties of a “form” object

- “form” is probably the most frequently-used DOM element. You can use it to retrieve and validate user input and change the behavior of the form.

- elements[ ]
  - An array of form elements

- By checking the values of form elements, one can do data validations before submitting a form to the server. Common validations include:
  - Detect empty text boxes (e.g., user’s last name)
  - Ensure that at least one radio button is selected
  - Respond to checked options in check boxes
  - Validate an email address
  - ……

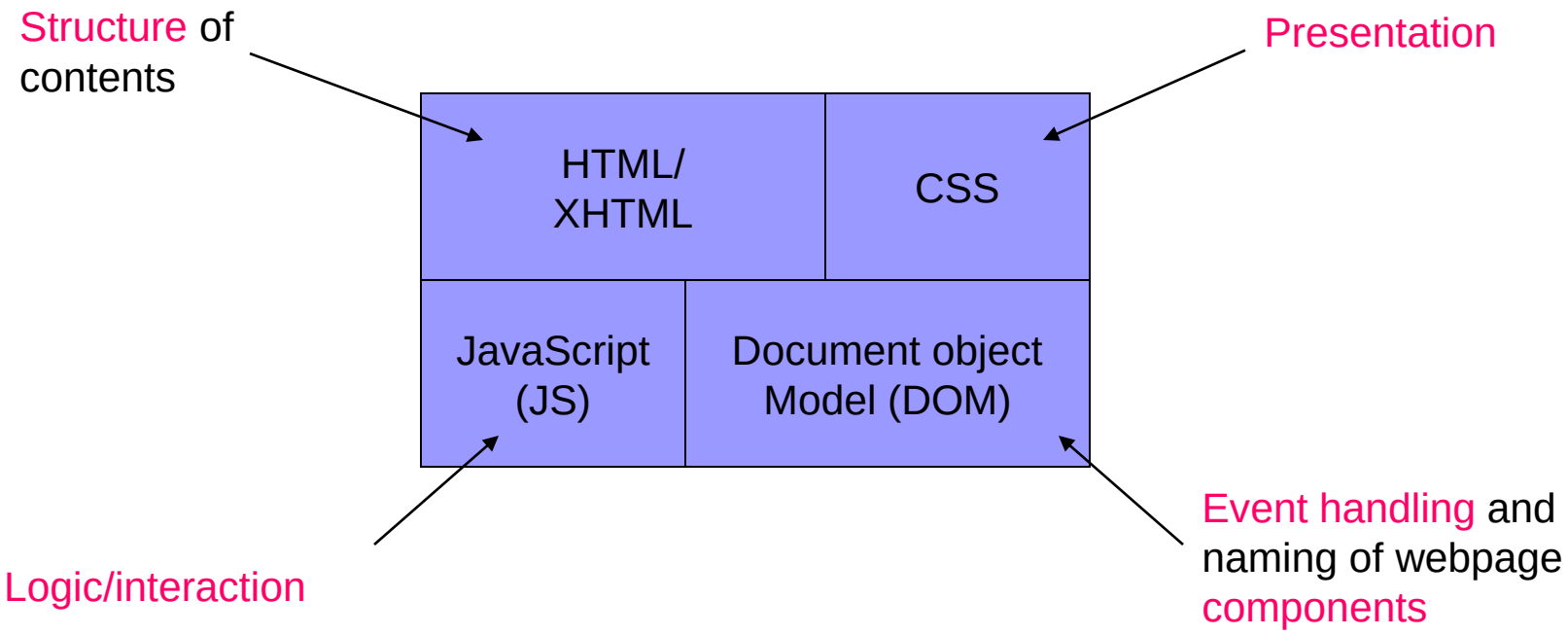
“Form” and element searching

- Examples of “form”:
  - E.g., HTML form elements can be referred to by
    document.forms[***formNumber***].elements[***elementNumber***]
  - If an HTML form element has an id attribute
    - The id can be used instead of numeric array index
    - Hence, if
      - <form id="myForm">
        <input type="button" id="myButton" ...>
      - Then instead of document.forms[0].elements[0]
      - you can use document.myForm.myButton
- Element searching:
  - E.g., to search for all <p> elements in the document:
    - document.getElementsByTagName("p");
  - To return all <p> elements that are “children” of the element with id="maindiv":
    - document.getElementById('maindiv').getElementsByTagName("p");

HTML DOM reference

- For a complete list of DOM methods and properties, along with examples, see:
  - HTML DOM Reference
  - https://www.w3schools.com/jsref/default.asp

Summary: building blocks of a webpage



Putting all together

Example: Login page of Facebook

Example: Facebook's login page

http://www.facebook.com/login.php
copyright: Facebook

Not as simple as it seems

- HTML
  - Various HTML elements and forms
- CSS
  - Colors, fonts, positioning, visibility, …
- JavaScript (and DOM)
  - Events, cookies, HTML elements updates, etc.

- Note: Facebook has been updated to use HTML5.
  - We will still look at its previous XHTML version, which only has non-semantic (“generic”) HTML elements
  - The current HTML5 code has the same organization as the XHTML version, but with semantic markup – more on this later

HTML - main structure

The strict DTD is being used

Document type definitions

HTML header

HTML body

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" id="facebook">
    <head>
        <title>Login | Facebook</title>
        <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
        <link rel="stylesheet" href="http://static.ak.facebook.com/rsrc.php/86877/css/bas
        <style type="text/css">..
        <meta id="robots" name="robots" content="noodp"/>
        <meta id="description" name="description" content="Facebook is a social utility t
        <!--[if lte IE 6]><style type="text/css" media="screen">/* <![CDATA[ */ @import u
        <link rel="search" type="application/opensearchdescription+xml" href="http://stat
        <link rel="shortcut icon" href="http://static.ak.facebook.com/favicon.ico"/>
    </head>
    <body class="login">
        <div id="book">
            <div id="sidebar">..
            <div id="widebar" class="clearfix">
                <div id="navigator">..
                <div id="page_body" class="pagebody login">..
                <div id="pagefooter" class="clearfix">..
            </div>
        </div>
        <script type="text/javascript">..
        <script type="text/javascript">..
    </body>
</html>
```

Some codes are hidden here

HTML header

- <head> … </head>: the header contains some “metadata” about the page, e.g., document title, document base URL, links to external documents, and other meta information
  - <title> … </title>: Sets the document title
  - <base href="..." />: Specifies a base URL for all links in the page
  - <link rel="[1]" type="[2]" href="[3]" />: Link to an external document at location [3] of MIME type [2] for use as [1]
    - example: <link rel="stylesheet" type="text/css" href="style.css" />
  - <meta>: meta-information about the document
    - <meta http-enquiv="..." content="...." />: “HTTP-header like” attributes, to specify MIME type, character set, cookies, etc.
      - example: <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
    - <meta name="..." content="..." />: textual descriptions. Usually for specifying keywords and descriptions for search engines
      - example: <meta name=“keywords" content=“HKU, ECOM, ICOM, the Web"/>

<title> … </title>: Document title

<meta http-equiv="content-type" …>: This is a text/html document in UTF-8

<link rel="stylesheet" ...>: Link to the external stylesheet

<style> … </style>: An embedded stylesheet (hidden here)

```
<head>
    <title>Login | Facebook</title>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
    <link rel="stylesheet" href="http://static.ak.facebook.com/rsrc.php/86877/css/base.css?57:86877" type="text/css"/>
    <style type="text/css">
    <meta id="robots" name="robots" content="noodp"/>
    <meta id="description" name="description" content="Facebook is a social utility that connects people with friends a
    <!--[if lte IE 6]><style type="text/css" media="screen">/* <![CDATA[ */ @import url(http://static.ak.facebook.com/c
    <link rel="search" type="application/opensearchdescription+xml" href="http://static.ak.facebook.com/opensearch_desc
    <link rel="shortcut icon" href="http://static.ak.facebook.com/favicon.ico"/>
</head>
```

<meta name="robots" ...>: *Tells* search engines what to do with this document
http://googlewebmastercentral.blogspot.com/2007/03/using-robots-meta-tag.html

<meta name="description" ...>: Textual descriptions about this page

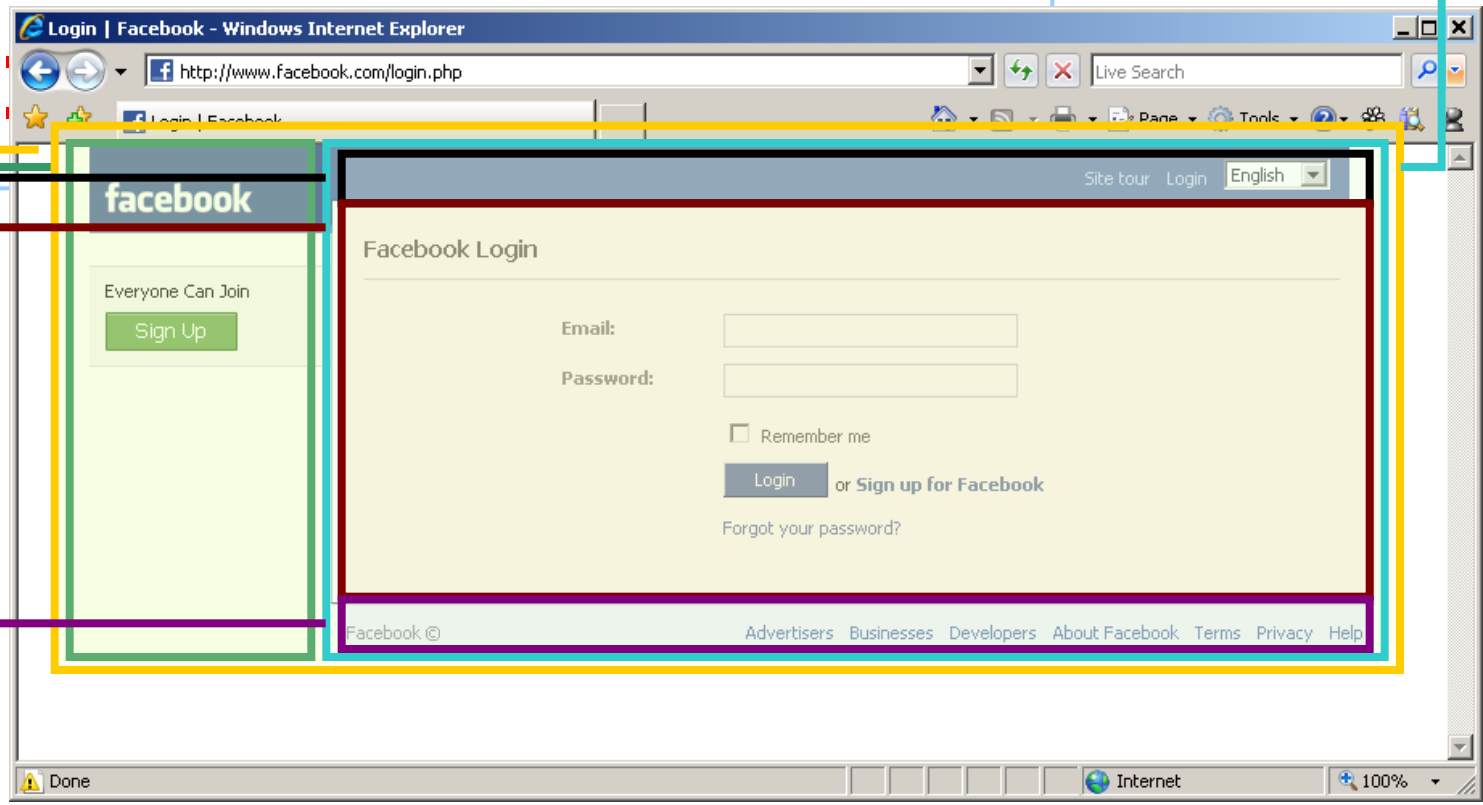
Some Internet Explorer specific settings

<link rel="search" …>: To describe searching capabilities. http://www.opensearch.org/

<link rel="shortcut icon" …>: Icon for this page

Main divisions

```
<body class="login">
    <div id="book">
        <div id="sidebar">..
        <div id="widebar" class="clearfix">
            <div id="navigator">..
            <div id="page_body" class="pagebody login">..
            <div id="pagefooter" class="clearfix">..
        </div>
    </div>
    <script typ
    <script typ
</body>
```



The login form

(inside page_body)

This form will be submitted to https://login.facebook.com/login.php using the POST method.

```
<form method="post" action="https://login.facebook.com/login.php">
    <div id="loginform">
        <div class="form_row clearfix">
            <label class="" for="email" id="label_ema
                Email:
            </label>
            <input type="text" class="inputtext" id="
        </div>
        <div class="form_row clearfix">
            <label class="" for="pass" id="label_pass">
                Password:
            </label>
            <input type="password" class="inputpassword" id="pass" name="pass" value=""/>
        </div>
        <label class="persistent">
            <input type="checkbox" class="inputcheckbox" onclick="document.getElementById(&qu
        </label>
        <div style="display: none" id="persistent_notification">
            <div class="status" id="standar_status">
                <h2><span id=status_title>By selecting "remember me" you will stay logged int
            </div>
        </div>
        <div id="buttons" class="form_row clearfix">
            <label class="">
            </label>
            <input type="submit" value="Login" name="login" id="login" onclick="this.disabled
        </div>
        <p class="reset_password form_row">
            <label class="">
            </label>
            <a href="http://www.facebook.com/reset.php">Forgot your password?</a>
        </p>
    </div>
    <input type="hidden" id="charset_test" name="charset_test" value="&euro;                    ;"/>
</form>
```

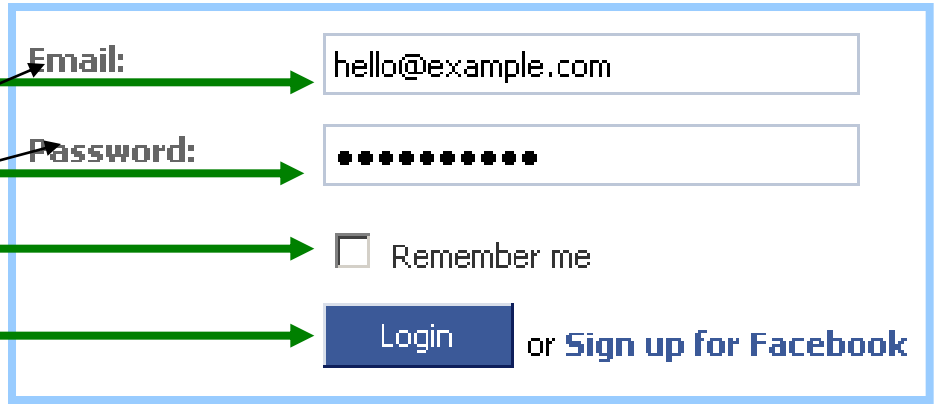
Form elements

Input types
text: normal textbox
password: masked text
checkbox: yes/no
submit: to submit the form

Other input types
hidden, **radio**, **button**, **reset**, **file**, **image**

value
text/password/hidden: default text
submit/reset/button: text on button
checkbox/radio: value when selected

<label>: Connects the content to a form element. Matches "for" with "id". Clicking on the label will select/"focus" the form element.




```
<label class="" for="email" id="label_email">
    Email:
</label>
<input type="text" class="inputtext" id="email" name="email" value="" />

<label class="" for="pass" id="label_pass">
    Password:
</label>
<input type="password" class="inputpassword" id="pass" name="pass" value="" />

<label class="persistent">
    <input type="checkbox" class="inputcheckbox" id="persistent" name="persistent" value="1" />
    <span id="remember_me_text">Remember me</span>
</label>

<div id="buttons" class="form_row clearfix">
    <input type="submit" value="Login" name="login" id="login" class="inputsubmit"/>
     or <strong><a href="http://www.facebook.com/r.php?" id="reg_btn_link">Sign up for Facebook</a></strong>
</div>
```

name
For identifying the form fields at the server side

CSS

- This page uses a lot of CSS, defined in an external and an embedded stylesheet
- The page would look like this when all CSS is removed
- Note that the "separation" of document structure (content) and presentation details is done very well in this page!

```
<style type="text/css">
    #content {
    .title_header {
    .title_header h2.no_icon {
    .login #loginform, #resetform {
    .login #loginform p, #resetform p {
    div.pwresetmain {
    .apinote {
    .apinote h2 {
    #booklet #content {
    #booklet #loginform {
    .form_row {
    .form_row label {
    .form_row input {
    .form_row .inputtext, .inputpassword {
    .form_row .checkbox {
    .save_login_text {
    .persistent {
    #persistent_notification .status {
    #persistent_notification .status h2 {
    #buttons {
    #buttons input {
    #share_login {
    #company_description {
    #share_prompt {
    .reset #content {
    #book #border_content_shadow {
</style>
```

The embedded stylesheet
Defined inside <head>

```
#error, .status, .explanation_note {
.created {
#error p, .status p, .explanation_note p {
#error a, .status a, .explanation_note a {
.status {
.status a {
.page_top_notice {
.explanation_note {
.explanation_note h1 {
.pipe {
.column {
.center {
.editor_title {
.standard_title {
.page_title {
.standard_title .page_title {
.needs_translated {
#book {
#sidebar {
#sidebar h2 {
#sidebar h2 a {
#sidebar h2 a:hover {
#sidebar a.go_home {
#sidebar a.go_home:hover {
#sidebar a.go_home h1 {
#sidebar_content {
#sidebar_signup_content {
#squicklogin {
#squicklogin label {
#squicklogin .inputtext {
#squicklogin .inputsubmit {
#squicklogin_label.persistent {
```

(Part of) the external stylesheet
via <link rel="stylesheet">

The sidebar

```
<div id="sidebar">
  <a href="http://www.facebook.com" class="go_home"
 style="background-image:url(http://static.ak.facebook.com/images/faceb
ook_logo.gif?57:67387)"></a>
  <div id="sidebar_content">
  </div>
  <div class="clearfix">
    <div id=sidebar_signup_content>
      Everyone Can Join
      <a href="http://www.facebook.com/r.php?r=200"
 class="link_btn_style  reg_btn_style" id="reg_btn_link">
        <div><div><div>
          <span class="btn_text">Sign Up</span>
        </div></div></div>
      </a>
    </div>
  </div>
</div>
```

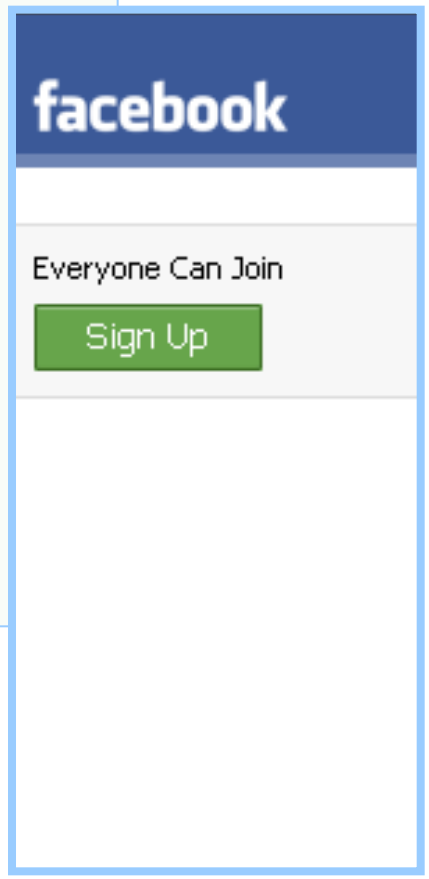
```
#sidebar {
    padding: 0px 0px 0px;
    font-size: 13px;
    width: 150px;
    float: left;
}
```

No “padding”

Font size 13

150 pixels wide

On the left of “container”



More examples of padding & margin

```
<style>
.padding {
  padding: 0px 10px 0px 10px;
  border: 1px solid red;
}

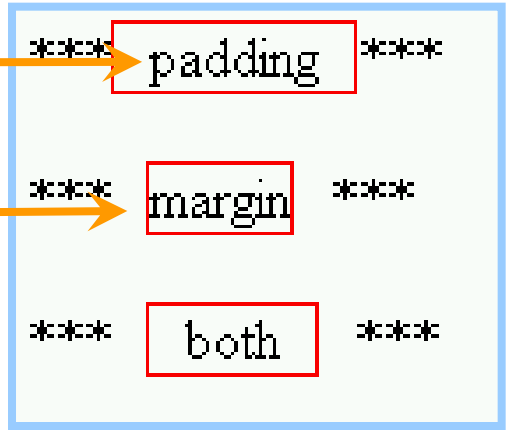
.margin {
  margin: 0px 10px 0px 10px;
  border: 1px solid red;
}

.both {
  margin: 0px 10px 0px 10px;
  padding: 0px 10px 0px 10px;
  border: 1px solid red;
}
</style>
<body>
***<span class="padding">padding</span>***<br><br>
***<span class="margin">margin</span>***<br><br>
***<span class="both">both</span>***<br><br>
</body>
```

Shortcut for defining padding & margin, in order of *top, right, bottom, left*.

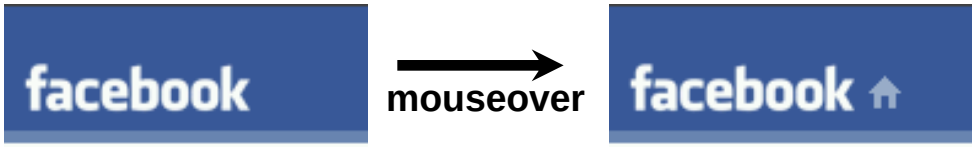
You may provide 2 or 3 numbers instead of 4, e.g., margin: 0px 10px

Padding: space *inside* the boundary



Margin: space *outside* the boundary

They did mouseover in a tricky way…using CSS!



This background image is actually 2 images of 151x55 pixels combined into a single image

```
<div id="sidebar">
  <a href="http://www.facebook.com" class="go_home"
 style="background-image:url(http://static.ak.facebook.com/images/faceb
ook_logo.gif?57:67387)"></a>
```

This would select the <a> element with class="go_home" contained in "sidebar"

```
#sidebar a.go_home {
    display: block;
    background-color: #3b5998;
    background-position: top left;
    background-repeat: no-repeat;
    height: 26px;
    width: 151px;
    position: absolut
    z-index: 3;
    margin: 0;
    padding: 29px 0px 0px;
    font-size: 13px;
}
#sidebar a.go_home:hover {
    background-color: #3b5998;
    background-position: bottom left;
    background-repeat: no-repeat;
}
```

positioning

padding-top 29px + height 26px
➔ This <a> is 151 pixels wide, 55pixels high (151x55)

hover: when the mouse is put over that <a> element

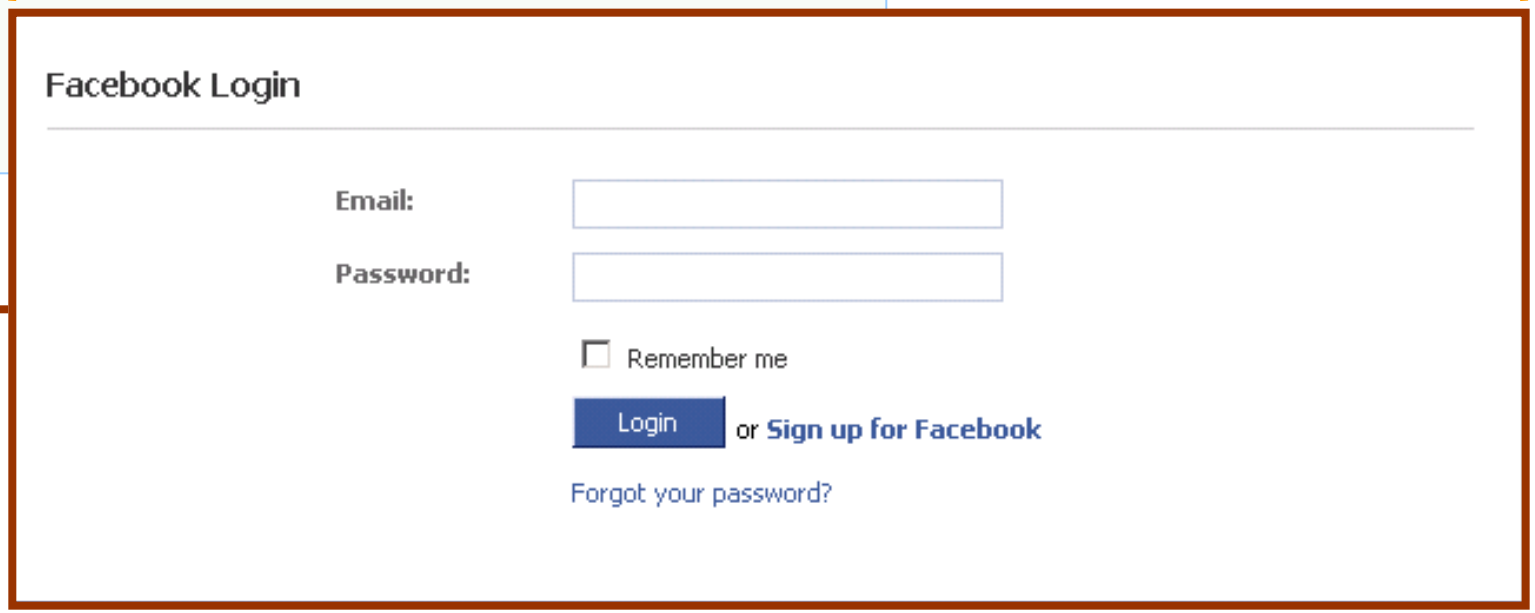
The background image's position changes from top-left to bottom-left aligned inside <a>

The “page_body”

```
<body class="login">
 <div id="book">
  <div id="widebar" class="clearfix">
   <div id="page_body" class="pagebody login">
    <div id="content_shadow">
     <div id="content" class="clearfix">
      <div class="login_title_header shorten add_border">
       <h2 class="no_icon">Facebook Login</h2>
      </div>
      <form method="post" action="https://login.facebook.c
       <div id="loginform">
       <input type="hidden" id="charset_test" name="charse
      </form>
     </div>
    </div>
   </div>
  </div>
 </div>
</body>
```

```
#widebar {
    width: 649px;
    float: left;
}
```

Contained in “widebar”;
Occupied its full width



Facebook Login

Email:

Password:

Remember me

Login or Sign up for Facebook

Forgot your password?

20 Auto margin

Auto margin 20

1px padding + 20px margin

page_body: The content

```
<div id="content" class="clearfix">
 <div class="login_title_header shorten add_b
  <h2 class="no_icon">Facebook Login</h2>
 </div>
 <form method="post" action="https://login.fac
  <div id="loginform">..
```

```
#content {
    margin: -2px 1px 0px -1px;
    border-top: none;
    border-left: solid 1px #b7b7b7;
    border-right: solid 1px #b7b7b7;
    border-bottom: solid 1px #3b5998;
    font-size: 11px;
}
```

no conflict

```
#content {
    padding: 1px 20px;
}
```

```
.login_title_header {
    padding: 20px 0px 10px;
    margin-bottom: 10px;
    border-bottom: solid 1px #ccc;
}
```

```
.login #loginform, #resetform {
    text-align: left;
    margin: 20px auto;
    width: 360px;
}
```

auto: calculate automatically

```
.login_title_header h2 {
    margin: 0px;
    padding: 0px;
    font-size: 14px;
}
```

JavaScript in the <script> block

Statements to execute

Function declarations

```
<script type="text/javascript">
  if (!window.ge) {
      window.ge = function(id){
          return document.getElementById(id);
      }
  }
  window.onload = function(){
      document.cookie = "test_cookie=1;domain=.facebook.com";
      var e = ge('email'), p = ge('pass');
      (e.value ? p : e).focus();
  };
  function formchange(){
      (ge('persistent') ||
      {}).checked = 0;
  }

  function pop(url){
      window.open(url);
  }
</script>
```

```
if (!window.ge) {
    window.ge = function(id){
        return document.getElementById(id);
    }
}
```

If window.ge is not defined {
define window.ge as a { function which takes a parameter id
and returns the JS object of the element identified by that id
}
}

Redefining the window's onLoad event handler function, which is triggered when the web page is completely loaded

```
window.onload = function(){
    document.cookie = "test_cookie=1;domain=.facebook.com";
    var e = ge('email'), p = ge('pass');
    (e.value ? p : e).focus();
};
```

Put something into the user's cookie

Details in the next slide…

getElementById()

```
<input type="text" class="inputtext" id="email"
name="email" value="" onkeypress="formchange()"/>

<input type="password" class="inputpassword" id="pass"
name="pass" value=""/>
```

Email:

Password:

Focus is by default put on "e" when "email" is empty, otherwise on "p" (e.g., when the browser fills in any saved email address)

```
window.onload = function(){
    document.cookie = "test_cookie=1;domain=.facebook.com";
    var e = ge('email'), p = ge('pass');
    (e.value ? p : e).focus();
};
```

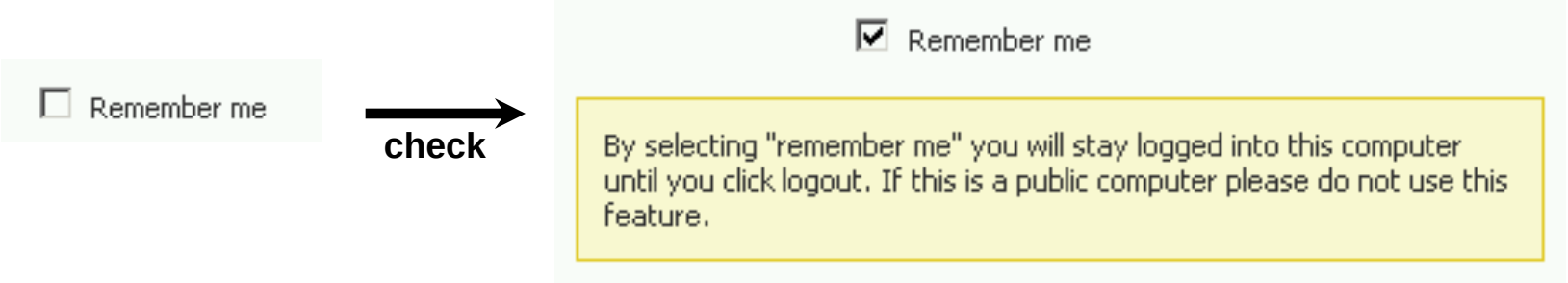
"(a ? b : c)" is a conditional assignment:-
If "a" is true or not-empty, the entire expression would be equal to "b", otherwise it would be equal to "c".

```
e = getElementById('email')
```

```
p = getElementById('pass')
```

e and p are objects representing the input boxes:
eg: *[object]*.value: the content in the box
eg: *[object]*.focus( ): focus on the element (put the cursor to it)

The hidden dialog box



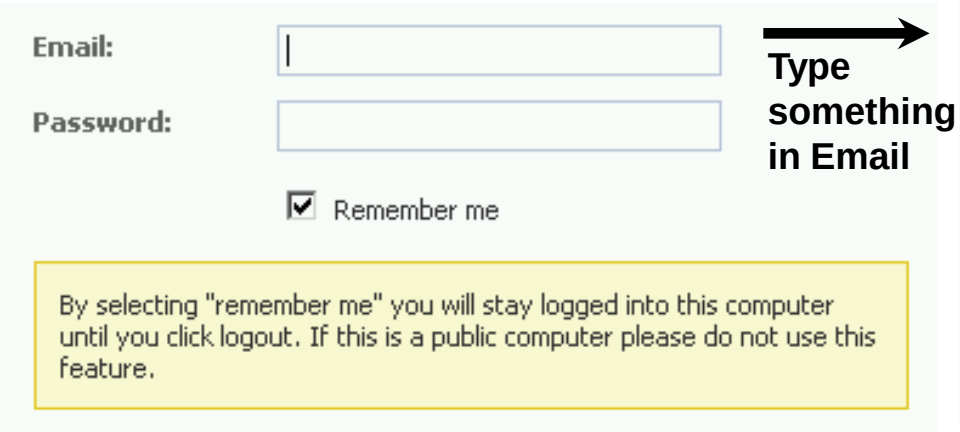

```
<label class="persistent">
 <input type="checkbox" class="inputcheckbox"
onclick="document.getElementById("persistent_notification") style.display=this.checked?"block":"none";
 id="persistent" name="persistent" value="1"/><span id="remember_me_text">Remember me</span>
</label>
<div style="display: none" id="persistent_notification">
 <div class="status" id="standar_status">
  <h2><span id=status_title>By selecting "remember me" you will stay logged into this computer until
    you click logout. If this is a public computer please do not use this feature.</span></h2>
  </div>
</div>
```

[CSS] *display* property controls whether or not to display the element. The message is not displayed by default

The onClick event handler for this checkbox: sets the *display* property of persistent_notification object accordingly

***this*.checked: a boolean field in a checkbox object. True when checked**
***this* is a special reference to the object of current context**

Another event handler



Type something in Email

Unchecked!

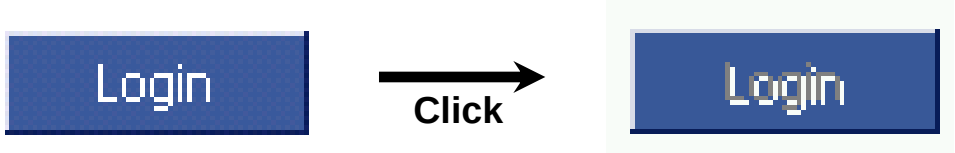
```
<input type="text" class="inputtext" id="email"
name="email" value="" onkeypress="formchange()"/>
```

```
function formchange(){
    {ge('persistent') || {}}.checked = 0;
}
```

The onKeyPress event: triggered when pressing a key when this element is focused

"persistent"is the checkbox's id

The submit button




```
<input type="submit" value="Login" name="login" id="login"
onclick="this.disabled=true; this.form.submit();" class="inputsubmit"/>
```

The onClick event is triggered when this button is clicked.
After clicking, the button is disabled before submitting the form.
This is a way to prevent multiple form submissions

Introduction to HTML5 and CSS3

HTML5 and CSS 3

HTML5

- Simplifies things that had to be done clumsily with XHTML / HTML 4
- New features for web (2.0) applications
- Friendly to search engines
- *HTML5 updates HTML 4.01*

CSS level 3

- Makes it much easier to do common but tedious designs
- *CSS 3 updates CSS level 2 (CSS 2.1)*

Main new features of HTML5

- New document structures like article, footer, header, nav (navigation section), section… having semantic meanings
  - As compared to HTML 4's generic <h1>, , , <p>, etc.
  - Any advantages?
- Other new elements, e.g.,:
  - The "M" tag
  - The canvas element for drawing
  - The video and audio elements for media playback
- New form controls, like calendar, date, time, email, url, search
- Better support for offline applications and storage
- Deprecated elements are removed: acronym, applet, basefont, big, center, dir, font, frame, frameset, noframes, strike, tt, u

Anything common for these retired elements?

Hint: "separation" between document structure and presentation details.

- A hello-world HTML5 page:

```
<!DOCTYPE html>
    <title>Small HTML 5</title>
    <p>Hello world</p>
</html>
```

- Note the new doctype in <html>
- <body>, <head> not essential
- Other syntax identical to HTML 4.01

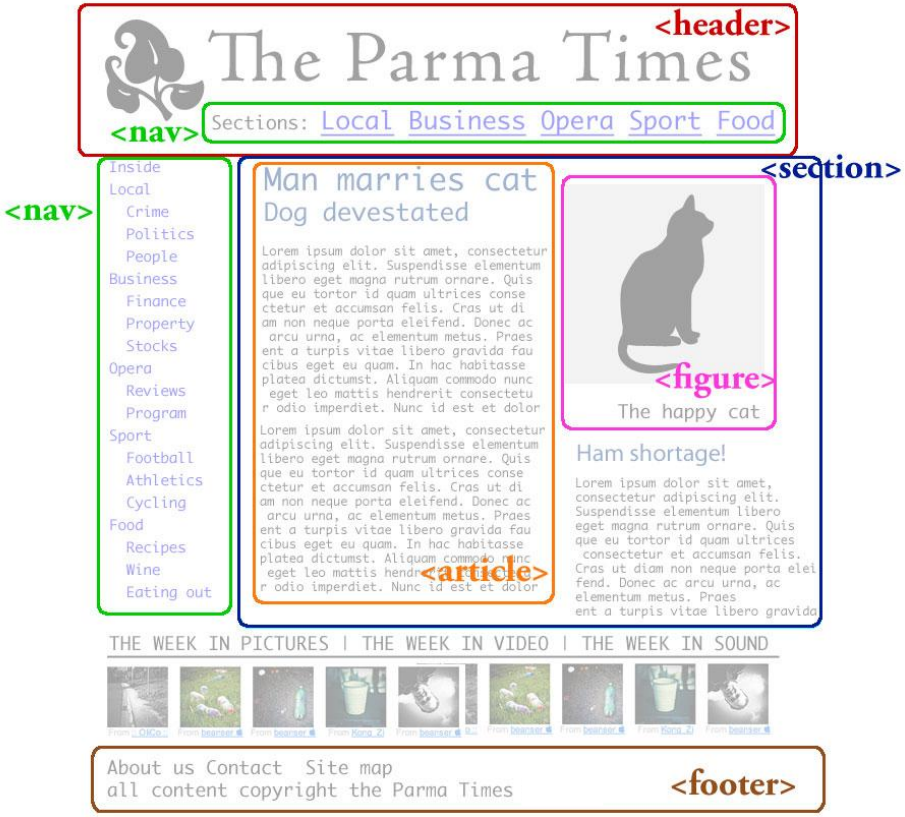
Markup in the past

- In XHTML/HTML4, if we want to markup (format) this page we would use a lot of tags, and classes.
- Poor semantic value of and ‘class’
- Can easily lead to overuse of and “class”
- Confusing to search engines/maintainers

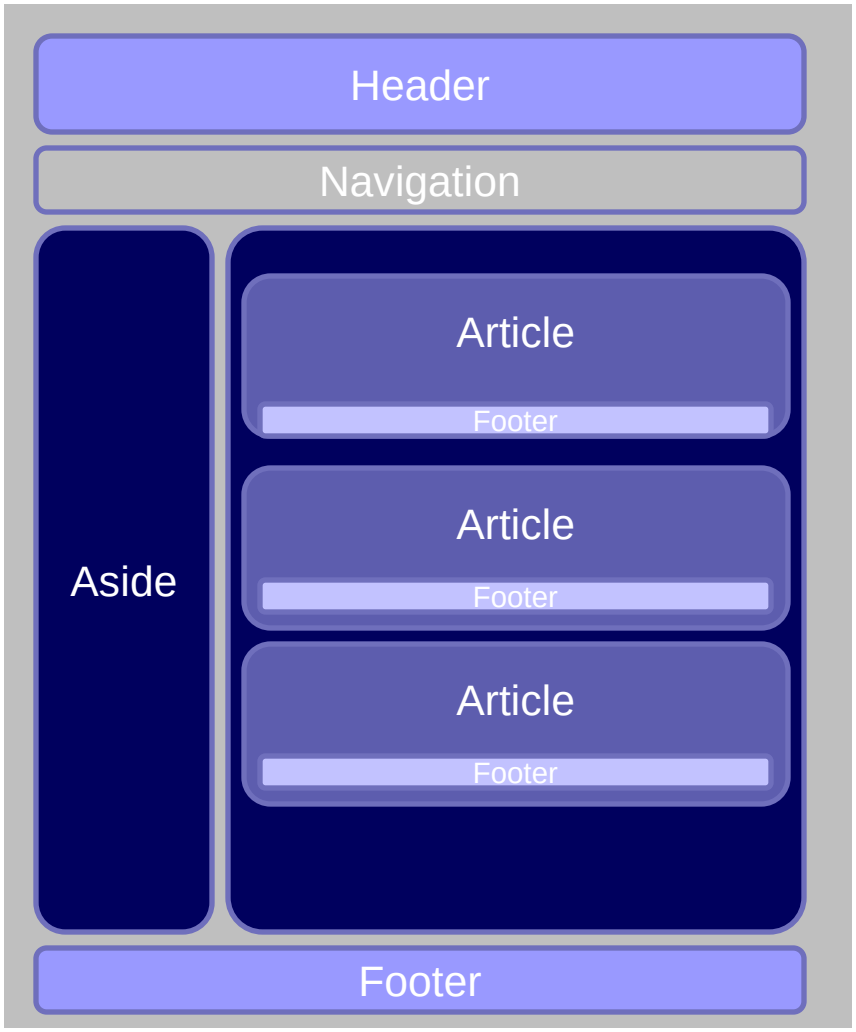
HTML5: new document structure

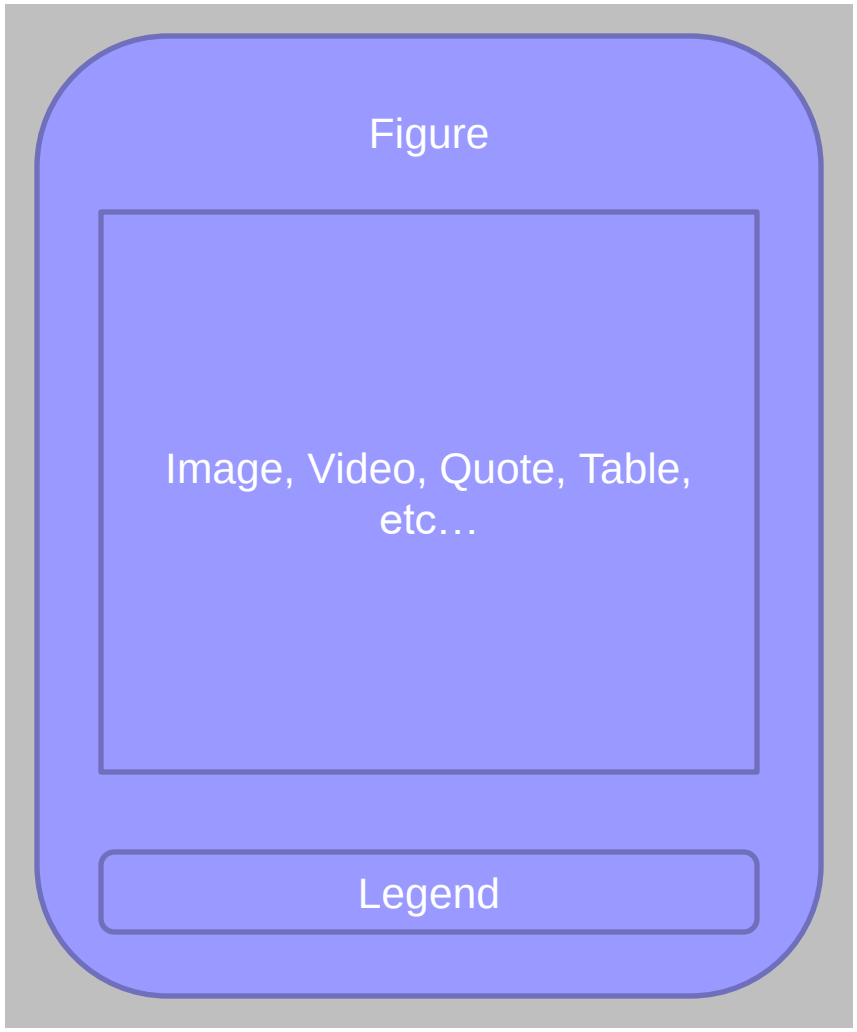
- <header>, <nav> (a set of navigation links), <article>, <section>, section numbering and other new tags.

- It's good for search engines, information architects, and the web in general.

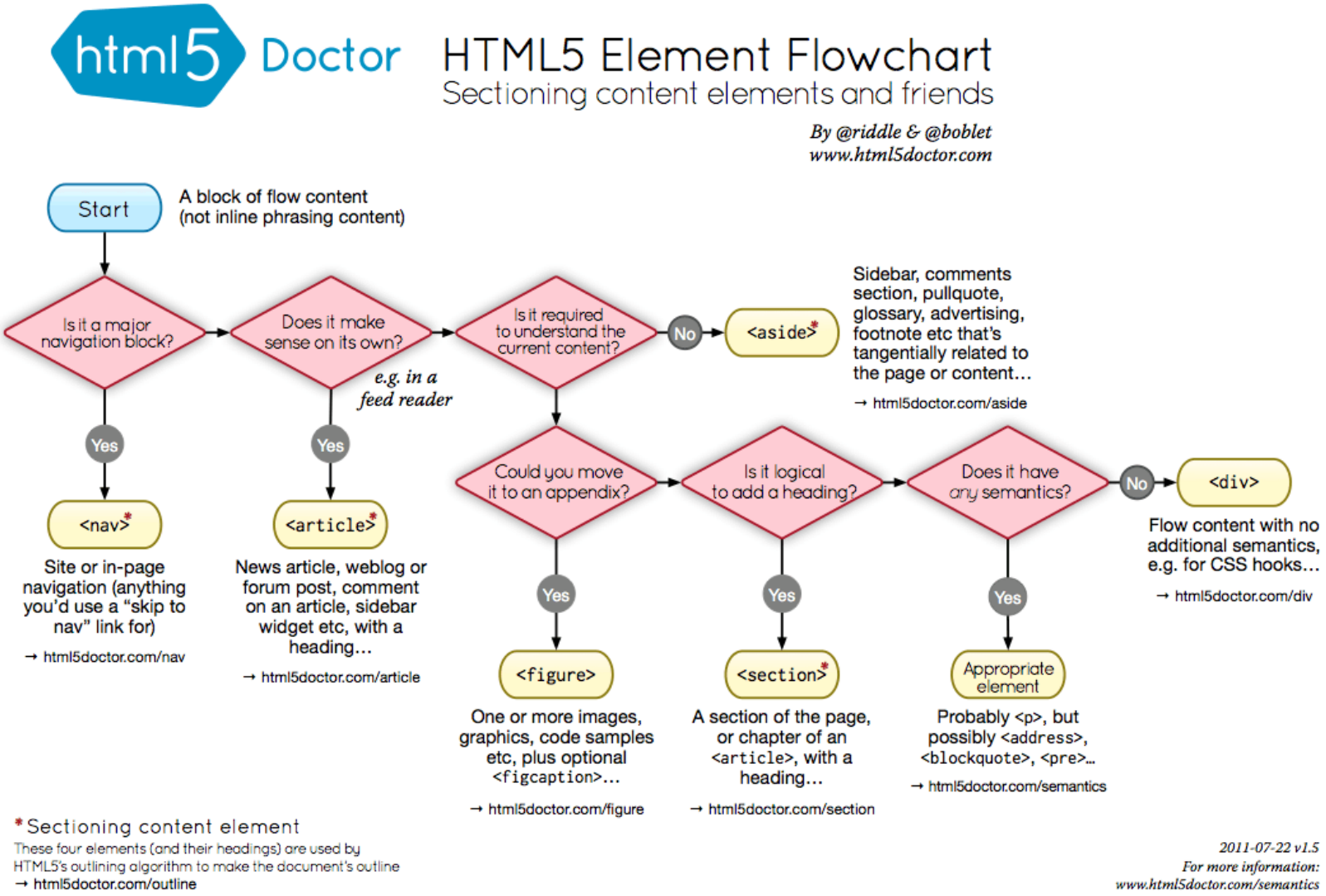


HTML5: new document structure





HTML5 – Element flowchart



http://html5doctor.com/downloads/h5d-sectioning-flowchart.png

HTML5: the new “M” tag

- Used for highlighting texts, e.g.,
  - Highlighting searched keywords in a document
  - Highlighting relevant code in a code sample

The highlighted part below is where the error lies:

```
var i: Integer;
begin
   i := 1.1;
end.
```

- ```
  The highlighted part below is
     where the error lies:
  <code>var i: Integer;
  begin
     i := <m>1.1</m>;
  end.</code>
  ```
  ```

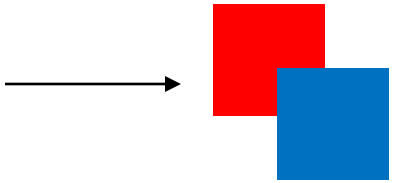
Canvas

- Dynamic and interactive graphics
- Draw images using 2D drawing API
  - Lines, curves, shapes, fills, etc.
- Useful for:
  - Graphs, applications, games and puzzles, etc.

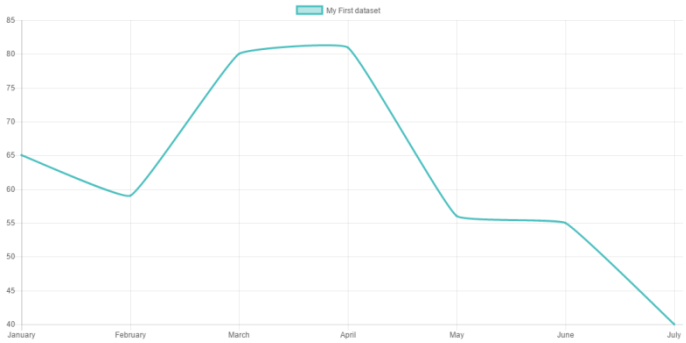
```
<canvas id="canvas" width="150" height="150">
</canvas>

function draw() {
    var canvas = document.getElementById("canvas");
    if (canvas.getContext) {
      var ctx = canvas.getContext("2d");
      ctx.fillStyle = "rgb(200,0,0)";
      ctx.fillRect (10,10,55,50);

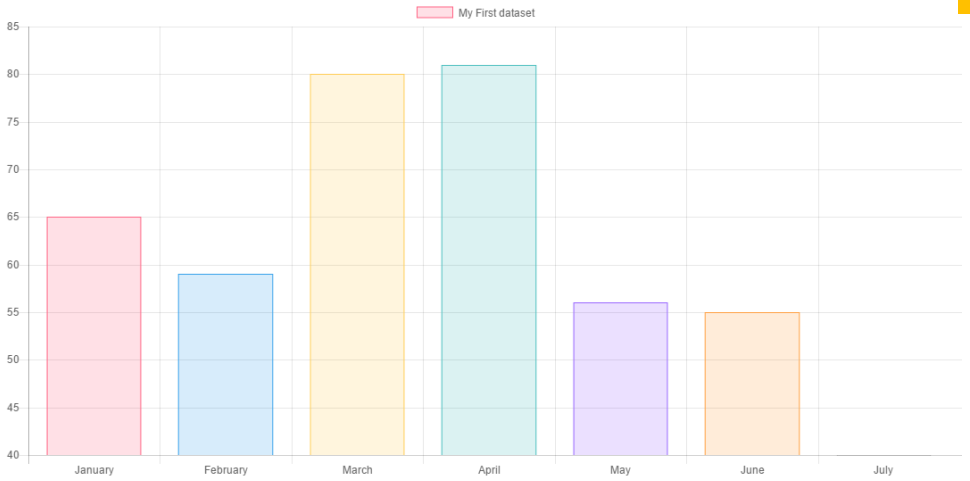
      ctx.fillStyle = "rgb(0,0,200)";
      ctx.fillRect (30,30,55,50);
    }
}
```

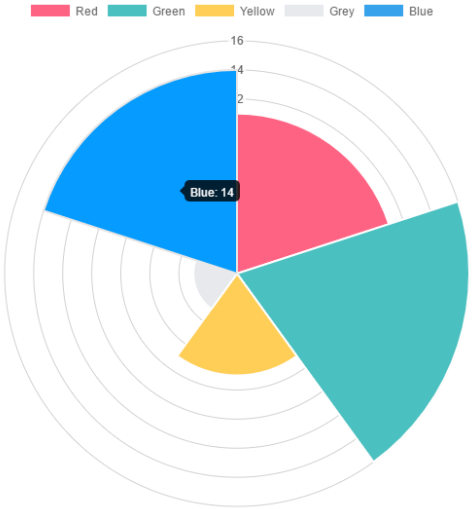


Canvas example: charts drawing

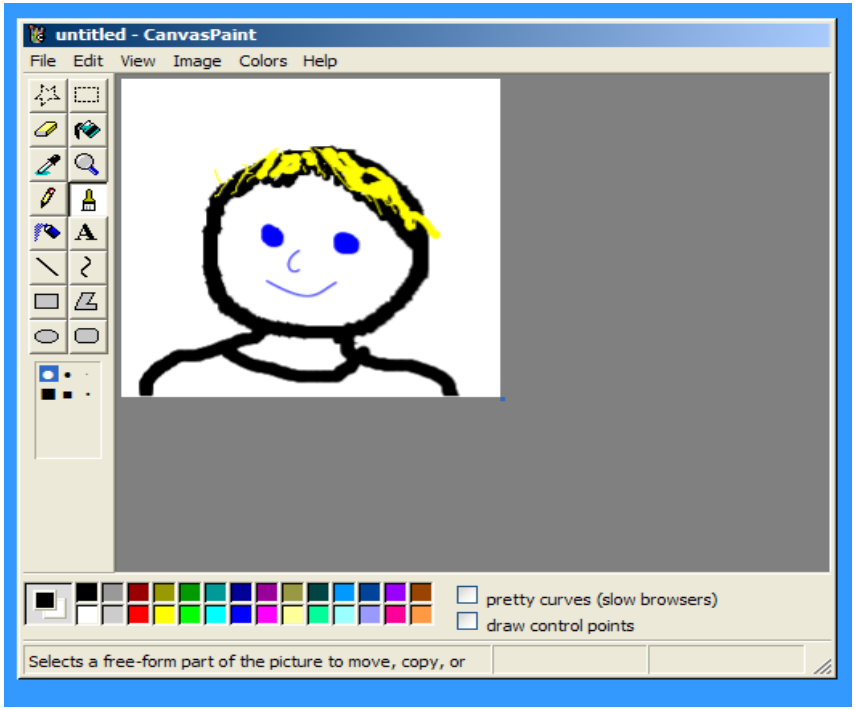


- Chart.js
- http://www.chartjs.org
- A JavaScript library that draws charts using the HTML5 canvas support and data stored in JS arrays.





Canvas example: applications

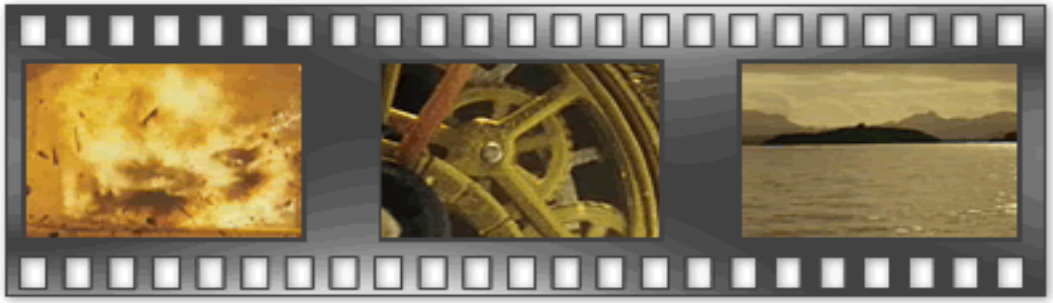


- CanvasPaint
- http://sigilmaster.com/
- Clone of MS Paint built with Canvas

- More canvas demos at:
- https://davidwalsh.name/canvas-demos

HTML5: video and audio

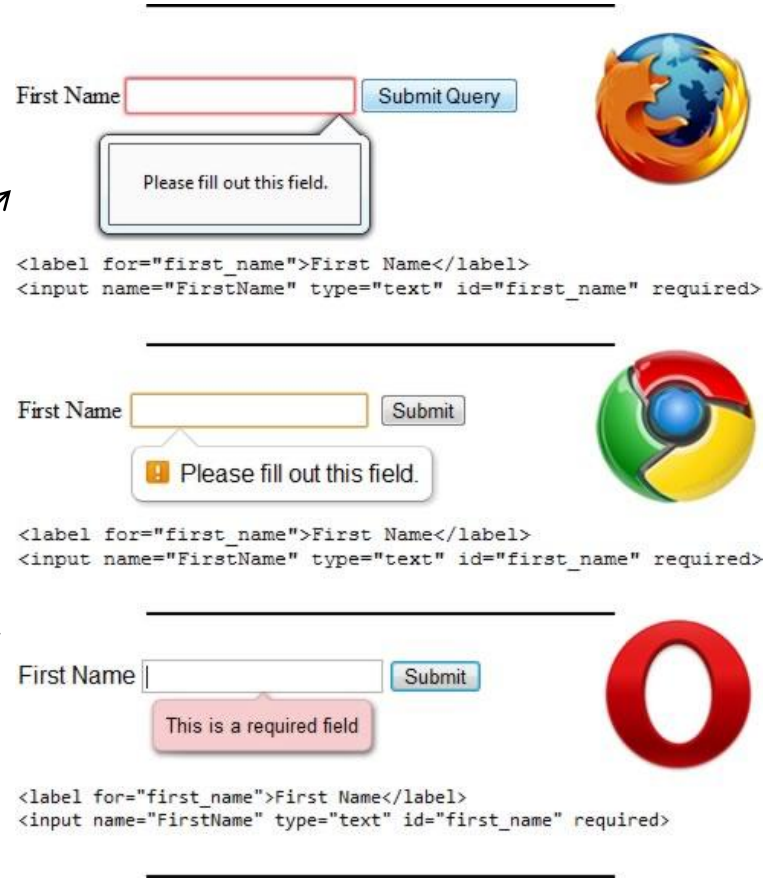
- In the past, many video and audio were handled by plugins (e.g., Flash, QuickTime, etc.)
- HTML5 supports the **<video>** and **<audio> tags**, which can be used just like **<img>** - (any advantages?)



- `<video src="movie.mp4“ id="video">...</video>`
-
```
<button onclick="video.play();">
  Play
</button>
```

HTML5: form handling

- New attributes for simplifying form handling
  - **required** attribute: browser ensures that the data has been filled in
  - **email** input type: a valid email must be entered
  - **url** input type: requires a valid web URL
  - **number** input type: requires a number

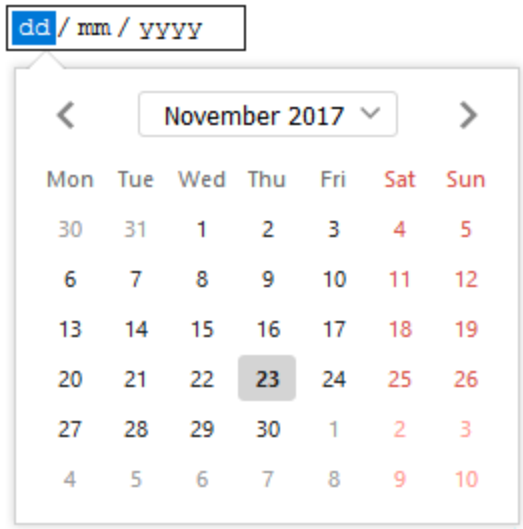


Other form enhancements

- Placeholder text

- Slider element

- Date picker / calendar



HTML5: web applications 1.0

- "Web applications" is an important part of HTML5. The HTML5 standard was initially called "Web applications 1.0"

- Extensive APIs included for developing web applications:
  - Offline applications
  - Web storage

  We will introduce these two; see post-class readings for details of the features below

  - Web Workers (JavaScript running in background)
  - Web Sockets (TCP connections with remote hosts)
  - Interactive UI:
    - Drag and Drop
    - Notifications
    - contenteditable (almost everything in the HTML becomes editable)
  - Geolocation

Offline applications (“app cache”)

index.html:

```
<html manifest="http://m.health.unm.edu/someapp.manifest">
…
</html>
```

```
CACHE MANIFEST
#v1.01

#Explicitly cached files
CACHE:
index.html
Stylesheet.css
fallback.html

#Not cached; available online only
NETWORK:
Search.html
Login.html

FALLBACK:
dynamic.html fallback.html
```

Once this version is changed, the client’s cache would be cleared.

Web storage

- Features:
  - Manipulated by JavaScript
  - No expiry
  - Secure (unlike cookies, which are to be sent back to servers)

- Local storage – “better cookies”
  - 5MB storage per site
- Session storage
  - Lasts as long as the current session is open
  - Each page and tab is a new session
- DB: SQLite or IndexedDB embedded in the browser

Web storage

- Local storage

```
localStorage.lastname="Smith";

document.getElementById("result").innerHTML=localStorage.lastname;
```

- Session storage

```
sessionStorage.useLater('firstname', 'Steven');
alert("Hello " + sessionStorage.firstname);
```

- Database storage

```
var database = openDatabase("Database Name", "Database Version");
database.executeSql("SELECT * FROM test", function(result1) {
  …
});
```

HTML5 roadmap

- First W3C Working Draft in October 2007.
- Candidate Recommendation in 2012.
- HTML5 became a W3C Recommendation in 2014.
  - Version 5.2 became Recommendation in December, 2017
  - Version 5.3 is still being a working draft
- **Popular browsers are supporting HTML 5…**
  - **… to different extents**
  - **Check compatibility before using any uncommon features**
    - **e.g., caniuse.com**

CSS3: new features

- CSS3 is backward-compatible with CSS2, don't have to change current designs.
- New features include:
  - Borders, Rounded, and shadow
  - Backgrounds: new background properties and greater control.
  - Text Effects: text-shadow, word-wrap, etc.
  - Fonts: web fonts.
  - 2D and 3D transforms, e.g., move, scale, turn, spin and stretch elements.
  - More transitions for events such as mouseover or mouseout.
  - Multi-column layout
  - Users can resize elements
- And many other features, some of them are less commonly-used

CSS3 example: round corners

Round corners

This demonstrates the use of rounded corners through CSS 3.

It's childishly simple to make rounded corners now – don't overdo it so! The code to do it is shown below (for different types of browser).

```
border-radius: 10px; -moz-border-radius: 10px; -webkit-border-radius: 10px;
```

Browser-specific variants; not needed for newer browsers

- *border-radius* is used to make rounded corners
- Example:
  `border-radius: 3px`
- The bigger the value of the radius, the more curvy and larger are the rounded corners
- Much simpler than using CSS 2 (background images no longer needed)

CSS3 example: box- and text-shadow

Text and box shadows

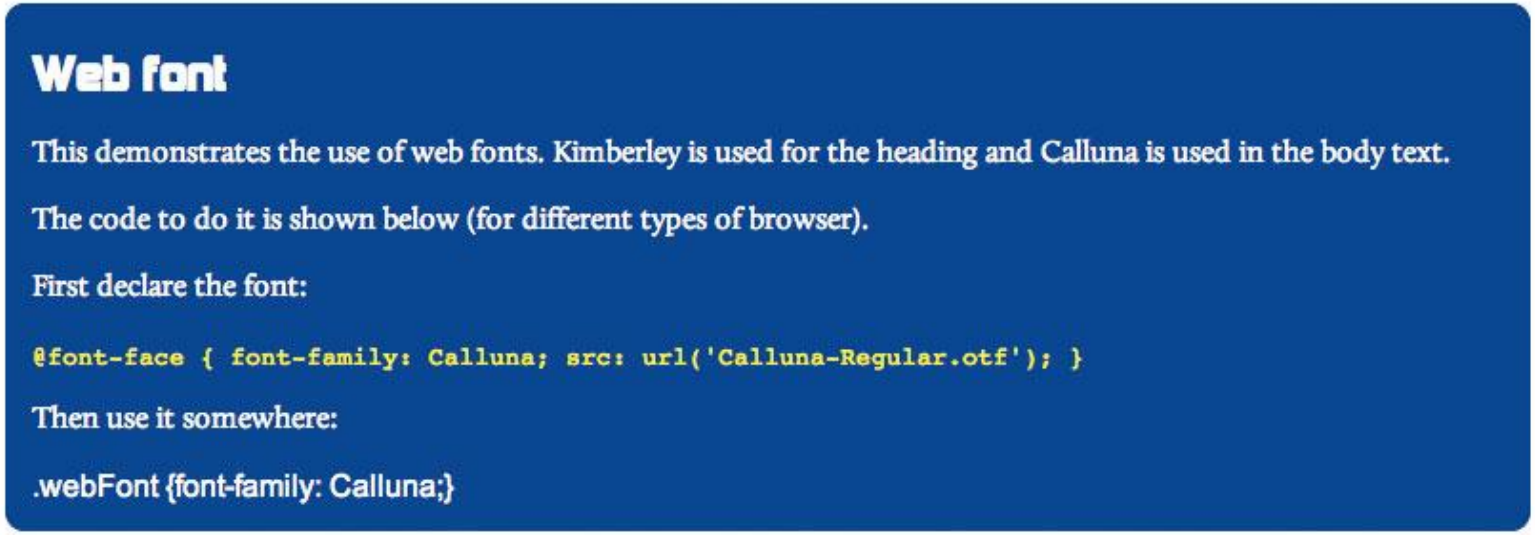
This demonstrates the use of box and text shadows through CSS 3.

The code to do it is shown below (for different types of browser).

```
box-shadow: 5px 5px 2px #666666; -moz-box-shadow: 5px 5px 2px #666666; -webkit-box-shadow: 5px
5px 2px #666666;
.textShadow p {text-shadow: 2px 2px 2px #666666; -moz-text-shadow: 2px 2px 2px #666666; -webkit-
text-shadow: 2px 2px 2px #666666;}
```

- *box-shadow* creates a drop shadow effect (3 lengths and a colour)
- Code example:
  `box-shadow: 10px 6px 5px #888;`
- 10px is horizontal offset, 6px is vertical offset, 5px is ‘blur radius’
- To put the shadow on the left and top, use negative values for the first two offsets
- Higher blur radius = more blurred
- *text-shadow* is similar but applied to text

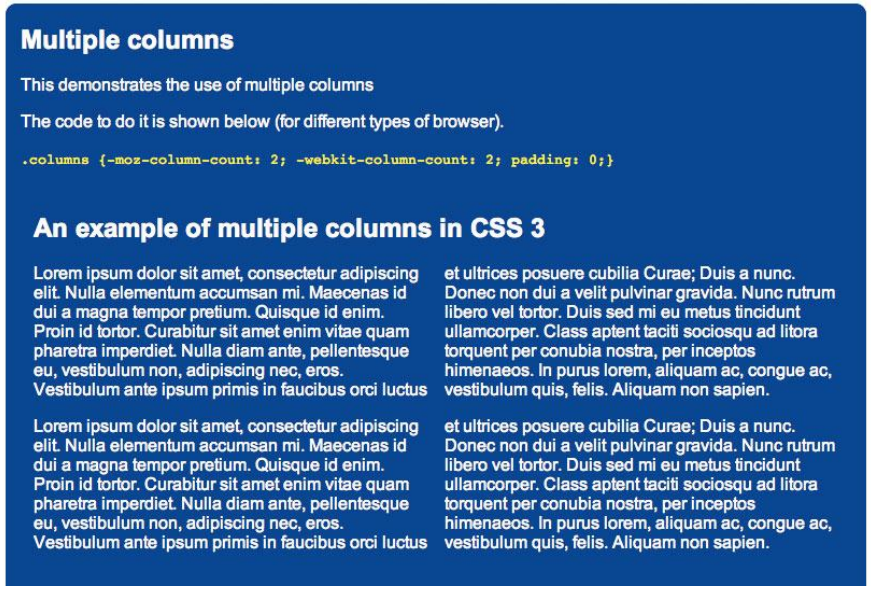
CSS3 example: web fonts



- *@font-face* lets users download TrueType (.ttf) or OpenType (.otf) fonts and use them in the current web page
- First declare the font:
  `@font-face { font-family: Calluna; src: url('Calluna-Regular.otf');}`
- Then use it like a normal font:
  `.webFont {font-family: Calluna;}`

CSS3 example: multi-column layout

- Allows you to split text newspaper-like across multiple columns
- Specify in terms of number of columns or width.
- Example 1:
  `column-width: 45%;`
  `column-gap 5%;`
- Example 2:
  `column-count: 3;`



HTML5/CSS3 implementation strategies

- The latest versions of Safari, Chrome, Edge, Firefox and Opera support many (but not all…) HTML5/CSS3 features.
  - Check **browser compatibilities** (e.g., http://caniuse.com/) before using an uncommon feature
- If you want to implement READ-dominating website (i.e., informational), the support for HTML5 is very mature
  - Make use of new document structures, features for form handling, etc.
- For full-featured web applications (e.g., Google Docs), explore those new applications-related APIs
  - Offline applications, geolocation, canvas, local storage, contenteditable, etc.
  - Again, check browser compatibility before deployment
  - Some interactive components (e.g., canvas) require careful testing across browsers
- Support for HTML5's advanced features (e.g., web workers, web sockets, etc.) is less as complete as that for CSS3, but the mobile world (e.g., web apps packaged as native apps in iOS/Android) is driving the development

Summary

- The web is shaped by standards; standardization bodies include W3C, IETF, WHATWG, etc.
- HTML/XHTML defines the structure of the document
- CSS defines how the document is presented; it separates presentation details from the structure (HTML) of the document
- JavaScript adds client-side logic and changes how X/HTML elements and CSS behaves during run-time

Covered in the last Session (Session 1)

The four low-level, client-side technologies of the web

- JavaScript relies on Document Object Model (DOM) for naming/updating HTML elements
- The login page of Facebook.com as an integrated example of HTML + CSS + JavaScript + DOM
- HTML5 and CSS3 as the ongoing updates

This Session

Separation of {structure (HTML), presentation details (CSS), behaviors (JS/DOM)} is crucial!

Post-class self-learning resources

- For those who wish to explore further; not covered in the exam
- The Modern JavaScript Tutorial - covering both JS (Part 1) and DOM (Part 2)
  - https://javascript.info/
  - For Part 1, you may read up to Section 5 (“Data types”) if the other sections appear difficult to you.
- W3Schools.com's tutorials on:
  - HTML5
  - CSS (you may read only “CSS Tutorial” and “CSS Advanced”)
  - JavaScript
  - DOM
- (Online tutorials on the same topic usually share similar contents; you may skip those duplicated parts.)
- Reference materials:
  - HTML DOM Reference
    - https://www.w3schools.com/jsref/default.asp
  - An e-book on HTML5, CSS3 and JavaScript:
    - M.J. Collins. Pro HTML5 with CSS, JavaScript, and multimedia: complete website development and best practices. Apress. 2017.
    - http://find.lib.hku.hk/record=HKU_IZ51479681950003414

Please see Moodle (under Session 2) for the links

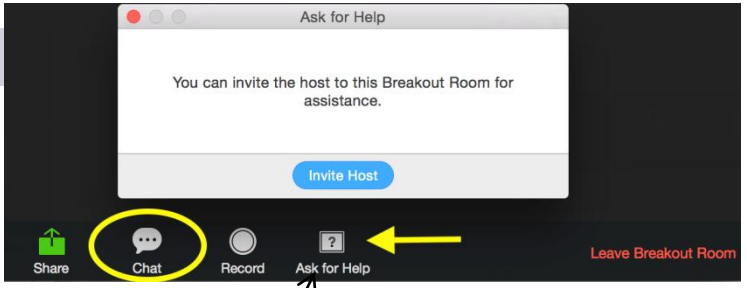
Pre-class reading for Session 3

- The first few sections of the PHP tutorial (i.e., from "PHP Tutorial" to "MySQL Database“; you may skip the section of “PHP OOP”)
  - https://www.w3schools.com/php/default.asp

Lab arrangements in the “hybrid” teaching mode

(updated)

Lab arrangements



- All students (including online students <u>and</u> those in the classroom) will be grouped in Zoom Breakout Rooms
  - Members in a group may discuss, help each other, and finish the lab tasks together
  - Please consider enabling audio (and <u>optionally</u> screen sharing or video if you like) to facilitate the discussion
  - **You may move freely between different breakout rooms** (but we haven't tested this feature yet)
  - Max. # in a room: 4 - please do not join a room that already has 4 members
  - Steven and I will also move around and join your discussion when we have time
- If you have any questions during the lab:
  - Discuss with other group members in your Breakout Room
  - Look at the "model answers" (if that can help)
  - If you attend the lab in-person – easy, simply raise your hand and let us know
    - So, **you are welcome to attend the lab in-person if you need more technical help** <u>and</u> **if the situation allows**
  - If you attend the lab online – use the "Ask for Help" function in Zoom to inform us
    - But we may not be able to appear immediately if there are many students asking for help. So please keep trying (re-clicking the "Ask for help" button) until we join your Breakout Room
    - You may optionally grant us the "remote control" capability if you think that would facilitate us to better help you
    - https://support.zoom.us/hc/en-us/articles/201362673-Requesting-or-giving-remote-control
- If you have any questions (on the lab/assignment/project) after the lab:
  - Post to Moodle's discussion board
  - Email us anytime

} sometimes screenshots may help