HTML Training 101

Just Enough to be Dangerous

Part I • Core Concepts

- Brief History of the Internet
- What Is HTML and Why?
- Structure vs. Appearance
- The WEB is Not PRINT!
- Komputers R Stoopid
- Stuff Not Covered

A Note on Humor: Let's face it, HTML is pretty dry stuff. By keeping this presentation fun, I hope to help you keep awake. Yeah, you, in the back, eyes already droopy. I see you there!

Part II • Specific Details

- Tags
- Formatting a Web Page
- Forms

Part III • Advanced Concepts

- Pitfalls for Government Websites
- Cross-Browser Compatibility
- Coding With Style

Part IV • Q & A

- Questions?
- Comments?
- Concerns?
- Snoring?

Part I • Core Concepts

- Brief History of the Internet
- What Is HTML and Why?
- Structure vs. Appearance
- The WEB is Not PRINT!
- Komputers R Stoopid
- Stuff Not Covered

Brief History of the Internet

- ARPANet–A cold war-era project designed as a decentralized communications network.
- UUCPNet–Set up by Duke University in 1979 to transfer news and messages.
- TCP/IP–Networking the networks.
- "Mosaic" (the first web browser) debuts in 1993, bringing "the web" to the world.

What Is HTML and Why?

- HTML stands for “hypertext markup language.”
- “Hypertext” = links! Or any text that references something else.
- “Markup” = tags! This tells the browser what it’s looking at. (More on that later.)
- 100% text. That makes it lean, mean, and low-bandwidth friendly!

Structure vs. Appearance

- "Clean" HTML is all about the page structure, not what it looks like. Is something a header? A paragraph? A list? A link?
- The appearance of an element is its style. Is it **bold**? Is it yellow? Is it underlined?
- The difference is important. A blind person can't tell if text is yellow.

The WEB Is Not PRINT!

- People accessing your web page might be:
  - Using a Web browser
  - Using an ooooooold Web browser!
  - Using a screen reader
  - Using a cellphone or PDA
  - Unable to use a mouse
  - In another country
- There's no way to be sure what they'll "see," so good structure is **vital**.

Komputers R Stoopid

- They do what you told them to, not what you wanted them to.
- They never guess — if they don't know what to do, they pitch an error.
- If they could guess, they'd guess wrong every time anyway.
- This is why getting commands right (and without typos) is important.

Stuff Not Covered

- Hosting/Serving Pages
- Scripts, Web Programming
- Database Integration

LINGO CHECK

PAGE: This is a file (or several files) you view in a web browser. It may be static (i.e., pregenerated) or dynamic (i.e., generated at the time it's served).

SITE: This is one or several pages under a single domain, usually with a unified purpose.

SERVER/CLIENT: A computer that provides information to another computer is a server. The computer that receives the information is the client. A host is an entity (such as a company) that houses servers.

Part II • Specific Details

- Tags
- Forms
- Formatting a Web Page

Tags • "Speaking HTML"

- Tags put the "markup" in hypertext markup language.
- They look like this:

```
<tag>stuff inside the tag</tag>
```

- Everything in HTML is done with tags!

Types of Tags

- "Opening" tags tell the browser "start applying the tag here." They are in brackets, such as `<tag>`
- "Closing" tags tell the browser "stop applying the tag here." They are indicated with a slash, such as `</tag>`
- "Self-closing tags" open and close themselves in the same tag. They are indicated by a slash at the end, such as `<tag />`

Some Real HTML

- Even without knowing any HTML, you can start to see how it works here:

```
<p>          <!-- opening tag! -->
  This is a sample of
  real HTML!<br /> <!-- self-closing tag! -->
  <strong>Isn't that exciting?</strong>
</p> <!-- closing tag! -->
```

Attributes

- Many tags have attributes, which define or alter the tag's behavior.
- These are in the opening tag, and usually have a format like this:

```
attribute = "parameter"
```

- Attributes have a default value; if the attribute is not specified, it will use the default.

“Name” vs. “ID”

- “Name” and “ID” are both very commonly-used attributes. For example, the field to input a person’s first name on a registration form might be named “first_name”.

```
<input type=“text” name=“first_name” />
```

“Name” vs. “ID” (cont’d)

- So what’s the difference?
- Many tags can have the same name, but each ID is unique.*
- A page’s styles are often linked to an element’s ID, but cannot be linked to the element’s name.

*In theory, at least. Web pages will work if you give different things the same ID, and sometimes Javascript depends on that. But it’s “technically” wrong.

Referencing an ID

- The neat thing about IDs is that you can use them to link to a specific part of a page.
- For instance, if you have a section with the ID "maincontent," you can link directly to it with the URL **`mypage.htm#maincontent`**.
- Besides being a handy shortcut, this becomes very important for accessibility, later.

Attributes vs. Styles

- Many tags have visual attributes. For instance, the image tag (`<img>`) has a "border" attribute.
- "But wait!" you say. "I thought structure and appearance should be separated!"
- You're absolutely right. That's why the "border" attribute is deprecated.

Attributes vs. Styles (cont'd)

- "Deprecated" elements are those which the W3C* has said should no longer be used, although most browsers still understand them.
- In general, if a tag has a visual attribute (such as "color," "border," "height," etc.), you should not use that attribute, but put that information into the tag's style instead. (We'll talk about that later.)

*W3C = "World Wide Web Consortium." These are the guys who set the standards that web programmers should use. We'll hear more from them later, too.

BLOCK vs. -inline-

- There are two other ways of describing tags, “block” or “inline.”
- Block-level tags indicate a piece of structure, one of the “blocks” that makes up the page, so to speak.
- Inline tags refer to items within a block, but do not in themselves define a block.

BLOCK vs. -inline- (cont'd)

- Some block-level tags…
  - Body `<body>`
  - Paragraph `<p>`
  - Header `<h1>`
  - Table `<table>`
  - Ordered List `<ol>`
  - Block quote `<blockquote>`
- Some inline tags…
  - Anchor `<a>`
  - Image `<img>`
  - Strong `<strong>`
  - Emphasis `<em>`
  - Line-break `<br />`
  - Quote `<q>`

Some Key Tags We'll Discuss

- **HTML, Head, Body** — The Skeleton of Your Page
- **DIV** — Creating Your Work Space
- **BR** vs. **P** — New Line or New Paragraph?
- **H1** and Its Children
- **A HREF** and Its Cousins
- **IMG** — Purdy Pitchers!
- **STRONG** and **EM** — Bold? Italics? What Are Those?
- **UL, OL, LI**, and Other Musical-Sounding Tags
- **SPAN** — When Nothing Else Works!

<html>, <head>, <body>

- These tags tell a browser that it's looking at a web page.

```
<!DOCTYPE><!-- we'll get back to this one -->
<html> <!-- tells the browser "This is an HTML
                document!" -->

        <head>
              <!-- contains important,
                non-displaying information -->
        </head>

        <body>
              <!-- this is what people see -->
        </body>

</html>
```

It's All in Your `<head>`

- Provides information about your page, such as what it's called and what it's about, to the browser, search engines, or other web pages.
- The only tags legal inside the page header are: `<base>`, `<link>`, `<meta>`, `<title>`, `<style>`, and `<script>`.
- Well-crafted meta-tags and other header info are important for search engine optimization (SEO) and other uses.

A Healthy `<body>`

- This is where the stuff people are supposed to see goes. This is the "page" part of the web page.
- The contents of the page are contained in nested tags, such as `<p>` for "paragraph."
- The page body can contain almost any HMTL tag, except for the ones that go into the document header!

`<div>` • Creating Your Workspace

- `<div>` (short for "division") only does one thing: create a block-level container.
- Each div is identified by an "id" attribute, so you can tell one from another.
- You can have divs inside other divs, paragraphs in divs, and so on.
- You apply styles to a div to determine its size, formatting, placement, etc.

• Creating Your Workspace

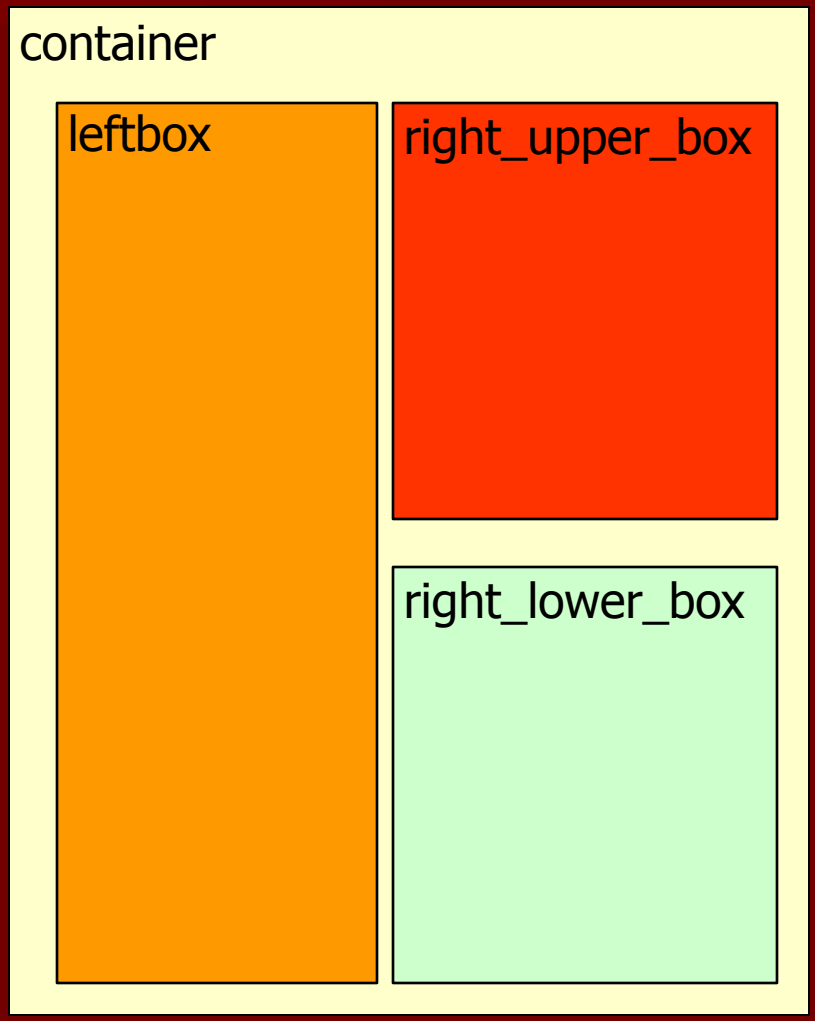
```
<div id="container">

   <div id="leftbox">
        <!-- stuff -->
   </div>

   <div id="right_upper_box">
        <!-- stuff -->
   </div>

   <div id="right_lower_box">
        <!-- stuff -->
   </div>

</div>
```



 vs. <p>

- HTML ignores the return key – and blank spaces. It only "sees" code!
-
 is a line break; <p> is a paragraph. They aren't the same!
- <p> is a structure, and should be used for almost all situations.
- When would be a good time to use <br /> instead of <p>?

Bonus question: why <br /> and not
?

 vs. <p>

```
<p id="circleaddress">
  Circle Solutions, Inc.<br />
  8280 Greensboro Drive<br />
  Suite 300<br />
  McLean, VA 22102<br />
  <a href="http://www.circlesolutions.com">
     www.circlesolutions.com
  </a>
</p>
```

`<h1>` and Its Children

- Meet the "header family!"
- `<h1>`, `<h2>`, `<h3>` … `<h5>`
- Like `<p>`, a header is a structure. The fact that its default formatting changes is just a handy shortcut.
- Some people do this: `<p class="header">`. Throw rocks at them!
- If it's a header, it should be in a header tag!

<h1> and Its Children

```
<body>

   <h1>HTML Training 101</h1>
   <h1 class="subtitle">Just Enough to be
   Dangerous</h1>

   <p>Hello and welcome blah blah blah…</p>

   <h2>Part I: Core Concepts</h2>
   <p>The internet started with blah blah blah…
   </p>

</body>
```

`<a href="">` and Its Cousins

- `<a>` = "anchor" – it's just a handy (inline) tag to hang things on.
- `<a>` tags can have lots of attributes, but 90% of the time, they use `href` (i.e., "hypertext reference") to create links.
- `href` – the ability of pages to link to other pages – is what makes the web so useful!

`<a href=“”>` and Its Cousins

- To create a link:

```
<a href="{URL you are linking to}">text</a>
<a href="http://www.google.com">Search Google!</a>
```

- Most URLs start with http:// or https:// but other types are possible (such as mailto: or ftp://).

LINGO CHECK

URL: “Uniform resource locator.” Basically, an item’s address on the web. (Remember, engineers came up with these terms.)

HTTP: “Hypertext transfer protocol.” The language computers use to call and send HTML files.

HTTPS: Just like HTTP, except with added security (such as encryption).

`<img>` • Show Me Something!

- `<img>`, as you might guess, stands for "image"
- The web generally supports .gif, .jpg, and .png image formats. Individual browsers may support more.
- The web displays images at 72 dpi ("dots per inch") – but your screen resolution determines how big an "inch" is.

Image Formats

- **.gif** stands for "graphics interchange format" and was created by CompuServe. They pronounce it "jif," like the peanut butter. But they're wrong. ;-P
- **.jpg** stands for Joint Photographic Experts Group (naturally).
- **.png** stands for "portable network graphics." Whee.

Image Formats (cont'd)

- **.gif** images are best for clean, simple line-art, such as technical illustrations or corporate logos. They can have transparent sections, but have limited colors.
- **.jpg** images are best for color-rich images such as photographs, paintings, or gradients. They have a wider range of color, but cannot have transparencies.

Image Formats (cont'd)

- **.png** images were created to have the best of both worlds (and to get around certain copyrights) … but came in kinda late.
- **.png** images have lots of advanced capabilities, but are still only used by a relatively small number of people.

`<strong>` and `<em>`

- `<strong>` indicates an important item, and is usually rendered in **bold type**.
- There is a `<bold>` tag, but it is deprecated. Why is this?
- `<em>` indicates an item has emphasis, and is usually rendered in italics.
- There is also an `<i>` tag, but it is also deprecated. Same reason.

, <ol>,

- **<ul>**, **<ol>**, and **<li>** are used to make lists – like this slide!
- **<ul>** makes "unordered lists" – i.e., bulleted lists.
- **<ol>** makes "ordered lists" – i.e., numbered or outline-style lists.
- **<li>** is wrapped around each item of a list (of either type).

Lists In Action • Unordered

```
<ul id="unordered_list_sample"
   style="list-style-type: square">

   <li>First bullet
       <ul>
          <li>First sub-bullet</li>
       </ul>
   </li>
   <!-- notice that the
       sub-bullet is inside -->

   <li>Second bullet</li>

   <li>Etc.</li>

</ul>
```

- First bullet
  - First sub-bullet
- Second bullet
- Etc.

Lists In Action • Ordered 1

```
<ol id="ordered_list_sample"
   style="list-style-type: decimal">

   <li>First bullet
       <ul>
          <li>First sub-bullet</li>
       </ul>
   </li>
   <!-- notice that the
       sub-bullet is inside -->

   <li>Second bullet</li>

   <li>Etc.</li>

</ol>
```

1. First bullet
    1. First sub-bullet
2. Second bullet
3. Etc.

Lists In Action • Ordered B (?!)

```
<ol id="ordered_list_sample"
   style="list-style-type: upper-
   alpha">

   <li>First bullet
       <ul>
          <li>First sub-bullet</li>
       </ul>
   </li>
   <!-- notice that the
       sub-bullet is inside -->

   <li>Second bullet</li>

   <li>Etc.</li>
```

A. First bullet
   A. First sub-bullet
B. Second bullet
C. Etc.

• When Nothing Else Works!

- **** is used to define an inline section within a text block, and is most commonly used to apply style information (font change, color, etc.).
- It is a structural element, but its primary use is formatting.

```
<p>Is anything prettier than the color <span
    class="bluetext">blue</span>?</p>
```

Is anything prettier than the color blue?

What's So Special About Special Characters?

- The WEB is not PRINT! Remember?
- The web doesn't normally have curly quotes, em-dashes, en-dashes, ©, etc., because not all fonts contain these characters.
- With the marketization of the web, however, some people have demanded them. What to do?

What's So Special About Special Characters?

- HTML has special codes that tell the browser to render these characters, font or not.
- The code usually looks like:

```
&{description};
```

- For instance: `&mdash;` draws "—", `&ntilde;` draws "ñ", and so on.

Some Common Special Characters

- `&mdash;` (—)
- `&ndash;` (–)
- `&ldquo;` (“)
- `&rdquo;` (”)
- `&lsquo;` (‘)
- `&rsquo;` (’)
- `&hellip;` (…)
- `&Ntilde;` (Ñ)*
- `&ntilde;` (ñ)*
- `&aacute;` (á)
- `&eacute;` (é)
- `&iacute;` (í)
- `&oacute;` (ó)
- `&uacute;` (ú)

*Yup, it’s case-sensitive! Darn tricky, those HTML elves.

Forms • The Basics

- Forms are used to modify the behavior of pages – but, by themselves, they don't do anything!
- HTML is "static," i.e., what is sent to the server is what it shows. For the page to be "dynamic," it requires some variety of scripting, such as Javascript, PHP, ColdFusion, et al.
- Therefore, the coverage of forms in this course is very limited, as we're concerned purely with HTML.

`<form>` Tag, `action=` Attribute

- Forms require at least two pieces, the form page itself, and the "landing" page (a.k.a. target page, processing page, etc.). These can be the same page – you can send a form back to itself.
- The form page has a `<form>` tag on it. Inside the form tag is an `action=` attribute, which tells the form the name of the landing page to go to for processing.
- All form elements require a `name=` attribute! Otherwise the landing page won't be able to **see** the data you're sending.

A Very (Very) Simple Form

```
<form action="landingpage.cfm" id="userinfo">

   <p>First Name:
       <input type="text" name="first_name"
       size="25" maxlength="25" /></p>

   <p>Last Name:
       <input type="text" name="last_name"
       size="25" maxlength="25" /></p>

   <input type="submit" name="submit"
       value="Go!" />

</form>
```

Can I Get Your `<input>`?

- **`<input>`** is where you enter data, which is then sent to the landing page for processing.
- By itself, it doesn't do anything. It needs a **`type=`** attribute to be useful!
- Valid **`type=`** attributes include text, checkbox, radio, button, password, reset, submit, and others.

<select> an <option>

- **<select>** creates a drop-down list, popular for constraining input values.
- **<option>** tags populate the otherwise-empty **<select>** list.
- The structure is somewhat similar to the lists we saw earlier.

```
<select name="daysoftheweek">
    <option value="1">
        Monday</option>
    <option value="2">
        Tuesday</option>
    <option value="3">
        Wednesday</option>
    <option value="4">
        Thursday</option>
    <option value="5">
        Friday</option>
    <option value="6">
        Saturday</option>
    <option value="7">
        Sunday</option>
</select>
```

DANGER, WILL ROBINSON! **<select>** statements are very hackable. Don't depend on them to secure a form!

`<textarea />`: For People With a Lot to Say

- **`<textarea>`** is similar to **`<input type="text">`** except that it creates a large field for people to type in (such as a "notes" or "comments" field).
- A **`<textarea>`** field's size is determined by a number of rows and columns, set as attributes.
- It is its own tag, rather than being a type of **`<input>`**. Why? Who knows? HTML is just sometimes weird that way. Blame it on design by committee.

```
<textarea rows="5" cols="60" name="comments_field" />
```

`submit` to My Will! Muahahahaaa!

- `<input type="submit">` creates the "submit" button, which tells the form to go to the landing page.
- There are other types of submit inputs, including `type="image"` … but the good old-fashioned submit button is usually your best bet.
- To have the button labeled something other than "Submit," give it a `value=` attribute.

Forms vs. URL Parameters

- Sometimes, instead of using a form, info will be sent to a page via a URL parameter. For instance: `http://www.fakeurl.com/page.htm?fontsize=small` tells the server "send me ***page.htm*** and tell it that I want the value of 'fontsize' to be 'small'."
- URL parameters are very limited, and very, very not secure!

Forms vs. URL Parameters (cont'd)

- Because URL parameters can easily be seen, somebody who wants to muck around with the page could send anything they wanted.

```
http://www.fakeurl.com/page.htm?fontsize=crashpage
```

- This is a common way for hackers to send malicious code or otherwise attempt to break stuff.

Forms vs. URL Parameters (cont'd)

- This is not to say that URL parameters are useless, or we wouldn't still have them, obviously. Just that they're limited.
- You should never send unencrypted critical data (say, credit card numbers…) via URL parameters, and any code on the landing page that uses the URL parameter needs to be carefully secured.
- They're best used for simple navigational stuff – e.g., "go to this page, show this image, make the font larger or smaller," that kind of thing. They can also be a great way to send "canned searches" to a search engine.

Formatting a Web Page

- Pure HTML pages work very well, but are dull. Most users prefer things to be shiny.
- So we add formatting! But formatting can be a trap.
- You mustn't confuse *format* with **CONTENT**.

Read a Page, Or Listen to It? (Why Structure Is King)

- Remember our mantra, the **WEB** is not **PRINT**!
- Just like books are printed in Braille for the blind, there are web browsers that "read" web pages aloud. And if you've put in a bunch of *pretty foofery* that the web reader thinks is **CONTENT**, the blind web user is going to hate your guts.

I Love `blockquote` – It Gives Me Bigger Margins! =^.^=

- Browsers add default formatting to various tags, e.g., **`<h1>`** tends to be large and boldface … or **`<blockquote>`** tends to have indented margins.
- In "the old days" (i.e., 1996), people would use these to format the page, and you still occasionally run into that.

I Love `blockquote` – It Gives Me Bigger Margins! =^.^=

- So what happens when a web-reader finds a chunk of text in a **<h2>** tag when all the coder really wanted was to make it large and bold?
- Or a **<blockquote>** section that isn't really a block quotation?

Tables: Good or Evil?

- "I don't care if the web is not print, I want my webpage to look like a brochure!" – the single largest cause of web developer mental breakdowns.
- Tables were once the closest thing web developers had to "print-like" control of webpage layouts.
- And then a humanity-hating vampire guy named David Siegel wrote Creating Killer Websites and people just went nuts with it.

Tables: Good or Evil?

- People used tables to create columns! Lists! Pages that looked like brochures! Addresses formatted so that the phone and fax numbers had the same tabular indent!
- And the poor people trying to view these monstrosities of web design with web-readers hated life.

Tables: Good or Evil?

- People who can't use a mouse and so move their cursor from item to item with a TAB key often get unpredictable or downright strange results with tables.
- Navigating inside tabular formatting with a web-reader is a nightmare of useless information and "empty" cells that only contain images or blank spaces to make things fall into the right place.

Tables: Good or Evil?

- So what are they actually good for?
- Well … tabular content. Seriously. For that, tables rock.

This Year	Last Year	Difference
591	222	369
-33	67	100
Fish	Frogs	Legs

Building a Proper Table

```
<table id="years_difference">
    <tr>
        <th id="this_year">This Year</th>
        <th id="last_year">Last Year</th>
        <th id="difference">Difference</th>
    </tr>
    <tr>
        <td headers="this_year">591</td>
        <td headers="last_year">222</td>
        <td headers="difference">369</td>
    </tr>
    <tr>
        <td headers="this_year">-33</td>
        <td headers="last_year">67</td>
        <td headers="difference">100</td>
    </tr>
    <tr>
        <td headers="this_year">Fish</td>
        <td headers="last_year">Frogs</td>
        <td headers="difference">Legs</td>
    </tr>
</table>
```

<th> creates a "table header," telling the page that this is the label for content in the table. Each header's ID must be unique.

<td> creates "table data." The "headers" attribute indicates which **<th>** blocks apply to this piece of data. You can have one, none, or lots of headers for each cell.

Cascading Style Sheets (CSS)

- CSS is the proper way to style your web pages, separating content from presentation.
- Of course, CSS isn't perfect. That would be too easy.

LINGO CHECK

CSS: "Cascading Style Sheets." This is simply a collection of styles that define how your page will render, be it on a screen, from a printer, etc.

Cascading: CSS is considered to be "cascading" because styles can be "inherited" from other styles, following set rules of priority. For example, if "H1" is defined as bold type, then "H1.leftmenu" would "inherit" the bold type property unless you specify otherwise.

CSS Is a BIG Topic

- Unfortunately, it's too big a topic for this course, but here are some basic concepts:
- Any HTML tag can have styles applied, but not every style applies to every tag.
- Styles can be applied in a linked file, in the header of your HTML file, or even within an individual tag.

More CSS Core Concepts

- The best way to apply CSS usually by defining classes for each tag. For instance, you might have:

`<P>`	Your basic paragraph tag.
`<P class="leftmenu">`	A paragraph in your left menu, probably a smaller font or possibly a different color.
`<P class="pullquote">`	A paragraph that highlights some text element in your current story, a large font with a shaded background and colored border.

Learning CSS

- A good place to start is http://www.w3schools.com/css/
- CSS is not hard, but it can be convoluted and often behaves in unexpected ways. Don’t fret if it takes a while to pick it up.

Part III • Advanced Concepts

- Pitfalls for Government Websites
- Cross-Browser Compatibility
- Coding With Style

Security and Privacy: A Vitally Important Pain in the Neck

- Even if Circle isn't exactly MI6, we still sometimes work with potentially-sensitive information and have to be aware of security issues.
- Even something as relatively benign as a conference "list of attendees" is a potential minefield. Imagine, for instance, thieves planning to rob someone's house because they looked at our website and found out the person wouldn't be home during the conference.

Cookies, and Why We Don't Use Them

- Websites can save and retrieve information from your computer by means of a "cookie" – which is very handy.
- Unfortunately, cookies can also store malicious code, or share information with people you'd rather they didn't.

LINGO CHECK

Cookie: An internet cookie is simply a file written by the web browser on your computer, which records information about a given website (such as your login information, items you've looked at on previous occasions, etc.). When used right, they are not only harmless, but beneficial – without them web sites have no "persistence."

Unfortunately, they aren't always used right.

Cookies, and Why We Don't Use Them

- Cookies have gained something of a bad rap – while they can be problematic, 99% of them are harmless and downright convenient.
- But, because politicians like simple, one-size-fits-all answers, the government policy more or less boils down to, "Cookies are bad. Don't use them."

Cookies, and Why We Don't Use Them

- One of the implications of this: government web sites are not "persistent" – i.e., when you leave a website, that website completely forgets you.
- Our clients, as a rule, hate this and want to get around it. But we are prevented by law from getting around it. So when our clients say, "Can't we just do it anyway?" our answer must be, "Sorry, but no."
- There are rumblings that the government may revisit its policy on cookies sometime soon – but they haven't yet!

Personally Identifiable Information ("Mmm, PII…")

- PII is anything that connects a specific user with a specific activity – whether it's a page they've visited, a document they've downloaded, or anything.
- We're not allowed to track or save PII except in very specific and limited ways. For instance, we can't ship you a booklet on depression if we don't know your address.

Keep Your Fingers Outa My PII

- Many people are understandably wary of having the government keep track of physical or mental health issues they may have wanted to research – and they especially don't want that information to be tracked across multiple websites.

What Ingredients Make PII?

- Name
- Address
- Phone Number
- E-mail Address
- Computer IP number (sometimes)
- Login ID's
- Passwords
- Social Security Number
- Credit Card Information
- Account Numbers
- Did we mention login ID's and passwords? That's an important one.

Section 508
(Or, Why Your Site is Dull)

- "Section 508" or "508 Compliance" basically boils down to this: our websites are for everybody to use – not just people who can see, easily click around with a mouse, and have Javascript enabled on their computer.
- This means we have to design sites that are easy to navigate, are screen-reader friendly, and do not require Javascript to work.

YES, Even for Client-Only Pages!

- Sometimes clients will ask us to ignore 508 restrictions for "internal only" utilities, such as project tracking. Sorry, it still ain't kosher.
- Legal issues aside, it sets up a discriminatory work atmosphere. Is a blind person going to be turned down for a job because they can't use an "internal only" non-compliant website?

The 16 Rules of 508

- Every non-text element (e.g., an image) must have a text equivalent (e.g., an alt attribute).
- Every multimedia presentation (e.g., a Flash movie) must have an equivalent alternative.
- Any information conveyed with color must also have a "non-color" indication (such as heavier type, underlining).
- The page must be readable without a stylesheet!
- You must have text links as well as any server-side image map links.
- If an image map requires a custom shape, it must be a "client-side" map.
- Data tables must have row and column headers.
- Data cells in a table must be associated with the appropriate headers.

The 16 Rules of 508 (cont'd)

- Don't use frames.*
- Don't make the screen flicker at low refresh rates – i.e., don't put in blinky or scrolling text.
- If there's no other way to comply with 508, make a text-only website.
- Don't code the page so that it requires scripting (e.g., Javascript) to work.
- If your page needs external software (such as PDF readers), you must provide a link to that software.
- Forms must be built in a way that enables "assistive technology" (e.g., screen readers) to work easily.
- Make it easy to skip navigation links.
- If a timed response is required, alert the user and give them time to ask for more time.

*Okay, that's not the official rule, but that's the easiest way to go. Frames are just bad, don't use them. WHAT was the W3C thinking with those???

Browsers (or, "It Looked Fine in Firefox!")

- In the early days of the web, Netscape and Microsoft Internet Explorer (a.k.a. IE), were battling it out for who would conquer the web, so they each piled on competing "features" that they tried to convince W3C to make standard.
- Netscape is history and IE is still here … but did it win the Browser Wars?
- Netscape's legacy lives on in Mozilla Firefox (and others).

W3C, Standards, and Other Things Microsoft Ignores

- The World Wide Web Consortium (a.k.a. "W3C") sets the standards for HTML, CSS, and other web-based technologies, off in an ivory tower somewhere.
- But, as the U.N. of the world wide web, they have exactly as much authority, i.e., none at all.

W3C, Standards, and Other Things Microsoft Ignores

- Mozilla (the makers of Firefox) and most other browser manufacturers do their best to comply with the W3C guidelines, but Microsoft, as always, is a law to itself.
- This has led to two things:
  - Web sites that work only in IE (or not at all in IE), or
  - Web designers jumping through hoops to make their pages work both in Firefox and IE

Why Firefox? Firefox is the most popular of the more-compliant browsers. If it works in Firefox, it'll probably work in the rest.

`!DOCTYPE`, The Quicker Browser-Fixer

- The very first line in your HTML code should be a declaration of what kind of document the browser is looking at, which is `!DOCTYPE`.
- `!DOCTYPE` defines what release of HTML (or other web language) the page is coded to, so the browser knows what tags it can expect to see and how to handle them.

`!DOCTYPE`, The Quicker Browser-Fixer

- Among other things, this tells more recent versions of IE: "Render to standards!"
- If `!DOCTYPE` is absent, IE will just go do its own thing and who knows what you'll get?
- Sadly, even the magic of `!DOCTYPE` can't completely tame IE. There are still plenty of ways IE will break pages even if you have a `!DOCTYPE` tag.

!DOCTYPE, The Quicker Browser-Fixer

- Circle currently codes to the standard of "Transitional XHTML 1.0", so as the first line of our HTML pages we put:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

- You're right, it does look like gobbledygook. But the browsers know what it means, that's what's important.
- As standards change, the !DOCTYPE tag we use will change with it. The W3C is looking to do away with "XHTML" and move to "HTML 5.0" … eventually.

XHTML? HTML? XML?

- These are all variations on a theme. Remember that HTML meant "hypertext markup language?"
- XML = "extensible markup language" – this is a language where you can define your own tags. Therefore…
- XHTML = "extensible hypertext markup language" – that is, a hypertext markup language where you can define your own tags. The only problem is, it didn't fly.

XHTML vs. HTML 5+

- The W3C decided that HTML had run its course and that the world should move to a new language, XHTML, that includes HTML.
- But the geek community just shrugged and said, "Meh. We're sticking with HTML, it does the job. Keep your XHTML, we don't need it."
- So after years of wrangling, the W3C has given up on XHTML. But until HTML 5 actually becomes the standard, XHTML 1.0 is what we're using.

Coding With Style • `META` Tags

- **`META`** tags go into the header of your webpage and convey lots of useful "non-rendered" stuff, especially for search engines – although they're nowhere near as important as they once were.

The decline and fall of the META tag: Marketing people, ever eager to abuse a resource, starting putting all sorts of junk into their META tags in order to "optimize their search engine rankings" – i.e., rate higher on any given search than they really deserved to be. This naturally led to search engines being less and less useful because they depended on META tags having accurate information. ("Why is my search for motorbikes coming up with results about Viagra?") To fight back, the search engines, starting with Google, came up with their own ways of indexing pages that didn't depend on META information. This is one reason Google is leader of the pack.

META Tags are Down, But Not Out!

- META tags are purely optional. Your page will work just fine without them. But…
- This isn't to say that META tags are useless! Many search engines (including Google) won't "take your page seriously" unless it has valid, well-constructed META tags.
- The most important of the META tags are "description" and "keywords."

Description, Keywords

- Guess the website!

```
<meta name="description" content="Online shopping from the earth's
   biggest selection of books, magazines, music, DVDs, videos,
   electronics, computers, software, apparel &amp;amp; accessories,
   shoes, jewelry, tools &amp;amp; hardware, housewares, furniture,
   sporting goods, beauty &amp;amp; personal care, broadband &amp;amp;
   dsl, gourmet food &amp;amp; just about anything else." />

<meta name="keywords" content="Amazon, Amazon.com, Books, Online
   Shopping, Book Store, Magazine, Subscription, Music, CDs, DVDs,
   Videos, Electronics, Video Games, Computers, Cell Phones, Toys,
   Games, Apparel, Accessories, Shoes, Jewelry, Watches, Office
   Products, Sports &amp;amp; Outdoors, Sporting Goods, Baby Products,
   Health, Personal Care, Beauty, Home, Garden, Bed &amp;amp; Bath,
   Furniture, Tools, Hardware, Vacuums, Outdoor Living, Automotive
   Parts, Pet Supplies, Broadband, DSL" />
```

Coding With Style • Indent Your Code

- In all of our examples, the code has been indented to make it easier to read.
- Remember that HTML ignores returns and blank spaces … so you can toss in as many of those as you like. White space costs no bandwidth and is easy on the eyes!
- Typically, each level of "nesting" gets an additional indent.

Coding With Style • Indent Your Code

- Which is easier to read?

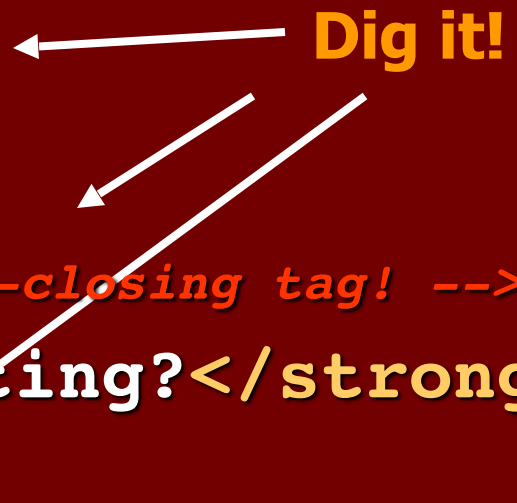
```
<table id="years_difference"><tr><th
id="this_year">This Year</th><th
id="last_year">Last Year</th><th
id="difference">This Year</th></tr><tr><td
headers="this_year">591</td><td
headers="last_year">222</td><td
headers="difference">369</td></tr><tr><td
headers="this_year">-33</td><td
headers="last_year">67</td><td
headers="difference">100</td></tr><tr><td
headers="this_year">Fish</td><td
headers="last_year">Frogs</td><td
headers="difference">Legs</td></tr></table>
```

```
<table id="years_difference">
  <tr>
    <th id="this_year">This Year</th>
    <th id="last_year">Last Year</th>
    <th id="difference">Difference</th>
  </tr>
  <tr>
    <td headers="this_year">591</td>
    <td headers="last_year">222</td>
    <td headers="difference">369</td>
  </tr>
  <tr>
    <td headers="this_year">-33</td>
    <td headers="last_year">67</td>
    <td headers="difference">100</td>
  </tr>
  <tr>
    <td headers="this_year">Fish</td>
    <td headers="last_year">Frogs</td>
    <td headers="difference">Legs</td>
  </tr>
</table>
```

Coding With Style • Comments

- Comments are little "notes for yourself" in the code that do not render on the page. We've been using them all along!

```
<p>          <!-- opening tag! -->
  This is a sample of
  real HTML!<br />  <!-- self-closing tag! -->
  <strong>Isn't that exciting?</strong>
</p>  <!-- closing tag! -->
```



Comments-as-Outline

- Programmers often use comments as an "outline" for their code.

```
<!-- page header will go here -->

<!-- page content will go here -->

<!-- page footer will go here -->
```

When Should You Comment?

- Early and often! If not for yourself, then for people who have to come along and maintain your code later so they don't have to sit there going, "What was that coder thinking???"
- When in doubt, comment. They take negligible amount of bandwidth but can really save you headaches later.

Types of Comments

- Just about every programming language has its own commenting schema, but they tend to fall into a few broad types. Here are the ones you'll typically encounter on the web:

HTML: <!-- comment here -->

CSS: /* comment here */

JavaScript: // comment here or

JavaScript: /* comment here */

ColdFusion: <!--- comment here --->

Classic ASP: ' comment here

VBScript (ASP.NET): ' comment here

PHP: // comment here or

PHP: /* comment here */

We’re Done!

- Congratulations, you’re a master of HTML now. Go build your pet a web page! ;)
- Seriously, tho … questions? Comments?