

LECTURE OVERVIEW

- The Web as a Medium
 - Brief HTML Review: Important Tags, Doctypes
 - Semantic HTML
 - Separation of Structure, Behavior, and Appearance
 - Progressive Enhancement
-

RELATED READING

No related reading this time around.

THE WEB AS A MEDIUM

Web design is defined by a total lack of control over, well, everything. In print design, typography can be exactly set and dimensions may be chosen in a pleasing and exact manner (for instance, using the golden ratio). These conveniences are not available to the web designer. Typography on the web is extremely limited (though a future lecture will describe ways to somewhat circumvent this limitation). We also give up exact control over the dimensions of most elements; vertical dimensions, in particular, are extremely difficult to pin down (and really shouldn't be pinned down, anyway). Thus, design techniques based on proportion and dimensions, like the golden ratio or even the rule of thirds, are much less effective.

Additionally, we lack control over how our users access and view our site. We have no idea whether our users will be accessing our content on a mobile phone, a laptop computer, or a Wii console. Our user may be a high school senior, an elderly man with poor vision who requires the ability to resize text, or a blind person accessing our content via a screenreader (a later lecture on accessibility covers these topics). Our design must work regardless; it must be "bulletproof" and we must accept the fact that it may not look the same to all of our users.

Interactivity, especially with the influx of technology like Ajax and Flash, is a key element of web design; interaction design (or IxD for short) alone is a

huge field. While we will likely touch on the theory and ideas behind interaction principles, we will not talk much about their technical implementation.

BRIEF HTML REVIEW

This class assumes some prior HTML experience. A lack of understanding of the true meaning and usage of some tags prevents many people from creating excellent, semantically correct sites. For reference, here is a list of some of the most important tags in HTML:

HTML	SPAN
HEAD	A
BODY	UL
TITLE	OL
META	LI
LINK	DL
IMG	DT
H1, ..., H6	DD
P	TABLE
DEL	THEAD
EM	TBODY
STRONG	TFOOT
I	TR
B	TH
BLOCKQUOTE	TD
Q	ACRONYM
CITE	ABBR
DIV	ADDRESS

This list is incomplete (no forms, etc.), but it covers most of the work that we'll do in this class and includes a few lesser-known tags like DL and CITE.

Also remember that every site *must* include a valid DOCTYPE to define which version of HTML is being used. For the purposes of the examples in this class, we'll be using XHTML 1.0 Transitional (the corresponding DOCTYPE is shown in the example below), but you can feel free to use whichever you're most comfortable with in your final project.

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

SEMANTIC HTML

Additionally, your HTML should carry semantic meaning on its own. This means choosing the right tags and HTML structure to describe your content.

Examine the B and STRONG tags, for example. These two tags are functionally identical (by default, all browsers will render them as bold). However, their semantic meaning is entirely different: a B tag is purely presentational (and thus is frowned upon in modern usage) while the STRONG tag implies some sort of strong emphasis. Most screenreaders will actually read text within a STRONG tag differently. Therefore, a warning message in the form of `Warning!` should instead be marked up as `Warning!`. This is why a fairly extensive knowledge of common HTML tags is so important.

SEPARATION OF STRUCTURE & APPEARANCE

One of the most widespread ideas in web development is the (almost) total separation of structure (HTML), appearance (CSS), and interactive behavior (Javascript) of any web site.

Essentially, your HTML should not include any attributes or inline CSS styles that define the appearance of an element, nor should it include inline Javascript events that define its behavior. Instead, you should simply link to external CSS and Javascript files that can be reused as much as possible. For example, bad HTML markup for a small grey comment might look like:

```
<font color="#555555">This is a grey comment.</font>
```

Or (less horrible, because it's semantic now):

```
<p style="color:#555555;">This is a grey comment.</p>
```

Ideal markup will have a separate CSS file that controls the appearance of a comment. For example:

```
<p class="comment">This is a grey comment.</p>
...
.comment { color: #555555; }
```

This separation makes the life of a developer much easier.

PROGRESSIVE ENHANCEMENT

As mentioned before, one of the many challenges with web development is the lack of control over what technology is available to users.

As a result, *progressive enhancement* should guide the development process. This process describes how to continually build or improve a site, while maintaining backwards compatibility. Progressive enhancement involves starting off with a bare-bones site (that works for everyone) and adding in additional features and functionality for more advanced users one step at a time. For example, check out the following kind of process:

1. Create the basic HTML structure of the page that defines all the content and structure in a semantically correct way.
2. Add CSS in an external file to define the visual appearance of that content.
3. If necessary, add Javascript in an external file to define the interactive behavior of the site.

At any point in the process, the site should still work for all users.

NEXT LECTURE

Next week, we'll go over the foundations of CSS: selectors and rules, pseudo-classes and pseudo-elements, the box model, margin collapsing, the cascade, and units of length. Naturally, more advanced students can probably skip this.