HTML5

FORGET ALL THAT BAD HTML YOU ALREADY KNOW!

Created by **Abram Hindle**

IF YOU KNOW HTML ALREADY

- How often do you use ?
- How often do you use <table>?
- Do you use <font>?

HTML HAS CHANGED

You can still use all that stuff you learned.

But there's newer better ways that will make your pages just look better!

So bear with me if you already know HTML and consider this an upgrade of your skills!

LET'S START HTMLING!

It's best to start from the bare essentials. You can validate this HTML using **http://validator.w3.org/check**.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<meta charset="UTF-8"/>
</head>
<body>
  Here's your minimal and validating HTML page.
</body>
</html>
```

DON'T FORGET!

You need to tell the world that this is a modern HTML file!

Put your DOCTYPE at the top!

Also enclose your content in the HTML tag!

```
<!DOCTYPE html>
<html>
...
</html>
```

HEAD /HEADER

The head tag is where we put information about the webpage. You will usually include a title here.

Meta tags contain meta information for browsers and other tools to help interpret.

```
<head>
  <title>Page Title</title>
  <meta charset="UTF-8"/>
  <meta name="description" content="Example Page"/>
  <meta name="author" content="Abram Hindle"/>

  <!-- proprietary extensions -->
  <meta name="apple-mobile-web-app-capable" content="yes" />
  <meta name="apple-mobile-web-app-status-bar-style" content="black-translucent" />
  <!-- mobile viewport information -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
  <!-- stylesheets / css -->
  <link rel="stylesheet" href="css/reveal.min.css">
  <link rel="stylesheet" href="css/theme/default.css" id="theme">
  <link rel="stylesheet" href="lib/css/zenburn.css">

</head>
```

BODY

```
<body>
<p>Visible content goes here.</p>
<p>Consider that you should not directly specify layout information
  here but that you should mark up blocks with a class allowing you to
  apply layouts later. You can abstract layout</p>
</body>
```

TAGS

Since many HTML parser expect XML like tags all tags should be closed to allow for ease of reading and parsing.

This is not a requirement of the HTML spec.

Start Tags are enclosed in < and >

End Tags are enclosed in </ and >

```
<tag>
  <enclosedtag>
  </enclosedtag>
</tag>
```

Void or atomic Tags are enclosed in < and />;

```
<tag>
  <atomictag/>
  <voidtag/>
</tag>
```

TAGS

Since many HTML parser expect XML like tags all tags should be closed to allow for ease of reading and parsing.

This is not a requirement of the HTML spec.

Tags can be any capitalization in HTML, in XML they are case sensitive and must be lower case. Stick to lower-case tags if they are HTML to stay XHTML compliant.

PARAGRAPH TAG: <P>

You might have been taught that you can use these <p> tags to start and end paragraphs.

The new style is to enclose paragraphs.

```
<p>
Now bears us onward one of the hard margins,
  And so the brooklet's mist o'ershadows it,
  From fire it saves the water and the dikes.
</p>
<p>
Even as the Flemings, 'twixt Cadsand and Bruges,
  Fearing the flood that tow'rds them hurls itself,
  Their bulwarks build to put the sea to flight;
</p>
```

We can use CSS to style these text blocks. Whitespace in the <p> block doesn't matter.

PARAGRAPH TAG

That code now looks like:

> *Now bears us onward one of the hard margins, And so the brooklet's mist o'ershadows it, From fire it saves the water and the dikes.*
>
> *Even as the Flemings, 'twixt Cadsand and Bruges, Fearing the flood that tow'rds them hurls itself, Their bulwarks build to put the sea to flight;*

PARAGRAPH TAG: <P> AND

The previous example broke up the explicit lines from that canto.

We can make line-breaks explicit with
.

You might have been taught that you can use
, you can but it isn't XHTML or XML complaint.

```
<p>
Now bears us onward one of the hard margins,<br/>
  And so the brooklet's mist o'ershadows it,<br/>
  From fire it saves the water and the dikes.<br/>
</p>
<p>
Even as the Flemings, 'twixt Cadsand and Bruges,<br/>
  Fearing the flood that tow'rds them hurls itself,<br/>
  Their bulwarks build to put the sea to flight;<br/>
</p>
```

We can use CSS to style these text blocks. Whitespace in the <p> block doesn't matter.

PARAGRAPH TAG AND

That code now looks like:

Now bears us onward one of the hard margins,
And so the brooklet's mist o'ershadows it,
From fire it saves the water and the dikes.

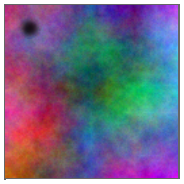
Even as the Flemings, 'twixt Cadsand and Bruges,
Fearing the flood that tow'rds them hurls itself,
Their bulwarks build to put the sea to flight;

<IMG> : THE IMAGE TAG

We want to show images, we cannot embed them in HTML easily so we use hyperlinks (URIs) to reference them and include them.

Remember to include alt tags so they are machine and human-readable.

```
<img src="images/plasma.png" alt="Plasma Pattern"/>
<!-- scaling -->
<img src="images/plasma.png" alt="Plasma Pattern" width="30"/>
<!-- scaling without respecting aspect -->
<img src="images/plasma.png" alt="Plasma Pattern" width="30" height="60"/>
```



<A> : THE ANCHOR TAG

We wouldn't have hypertext without hyperlinks!

Anchor tags let us link to other documents or locations. The href attribute links to a URL. The URL can be relative to the location of the current page. Anchor tags used to place anchors on pages, but in HTML 5 they just navigate to ID'd sections now.

```
<a href="http://slashdot.org">Slashdot.org: News for Nerds Stuff that
  Matters</a><br/>
<a href="http://cnn.com">T</a><a href="http://msn.com">a</a>
<a href="http://softwareprocess.es/">g</a><a href="http://ualberta.ca">s</a>
don't have to be very long.<br/>
<a href="../">The a directory down!</a> Relative Link
```

Slashdot.org: News for Nerds Stuff that Matters
Ta gs don't have to be very long.
The a directory down! Relative Link

TEXT LAYOUT PHILOSOPHY

Let the user agent decide how to display the text. You don't need to control everything. But provide the appropriate semantics first. Then apply layout information to those semantics.

If you want something to show up in a certain make a CSS class and style a span or a div or a paragraph that way.

Don't force the situation with a table.

<DIV> AND <SPAN> TAGS

 and tags have no particular meaning.

 will cause a line break before it is displayed.

 will be inlined.

```
<div>
<span>Now bears us onward one of the hard margins,</span><br/>
  And so the brooklet's mist o'ershadows it,<br/>
  From fire it saves the water and the dikes.<br/>
</div>
<div>
Even <span>as the Flemings</span>, 'twixt Cadsand and Bruges,<br/>
  Fearing the flood that tow'rds them hurls itself,<br/>
  Their bulwarks build to put the sea to flight;<br/>
</div>
```

<DIV> AND <SPAN> TAGS

Looks like this:

> *Now bears us onward one of the hard margins,*
> *And so the brooklet's mist o'ershadows it,*
> *From fire it saves the water and the dikes.*
> *Even as the Flemings, 'twixt Cadsand and Bruges,*
> *Fearing the flood that tow'rds them hurls itself,*
> *Their bulwarks build to put the sea to flight;*

It did not change.

<DIV> AND <SPAN> TAGS: WITH STYLE

We can add a style attribute to these tags

```
<div>
<span style="background-color:#AAAAAA;font-weight:bold">
  Now bears us onward one of the hard margins,</span><br/>
  And so the brooklet's mist o'ershadows it,<br/>
  From fire it saves the water and the dikes.<br/>
</div>
<div style="font-size:150%">
Even <span style="color:red">as the Flemings</span>, 'twixt Cadsand and Bruges,<br/>
  Fearing the flood that tow'rds them hurls itself,<br/>
  Their bulwarks build to put the sea to flight;<br/>
</div>
```

<DIV> AND <SPAN> TAGS: WITH STYLE

Looks like this:

> ***Now bears us onward one of the hard margins,***
> *And so the brooklet's mist o'ershadows it,*
> *From fire it saves the water and the dikes.*
> *Even as the Flemings, 'twixt Cadsand and Bruges,*
> *Fearing the flood that tow'rds them hurls itself,*
> *Their bulwarks build to put the sea to flight;*

STYLE

I showed how to modify the style properties of tags. Tags can have attributes defined liked so:

```
<tag attribute1="value1" attribute2="val2"/>
```

We added style attributes. The style attributes are Cascading Style Sheet properties.

Cascading refers to accumulation of CSS properties (like color, font-weight, etc.).

```
<span style="zoom:2;color:blue;">Hello!
  <span style="zoom:2;font-weight: bold;">Hello!
    <span style="zoom:2;color:red;box-shadow: 10px 10px 5px #888888;">Hello!</span>
  </span>
</span>
```

STYLE

Looks like this:

Hello! ***Hello!*** ***Hello!***

CASCADING STYLE SHEETS

- Cascades
  - Children Inherit Properties
- Style
  - Properties refer to style properties such as layout, position, color, font, background, padding, borders
- Sheets
  - Apply to a page, change a page.

CASCADING STYLE SHEETS

CSS can applied on a per tag level, but can also be applied globally.

```
<!-- Include a file of CSS  -->
<link href="path/to/cssfile.css" rel="stylesheet">


<!-- Inline CSS -->
<style type="text/css">
.angry {
font-weight:900;
zoom: 3;
}
.angry:nth-child(odd)
{
color:green;
transform:rotate(7deg);
-webkit-transform:rotate(7deg); /* Safari and Chrome */
float: left;
}
.angry:nth-child(even)
{
color:red;
transform:rotate(-12deg);
-webkit-transform:rotate(-12deg); /* Safari and Chrome */
float: right;
}
</style>
<p style="color:orange">Apply style directly</p>
<div>
<div class="angry">HULK</div> <div class="angry">SMASH!</div>
</div>
```

STYLE

Looks like this:

Apply style directly

HULK

SMASH!

CSS: PROPERTIES TO KNOW!

- color -- color of the text or object
  - color:green
  - color:#abc (short form)
- font-family -- the font
  - font-family:"Times New Roman"
  - font-family:"Verdana"
- font-size -- the font size
  - font-size:10px;
  - font-size:10pt;
  - font-size:large;
  - font-size:200%;
- font-style -- normal, italic, oblique
  - font-style:normal;

CSS BACKGROUNDS

- background-color -- Background color of a div or a span

```
.bg1 { background-color:red }
.bg2 { background-color:#aabbcc }
```

- background-image -- Background image

```
.bg3 {background-image:url('images/plasma.png');}
```

- On this HTML

```
<p class="bg1">This is example 1</p>
<p class="bg2">This is example 2</p>
<p class="bg3">This is example 3.<br/>
  This is example 3.<br/>
  This is example 3.<br/>
</p>
```

CSS BACKGROUNDS

This is example 1

This is example 2

This is example 3.
This is example 3.
This is example 3.

CSS SELECTORS

Did you notice I had .bg1 and .bg2 in the previous example?

Those are CSS selectors. .bg1 and .bg2 are classes. We can use HTML attributes to label tags with classes that allow the tags to inherit CSS style information.

```
.bg1 { background-color:red; }
.bg2 { background-color:#aabbcc; }
.bg3 {background-image:url('images/plasma.png');}
```

CSS SELECTORS

- .class
  - class selectors let you mark up HTML with a class

```
.highlight { background-color:yellow; }

<p class="highlight">ALERT!</p>
```

- You can also combine classes with HTML tag selectors; this example makes p tags with class="highlight" orange instead of yellow. You can override other CSS as well.

```
p.highlight { background-color:orange; }

<p class="highlight">ALERT in orange!</p>
```

Here's some highlighted stuff in a div
ALERT in orange!

CSS SELECTORS

- #id
  - ID selectors are like class selectors except they aim at the one tag with the id="idtag" as an attribute.

```
#yellowtag { background-color:yellow; }

<p id="yellowtag">Yellow Tag Sale!</p>
```

CSS SELECTORS

- element
  - element selectors let you style entire HTML elements (tags).
  - Important because you might want to theme all divs or imgs or links

```
p { background-color:yellow; }

<p>Great. Everything is yellow now.</p>
```

Great. Everything is yellow now.

CSS SELECTORS

- :context
  - Selectors that behave depending on the context
  - Can be chained with other selectors
  - :hover - when the mouse is over
  - :active - active link
  - :first-letter - operate on the first letter
  - :nth-child(2) - second child

```
span.x:hover { background-color:yellow; }
.x:nth-child(even) { background-color:red; }
```

```
<span class="x">How</span>
<span class="x">is</span>
<span class="x">this</span>
<span class="x">going</span>
<span class="x">to</span>
<span class="x">work?</span>
```

How is this going to work?

CSS POSITIONING

- position: (use left, right, top, bottom)
  - fixed -- stays on one spot in the browser

```
p.fixed { position: fixed; right: 10%; top: 20%; }
```

  - relative -- position relative to where it normally goes.

```
<!-- a little to the left -->
p.relative { position: relative; left:-10px; }
p.bigrelative { position: relative; left:-100px; }
```

FIXED

!relative

Relative

Big Relative

CSS POSITIONING

Ab-solutely!

Redbox!

Bluebox!

- position: (use left, right, top, bottom)
  - absolute -- absolute positioning relative to the first parent that was positioned (often the page itself).

```
p.abs { position: absolute; left: 10%; top: 5%; }


<p class="abs">Ab-solutely!</p>
```

  - z-index can be used to order overlapping elements!

```
p.redbox { z-index: 5; position: absolute; right: 11%; top: 10%;
background-color:red;}
p.bluebox { position: absolute; right: 15%; top: 15%;
background-color:blue;}


<p class="redbox">Redbox!</p>
<p class="bluebox">Bluebox!</p>
```

CSS HELP!

CSS is spread across many specs, so it is hard to really get a clear grasp on it.

- **CSS Specifications** -- Not all of these are available or work
- **CSS3 Tutorial** from W3Schools.
- **CSS Tutorial** from W3Schools.
- **Full property table from CSS2** -- The CSS2 Spec is more encapsulated in one place

HTML FOR USER INTERFACES

The HTML Form elements let us accept input from browsers in a structured way and form HTTP GETs and POSTs.

In the lab you should have covered some of this.

HTML FOR USER INTERFACES

The HTML Form elements let us accept input from browsers in a structured way and form HTTP GETs and POSTs.

<FORM> TAG

The form tag encloses a group of HTML input/UI widgets which can then be submitted as once.

- method - We can specify the HTTP method (POST/GET)
- action - We can specify the URI to GET or POST to.

```
<form action="http://webdocs.cs.ualberta.ca/~hindle1/1.py" method="get">
What is your name? <input name="name"/></br>
What is your quest? <input name="quest"/></br>
What is your favorite colour? <input name="colour"/></br>
<input type="submit"/>
</form>
```

What is your name?

What is your quest?

What is your favorite colour?

Submit Query

<INPUT> TAG

The input tag can take in textual input, passwords, or act as submit button.

- type - button / *checkbox* / color / date / datetime / datetime-local / email / file / *hidden* / image / month / number / password / *radio* / range / reset / search / *submit* / tel / *text* / time / url / week
- name - the name of the data sent to the URI
- value - the default value Hidden types allow you to embed values to send along to help the request.

<INPUT> TAG

```
<form action="http://webdocs.cs.ualberta.ca/~hindle1/1.py" method="get">
What is your name? <input name="name"/></br>
What is your quest? <input name="quest"/></br>
What is your favorite colour? <input name="colour"/></br>
<input type="hidden" name="hitchhiker" value="I'm coming too!"/>
<input type="submit"/>
</form>
```

What is your name?

What is your quest?

What is your favorite colour?

Submit Query

<INPUT> TAG

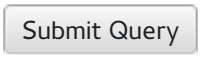
```
<form action="http://webdocs.cs.ualberta.ca/~hindle1/1.py" method="get">
Do you like cake? <input name="cakeliker" type="checkbox" checked/><br/>
How much do you like cake? <input min="0" max="15" value="10"
                                  type="range"/><br/>
<!-- does not work on FF right now -->
When should we eat cake? <input type="date" name="cakewhen"/><br/>
What kind of filling?
Chocolate: <input name="filling" value="chocolate" type="radio"/>
Vanilla: <input name="filling" value="vanilla" type="radio"/><br/>
<input type="submit"/>
</form>
```

Do you like cake? 

How much do you like cake? 

When should we eat cake? 

What kind of filling? Chocolate: Vanilla:



<SELECT> TAG

Select tag is for a dropdown box of options

```
<form action="http://webdocs.cs.ualberta.ca/~hindle1/1.py" method="get">
What kind of cake shall we have?</br>
<select name="caketype">
  <option value="angel">Angel Food Cake</option>
  <option value="devil" selected>Devil's Food Cake</option>
  <option value="cowpatty">Marie Antionette's Cake</option>
</select>
<input type="submit"/>
</form>
```

What kind of cake shall we have?

Devil's Food Cake Submit Query

FILE UPLOAD

You can upload files with the input tag but we need to switch the POST encoding.

```
<form action="http://webdocs.cs.ualberta.ca/~hindle1/1.py"
      enctype="multipart/form-data"
      method="get">
  Choose a file to upload!
  <input name="uploadFile" type="file" />
  <input type="submit"/>
</form>
```

Choose a file to upload! Browse... No file selected. Submit Query

CGI DIVERSION

In the lab you setup a CGI server and wrote a very small CGI script.

Did you know you can write CGI in just about any language?

All it takes is printing HTTP Response headers, reading environment variables and reading from STDIN!

Here's a minimal PERL example

```
#!/usr/bin/perl
use Data::Dumper;
print "Content-type: text/plain\r\n\r\n";
print Dumper(\%ENV);
my $line = 0;
while(<>) {
    print $line++ . " " . $_ . $/;
}
```

CGI DIVERSION

Perl example:

```
hindle1@st-francis:~$ curl -v -X POST -F what=1 -F zuh=weawifsdifsdifsd http://webdocs.cs.ualberta.ca/~hindle1/simple
* About to connect() to webdocs.cs.ualberta.ca port 80 (#0)
*   Trying 129.128.184.6... connected
> POST /~hindle1/simple.pl HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: webdocs.cs.ualberta.ca
> Accept: */*
> Content-Length: 248
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=----------------------------6f7da7637a61
>
< HTTP/1.1 100 Continue
< HTTP/1.1 200 OK
< Date: Fri, 31 Jan 2014 02:07:05 GMT
< Server: Apache/2.2.3 (Red Hat)
< Content-Length: 1570
< Connection: close
< Content-Type: text/plain; charset=UTF-8
<
$VAR1 = {
          'SCRIPT_NAME' => '/~hindle1/simple.pl',
          'SERVER_NAME' => 'webdocs.cs.ualberta.ca',
          'SERVER_ADMIN' => 'helpdesk@cs.ualberta.ca',
          'REQUEST_METHOD' => 'POST',
          'CONTENT_LENGTH' => '248',
          'HTTP_ACCEPT' => '*/*',
          'SCRIPT_FILENAME' => '/compsci/webdocs/hindle1/web_docs/simple.pl',
          'SERVER_SOFTWARE' => 'Apache/2.2.3 (Red Hat)',
          'HTTP_EXPECT' => '100-continue'.
```

CGI DIVERSION

Now in bash

```
#!/bin/bash
echo -e "Content-type: text/plain\r\n\r\n"
/usr/bin/env
cat
```

Results in:

```
hindle1@st-francis:~$ curl -v -X POST -F what=1 -F zuh=weawifsdifsdifsd http://webdocs.cs.ualberta.ca/~hindle1/2.cgi
* About to connect() to webdocs.cs.ualberta.ca port 80 (#0)
*   Trying 129.128.184.6... connected
> POST /~hindle1/2.cgi HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: webdocs.cs.ualberta.ca
> Accept: */*
> Content-Length: 248
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=----------------------------8127c63a32a8
>
< HTTP/1.1 100 Continue
< HTTP/1.1 200 OK
< Date: Fri, 31 Jan 2014 02:08:34 GMT
< Server: Apache/2.2.3 (Red Hat)
< Connection: close
< Transfer-Encoding: chunked
< Content-Type: text/plain; charset=UTF-8
<

SERVER_SIGNATURE=<address>Apache/2.2.3 (Red Hat) Server at webdocs.cs.ualberta.ca Port 80</address>

HTTP_USER_AGENT=curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
SERVER_PORT=80
HTTP_HOST=webdocs.cs.ualberta.ca
HTTP_EXPECT=100-continue
DOCUMENT_ROOT=/var/www/html
SCRIPT_FILENAME=/compsci/webdocs/hindle1/web_docs/2.cgi
REQUEST_URI=/~hindle1/2.cgi
```

CGI DIVERSION

So you can use multiple languages.

I really really really recommend using the CGI library that comes with your language if you do. Mostly for parsing of GET and POST arguments.

Perl:

```
use CGI;
print CGI->new->header();
```

Python:

```
import cgi
import cgitb
cgitb.enable()
# print your header
```

CGI PROBLEMS

What are some problems with CGI?

CGI PROBLEMS

- Slow
- 1 process per invocation
- Ineffecient communication of requests
- Lack of OO representation of a request
- Difficult to share state.

NOW ONTO JAVASCRIPT!

Modern UIs are now part HTML and CSS and part Javascript.

Javascript tends to smooth over the rough edges and fill in the gaps that HTML, HTML Forms and CSS leave.