15-Nov-17

# Learn to Code HTML & CSS

<https://learn.shayhowe.com/html-css/>

## HTML Document Structure

All HTML documents have a required structure that includes the following declaration and elements: <!DOCTYPE html>, <html>, <head>, and <body>.

**<head>**

* including any metadata (accompanying information about the page); the document title (which is displayed on the title bar in the browser window), links to any external files, or any other beneficial metadata.

#### Self-Closing Elements:

#### simply receive their content or behavior from attributes within a single tag

* <br>
* <embed>
* <hr>
* <img>
* <input>
* <link>
* <meta>
* <param>
* <source>
* <wbr>

## **Common CSS Terms**

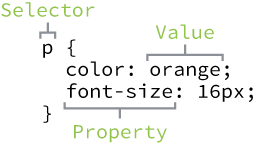
### Selectors

* designates exactly which elements within our HTML to target and apply styles
* generally target an attribute value: id or class value, or target the type of element, such as <h1> or <p>

### Properties

* numerous properties we can use, such as background, color, font-size, height, and width

### Values



### Type Selectors

* target elements by their element type

div { ... }

### Class Selectors

.awesome { ... }

### ID Selectors

#shayhowe { ... }

## Referencing CSS

* In order to get our CSS talking to our HTML, we need to reference our CSS file within our HTML
* within the <head> element of our HTML document, <link> + rel + href
* saved within same folder/ subfolder with HTML file

<head>

<link rel="stylesheet" href="subfolder/main.css">

</head>

## Using CSS Resets

* Every web browser has its own default styles for different elements >>> To ensure cross-browser compatibility: CSS resets
* One of the most popular resets is [Eric Meyer’s reset](http://meyerweb.com/eric/tools/css/reset/)

Week2 HTML

## Semantics

* Semantic code describes the value of content on a page, regardless of the style or appearance of that content
* Benefits: enabling computers, screen readers, search engines, and other devices to adequately read and understand the content on a web page, easier to manage as it shows clearly what each piece of content is about.
* <div> and <span>
  + not hold any such meaning and are simply containers
  + give us the ability to apply targeted styles
  + <div> is a block-level element <span> is an inline-level element
  + Usually come with class/ id > choose name/ value carefully: based on content not style

## Text-Based Elements

### Headings

<h1> … <h6>

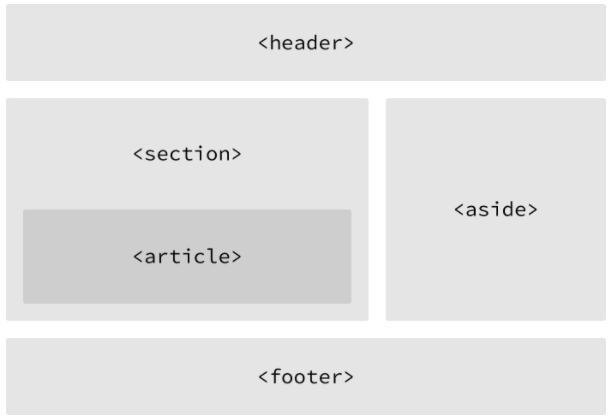
### Paragraphs

<strong> Semantics with <b>

<em> to emphasize with <i>

## Structure

* HTML5 introduced new [structurally based elements](http://dev.opera.com/articles/new-structural-elements-in-html5/), including the <header>, <nav>, <article>, <section>, <aside>, and <footer> elements.



### Header

<header>

### Navigation

<nav>

* links included within the <nav> element will link to other pages within the same website or to parts of the same web page
* one-off links should not be wrapped within the <nav> element, instead: <a>

### Article

* used to identify a section of independent, self-contained content that may be independently distributed or reused

### Section

* commonly used to break up and provide hierarchy to a page.

#### >>> Deciding Between <article>, <section>, or <div> ?

* look at the content.
* If the content is being grouped solely for styling purpose: <div> element.
* If the content adds to the document outline and it can be independently redistributed or syndicated, use the <article>
* If the content adds to the document outline and represents a thematic group of content, use the <section>

### Aside

* holds content, such as sidebars, inserts, or brief explanations

### Footer

## Hyperlinks

### Relative & Absolute Paths

* relative path: pointing to other pages of the same website
* absolute path: to other websites outside of the current one (href attribute value must include the full domain)

### Linking to an Email Address

* href attribute value would be mailto
* ? to bind to others like Subject + encoded space by %20
* body= to add body text + & to separate words

<a href="mailto:shay@awesome.com?subject=Reaching%20Out&body=How%20are%20you"> Email Me</a>

### Opening Links in a New Window

* typically will open in the same window
* open in new window: target="\_blank" (target determines exactly where the link will be displayed, and the \_blank value specifies a new window)

### Linking to Parts of the Same Page

* setting an id attribute at element want to link to+ <a href=”#id”>

week3 – CSS

## Cascade

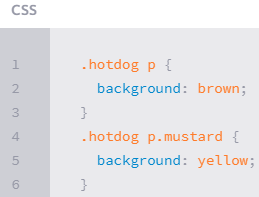
* all styles cascade from the top of a style sheet to the bottom >>> able to add/ overwrite

( what come later will take precedence in the cascade)

## Specificity

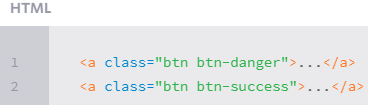
* id > class > type (element) selector

## Combining Selectors



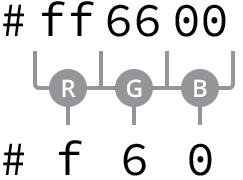
## Layering Styles with Multiple Classes

* to keep the specificity weights of our selectors low, clean code > be modular > multiple classes in HTML



### COLOR

* Hexadecimal Colors: 3 pairs (16\*16)^3 colors!

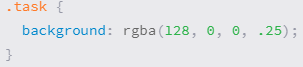


[Adobe Kuler](https://kuler.adobe.com/), a free application that provides a color wheel  to locate hexadecimal color values

* RGB & RGBa Colors:



include an alpha, or transparency, channel by using the rgba() function.



### Lengths

Absolute:

* pixel (px) is equal to 1/96th of an inch; thus there are 96 pixels in an inch

Relative:

* % : defined in relation to the length of another object (parent element).
  + extremely helpful for setting the height and width of elements and building out a web page’s layout.
* em:  equivalent to an element’s font size.
  + When a font size is not explicitly stated for an element, the em unit will be relative to the font size of the closest parent element with a stated font size.

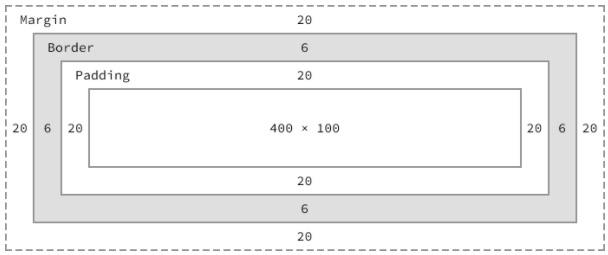
Week4: Opening the Box Model

### Display

* how elements are displayed—as block-level elements, inline elements, or something else
* popular values: block, inline, inline-block, and none

## **Box Model**: *Every element on a page is a rectangular box*

* width, height, padding, border, and margin.



* to determine the actual size of a box we need to take into account padding, borders, and margins for all four sides of the box:
  + Ex: set width of an element is 400px > real: 492px

#### Width

* Block-level elements have a default width of 100%.
* Inline-level elements cannot have a fixed size
* inline-block elements have or expand and contract horizontally to accommodate their content

#### Height

* default by its content

#### Margin

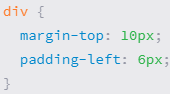
* provide breathing room, keeping all other elements a safe distance away
* top and bottom, are not accepted by inline-level elements.

#### Padding

* unlike the margin property, works vertically on inline-level elements.

Top-down left-right

#### Shorthand – longhand

 it is best to use the longhand properties. Doing so keeps our code explicit and helps us to avoid any confusion down the road.

### Borders

* three values: width, style, and color.
* Style: [different appearances](http://www.quackit.com/html/codes/html_borders.cfm) , most common: solid, double, dashed, dotted, and none
* 
* border-radius property, to round the corners of an element.



### Box Sizing

* CSS3 introduced the box-sizing property, which allows us to change exactly how the box model works and how an element’s size is calculated.
* primary values—content-box, padding-box, and border-box

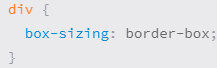
#### Content Box

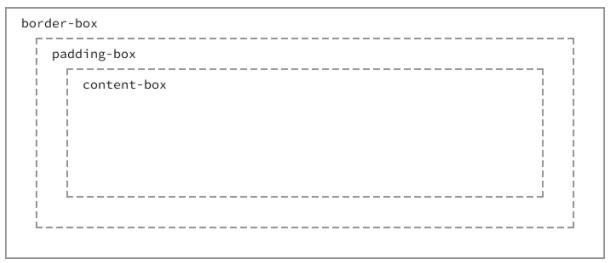
* is the default value, leaving the box model as an additive design
* vendor prefixes (-webkit- (GG Chrome and Apple Safari)

#### Padding Box

* alters the box model by including any padding property values within the width and height of an element
* the actual width will remain, padding increase – content size within shrinks proportionately
* Has Been Deprecated

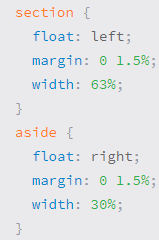
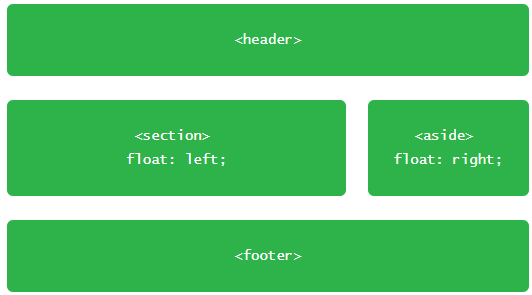
#### Border Box

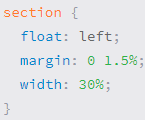
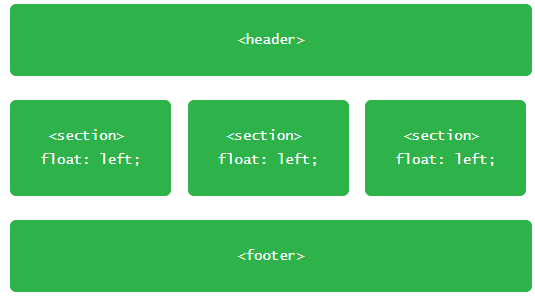
* Similar to Conent box
* the best box-sizing value to use is border-box. (remain value no matter what padding, border are added)
* drawbacks: lack support on older browsers
* 



Week5 Positioning Content

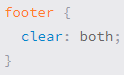
## Option1: Positioning with Floats

* allows us to take an element, remove it from the normal flow of a page, and [position it](http://www.smashingmagazine.com/2007/05/01/css-float-theory-things-you-should-know/) to the left or right of its parent element
* An <img> element: allow the paragraphs to wrap around the image as necessary.
* when an element is floated,
  + it will float all the way to the edge of its parent element/ page
  + take it out of the normal flow of the HTML file >> width of that element to default to the width of the content within it
  + may change an element’s default display value: to block
  + Solution: Can use a fixed width property value to each column | use the marginproperty to create space between elements
* 
* Column: instead of float 1 to left-1 to right >> float all to left + adjust the width of the <section> elements to account for the additional columns and to get them to sit one next to the other

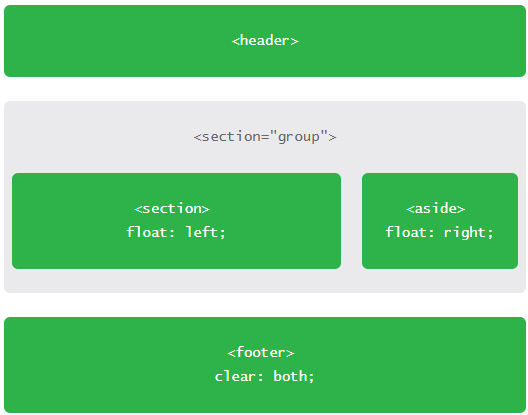
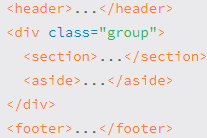
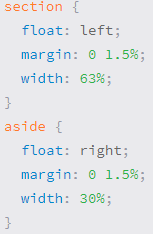
 

### Clearing /Containing Floats

* If not clear >> pitfalls:
  + the proper styles will not render on an element that it is sitting next to or is a parent element of a floated element.
  + unwanted content begins to wrap around a floated element.
* Clear: left, right, and both
  + clear be applied to an element appearing after the floated elements, not before, to return the page to its normal flow.



* **Contain**: similar but ensure that all of our styles will be rendered properly
  + the floated elements must reside within a parent element (a container) to leave the outside of it to normal flow
  + often seen as class name: clearfix or cf or group
  + :before and :after pseudo-elements: dynamically generated elements above and below the element with the class of group
    - Those elements do not include any content and are displayed as table-level elements (similar to block)
    - Clear float within after-&-before elements

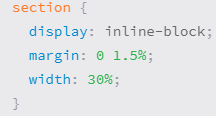
Html CSS  result:

## **OPTION 2: with Inline-Block**

* primarily helpful for laying out pages or for placing elements next to one another within a line.
* inline-block value for the display property will display elements within a line while allowing them to accept all box model properties (height, width, padding, border, and margin) >>> Using inline-block elements allows us to take full advantage of the box model without having to worry about clearing any floats.

### Removing Spaces Between Inline-Block Elements

* make total width too bog >> may push last elements to next row >>> need remove spaces
* 2 easiest options:
  + Put new <section> opening to the previous closing
  + <!—at previous closing tag + -- > at next opening tag



## Creating Reusable Layouts

* always best to write modular styles that may be reused elsewhere
* 1 approach: use inline-block elements to create the grid (layout) of a page + use floats when I want content to wrap around a given element

## **Uniquely Positioning Elements**

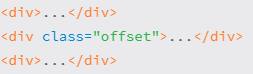
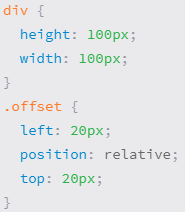
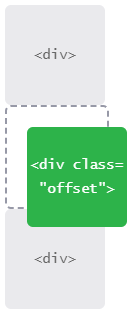
* Floats: remove an element from the flow of a page, often produce unwanted results as surrounding elements flow around the floated element
* Inline-block: unless creating columns, can be fairly awkward to get into the proper position.

>>> position property + box offset (top, right, bottom, and left)

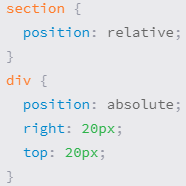
Default position: static >>> relative/ absolute

### Relative Positioning

* Within normal flow of a page, but not allow other elements flow around it (preserve original position)
* box offset properties identify where an element will be moved from given its original position
* may overlap surrounding elements rather than moving that elements away as  margin or padding would.

### Absolute Positioning

* not appear within the normal flow
* original position will not be preserved
* be moved in relation to its closest relatively-positioned-parent-element (not original post.) >> if not exist: to its <body>



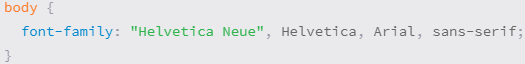


Typically, most positioning can be handled without the use of the position property and box offset properties

Week6: Working with Typography

### Font Family

* The first declared font is the primary font choice.
* Font names consisting of two or more words need to be wrapped in quotation marks “”
* the last font should be a keyword value, which will use the system default font for the specified type (commonly sans-serif or serif)



### Font Size



### Font Style

normal, italic, oblique, and inherit



### Font Variant



### Font Weight

normal, bold, bolder, lighter, and inherit



 100, 200, 300, 400(normal), 500, 600, 700(bold), 800, and 900



Note: Before using a numeric value, we need to check and see whether the typeface we are using comes in the weight we’d like to use.

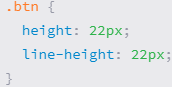
Ex: Times New Roman typeface: 400-normal, 700-bold. If use 900 > default to 700

### Line Height

* Distance b/w 2 text lines
* Best practice: = 1.5\*font-size (150% or 1.5)



* Center vertically buttons, alert messages, and other single-line text blocks:



### Shorthand Font Properties

* from left to right:

font-style variant weight **size** / height **family**.



#### CSS Pseudo-Classes



## **Text Properties**

* align, decorate, indent, transform, space text

### Text Align

* left, right, center, justify, and inherit
* Align (text within an element) =/ Float (entire element)

### Text Decoration

* none, underline (default), overline, line-through, and inherit

### Text Indent

* (inward): + (outward): -

### Text Shadow

* 3 length (horizontal(+/-), vertical (+/-), blur radius) + 1 color
* Other option: box-shadow shadow whole element



### Text Transform

none, capitalize, uppercase, lowercase, and inherit.

### Letter Spacing

* Adjust space between letters (useful when change font-size) +/-

### Word Spacing

* Adjust space between words (useful when change font-size) +/-

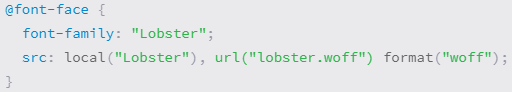


## Web-Safe Fonts

…

## **Embedding Web Fonts**

via the CSS @font-face



## Citations & Quotes

* <cite>: Used to reference a creative work, author, or resource
* <q>: Used for short, inline quotations
* <blockquote>: Used for longer external quotations







### External Quotation

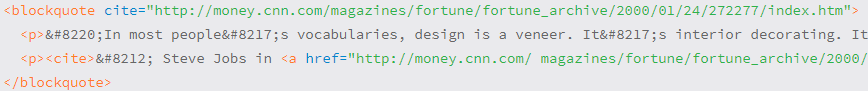
To quote a large block of text that comes from an external source and spans several lines

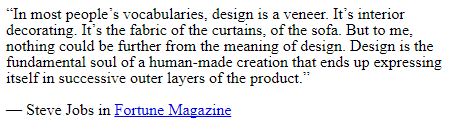


### External Citation

<blockquote> element with a cite attribute to specify where the quote originally appeared

In the <blockquote>element, the <cite> element, along with an <a> element, provides an additional citation and reference for the quote that is visible to users.





Week7: Backgrounds & Gradients

## Background Color



## **Background Image**



* by default the background image will repeat horizontally and vertically from the top left of the given element to fill up the element’s background.

>>> background-repeat & background-position

### Background Repeat

* repeat, repeat-x (horizontally), repeat-y (vertically), and no-repeat

### Background Position

* by default , be positioned at the left top corner
* background-position property requires two values: a horizontal & a vertical offset. If 1 > horizontal and vertical default to 50%



* value can be: px, %, keywords, any length measurement

### Shorthand Background Image Values

Color – image – position – repeat

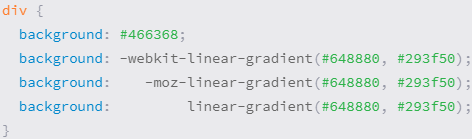
background: #b2b2b2 url("alert.png") 20px 10px no-repeat;

## **Gradient Backgrounds**

* Within CSS, gradient backgrounds are treated as background images
* Gradient background values were one of the values that required the use of vendor prefixes (so that browser can support recently developed CSS features)
* Linear - radial

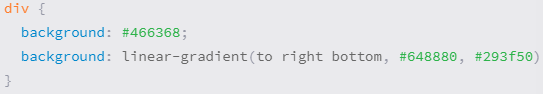
### Linear Gradient Background

* linear-gradient() function within the background or background-image property.
* 2 values: beginning+ending color
* put in a default background property with a solid color (as a fallback should a browser not support gradient backgrounds)

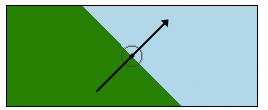


### Direction of a Gradient Background

* by default, change from top to bottom
* use keywords to change: to left/ right/ top/ bottom



* -degree values are also acceptable

45deg

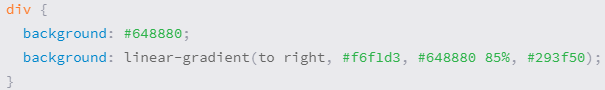
### Radial Gradient Background

* From inside to outside
* 1st color sit in the absolute center

Tool for newbie:  [CSS3 gradient generators](http://www.cssmatic.com/gradient-generator)

### Gradient Color Stops

* Add multiple colors to a gradient > color stops
* By default, equal distance, 1st start at 0% - last end at 100%>> want to control > add location (length)



## **Multiple Background Images**

* Separated by ,
* Come first: foremost – last: rearmost

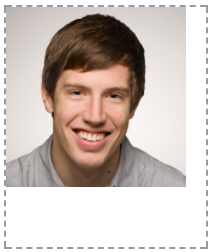
background: url("tick.png") 20px 50% no-repeat, linear-gradient(#72c41f, #5c9e19);



## **New Background Properties**

### CSS3 Background Size

* Keyword auto: either used for width/ height – preserve aspect ratio of background image
* % : in relation to element’s size, not original size of the background image

background-size: auto 75%;

* Other keywords:
  + Cover: image will be resized to cover whole element (ratio is preserve but some part may be cut off)
  + Contain: contrast to Cover (full image but usually not occupy whole element)

### CSS3 Background Clip & Background Origin

div {

background: url("shay.jpg") 0 0 no-repeat;

background-clip: padding-box;

background-origin: border-box;

}

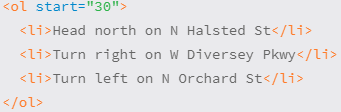
Week8: Creating Lists

## Unordered Lists

## **Ordered Lists <ol>**

### Start Attribute

* defines the number from which an ordered list should start (default 1)

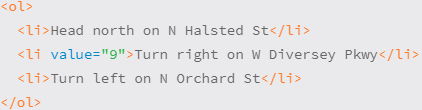
 

### Reversed Attribute



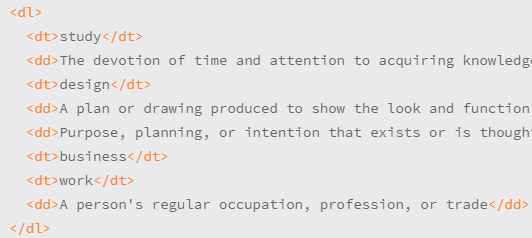
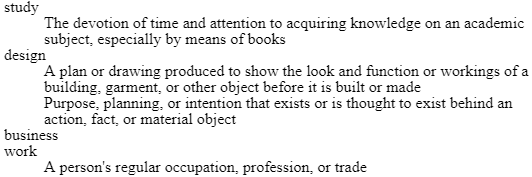
### Value Attribute

* Change order no. of a list, below it will be recalculated accordingly

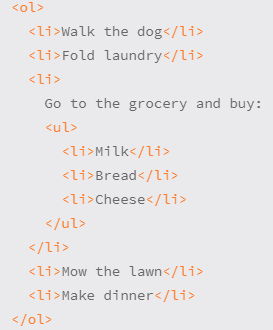
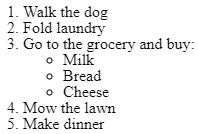
 

## **Description Lists <dl>**

* used to outline multiple terms and their descriptions (ex: a glossary)
* description term <dt> & child: description <dd>

## **Nesting Lists**

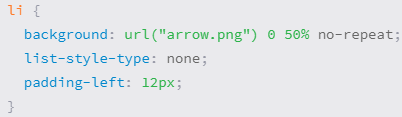
## **List Item Styling**

### List -Style Type Property

* Any list-style-type property value can be added to either unordered or ordered lists > possible to use a numeric marker on <ul>
* none – disc (filled circle) – circle (hollow) – square – decimal …

list-style-type: square;

### Image as a List Item Marker

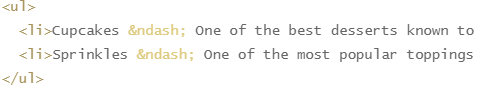
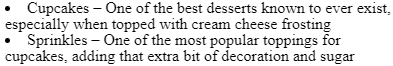
* When default list-style-type property values are not enough >> customize
  + Remove default
  + Add a background imange
  + Add padding



### List Style Position Property

* By default: outside  > can change to inside / inherit

list-style-position: inside;

### Shorthand List Style Property

list-style-type followed by list-style-position.

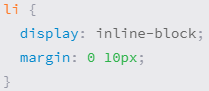


## **Horizontally Displaying List**

by making the display property value of <li> elements inline or inline-block or by floating

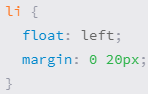
### Displaying List

* use the inline-block property value rather than the inlineproperty value
  + easily add vertical margins & other spacing
* quick, but list item markers will be removed

### Floating List

* if want to keep markers
* float to left: align horizontally elements next to each other w/o any space + horizontal margin/ padding to prevent markers being on top of other <li>

* Note: remember to clear floats (by clearfix) and return to normal flow

Week9: Adding media

## **Adding Images**

* <img>
  + Src: url
  + Alt: describes the contents of an image (search engines…) displayed if img not showed
* *jpg* (faster load – phototgraphs), *png* (icons, background – low color count), gif

### Sizing Images

* Tell browser before load > browser can reserve space and render the page faster
* width and height: if use both HTML attributes & CSS properties > CSS take precedence
* to manage numerous images that all need to be the same size: >> CSS to resize
* Set both width & height may cause distortion sometimes

#### Positioning

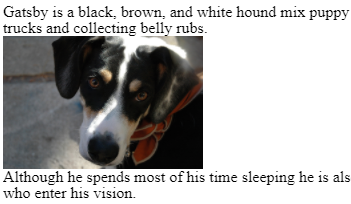
#### Inline:

* Default
* Height of the existing line will adjust = image height > may have gaps

>>> common: block-level or float to side

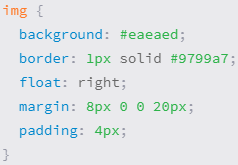
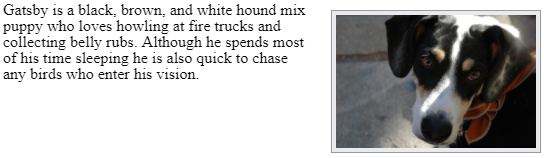
#### Block

* image appear on its own line

display: block; 

#### Float

* Flush to lest/ right
* Use margin property to provide spacing (and padding, border, and background to build a frame for the image)

#### Side Note: when to use  Image Element vs. a Background Image?

* <img> element within HTML: relevant to the content, hold semantic value
* background or background-image property CSS: design/ user interface

## **Audio** <audio>…</audio>

### Audio Attributes

autoplay, controls, loop, and preload.

* By default, not display
* To display: controls <audio src="jazz.ogg" controls></audio>
* By default, auto preload all >> page load faster: preload attribute with a value of metadata or none

### Audio Fallbacks

* 3 popular file formats:  ogg, mp3, and wav
* To support older browsers: replace src attribute with <source>
* Pick up what format browser recognize, if none >> link to download



## **Video**

* Similar to audio
* Take up space >>> should specify  width and height CSS
* Specify *size:* help browser render video faster and allocate proper display space

### Poster Attribute

* Specify an image (in the form of URL) to be shown representing the video

<video src="earth.ogv" controls poster="earth-video-screenshot.jpg"></video>

### Video Fallbacks

* Similar to audio, last fallback: youtube or Vimeo



## **Inline Frames** <iframe>

* to embed another HTML page within the current page

<iframe src="https://www.google.com/maps/embed?..."></iframe>

* styles:
  + frameborder, width, and heightHTML attributes
  + border, width, and height CSS properties

## **Figures & Captions** <figure> and <figcaption>

### Figure

 If the <figure> element is moved from the main portion of a page to another location (for example, the bottom of the page), it should not disrupt the content or legibility of the page.

## Figure Caption <figcaption>

## the <figcaption> element may replace an <img> element’s alt attribute if the content of the <figcaption> element provides a useful description of the visual content of the image.

### 

### 

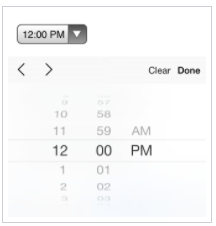
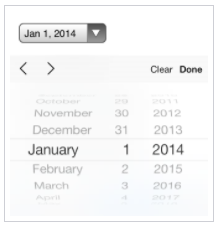
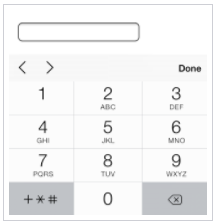
Week10: Forms

By default, every browser has its own interpretation of how the styles for form controls should appear >>> may need style it to ensure consistent (font-based..)

## **Text Fields & Textareas**

* <input> primary element to obtain user input
  + Type: text/ password/ email/ date/ url/ radio/ checkbox… If browser not understand > fall back to text
  + Name:  as the name of the control and is submitted along with the input data to the server.

|  |
| --- |
| <input type="text" name="username"> |

### Textarea

* can accept larger passages of text spanning multiple lines. Only accept Text
* Start + end tags
* attributes: cols for width in terms of the average character width and rows for height in terms of the number of lines of visible text
* size: width and height properties within CSS.

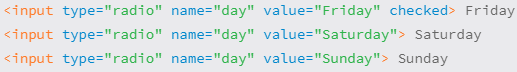
 

## **Multiple Choice Inputs & Menus**

checked Boolean attribute for radio buttons and check boxes

selected for Menus

### Radio Buttons

### Check Boxes

### Drop-Down Lists <select> + <option>s

* Boolean attribute multiple: allow to select >1 options

## **Form Buttons**

### Submit Button

* By default: <button> element acts as if it has a type attribute value of submit

## **Other Inputs**

### Hidden Input

* pass data to the server without displaying it to users
* can be found by viewing the source code of a page>> not be used for sensitive or secure information.



### File Input

* allow users to add a file to a form, much like attaching a file to an email

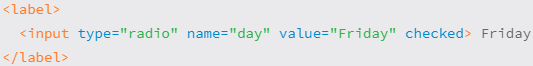
 

## **Organizing Form Elements** labels, fieldsets, and legends

### Label

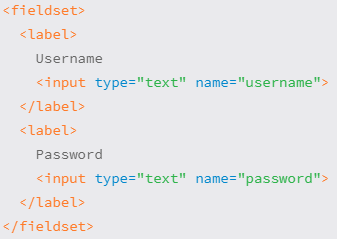
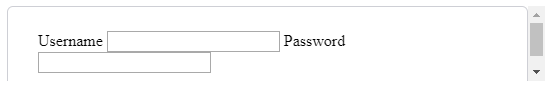
* provide captions or headings for form controls
* for + id attribute / wrap: allowing users to click on the <label> element to bring focus





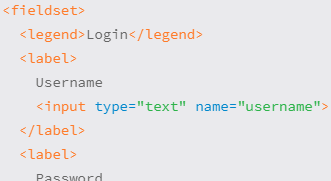
### Fieldset

* group form controls and labels into organized sections
* a block-level element
* by default, also include a border outline, which can be modified using CSS

### Legend

* provides a caption, or heading, for the <fieldset> element

## **Form & Input Attributes**

### Disabled

* turns off an element or control so that it is not available for interaction or input
* <input type="text" name="username" disabled>

### Placeholder

* provides a hint or tip within the form control of an <input> or <textarea> element
* <input type="email" name="email-address" placeholder="name@domain.com">

### Required

* enforces that an element or form control must contain a value upon being submitted to the server
* <input type="email" name="email-address" required>

### Additional Attributes

* Autocomplete, autofocus, maxlength,…

;

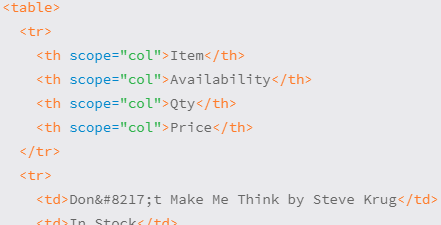
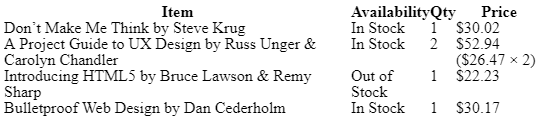
Week11: Organizing Data with Tables

## **Table**

<table>, <tr> (table row) <td> (table data)  <th> (table header)

### Table Header <th>

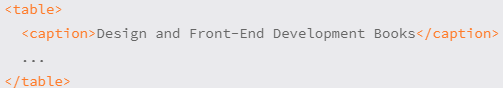
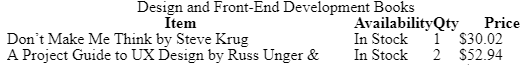
* The scope attribute helps to identify exactly what content a table header relates to
  + col: every cell within the column relates directly to that table header
  + row: every cell within the row relates directly to that table header

## **Table Structure**

 <caption>, <thead>, <tbody>, and <tfoot>

### Table Caption

### Combining Multiple Cells

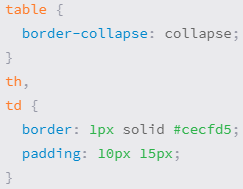
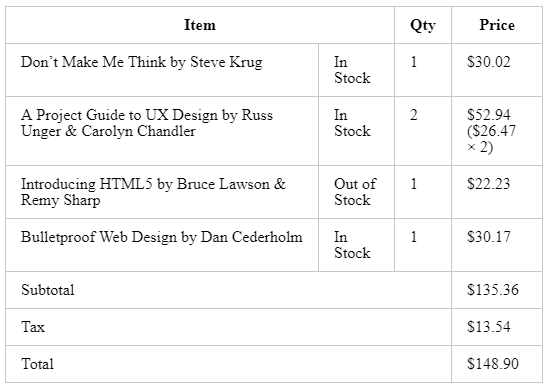
* colspan and rowspan

## **CSS**

## **Table Borders** border-collapse & border-spacing

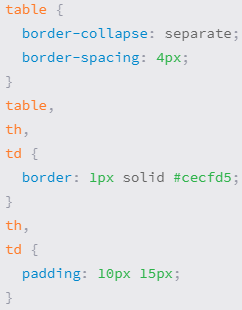
### Border Collapse

* separate (default): different borders will stack up next to one another
* collapse: condenses the borders into one, table cell as the primary border

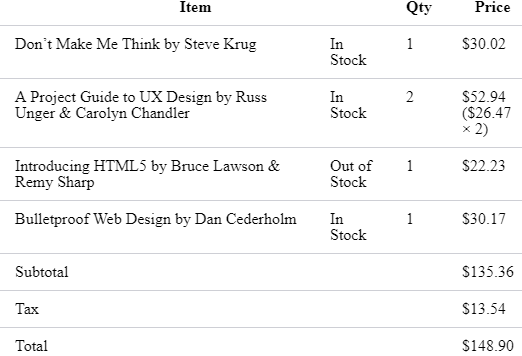
### Border Spacing

* determine how much space, if any, appears between the borders (with separate)
* 2 length values: horizontal vertical spacing

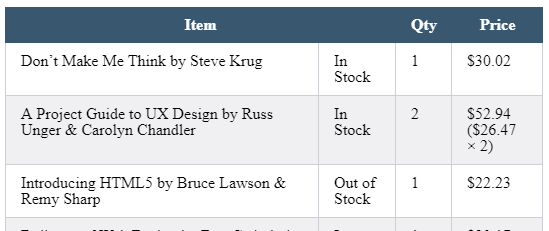
### Adding Borders to Rows

* table’s border-collapse: collapse + bottom-border each cell + 0 for :last-child

## **Table Striping**

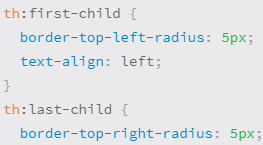
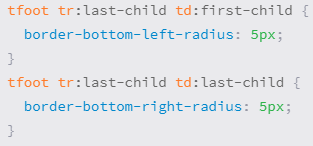
* “stripe” table rows with alternating background colors
* use the :nth-child with an even or odd argument to select <tr>



## **Aligning Text** text-align/ vertical-align

* text-align : move text horizontally
* vertical-align:  works only with inline and table-cell elements—it won’t work for block, inline-block, or any other element levels.

Other styles:

tbody td:only-of-type | tbody td:first-of-type

# lastweek: Writing Your Best Code

### Use the Alternative Text Attribute on Images

alt

### Separate Content from Style

### pages that take longer to load

### difficult to maintain

>>> use external style sheets, classes to target elements

### Proper Class Names

### all lowercase and should use hyphen delimiters.

alert-message

### Proficient Selectors

### The longer a selector is >> the higher specificity it will contain >>

more likely a selector is to break the CSS cascade and cause undesirable issues

### let’s not use IDs within our selectors:

IDs are overly specific, quickly raise the specificity of a selector, and quite often break the cascade within our CSS files

>>> shorter and primarily direct selectors. only two to three levels deep

### Use Specific Classes When Necessary

### when a CSS selector is so long and specific that it no longer makes sense

### >> creates a performance lag and is strenuous to manage >> instead, use class

### Use Shorthand Properties & Values

## **Additional Resources & Links**

Every lesson has come with a few resources for additional learning and discovery. Outlined below is a longer list of resources, as well as beneficial links.

### HTML & CSS

* [Mozilla Developer Network](https://developer.mozilla.org/en-US/) via Mozilla
* [Opera.Dev](http://dev.opera.com/) via Opera
* [HTML and CSS Tutorials](http://www.htmldog.com/) via HTML Dog
* [DevDocs](http://devdocs.io/) — Instant documentation search

### Design Inspiration

* [Dribbble](http://dribbble.com/)
* [Pattern Tap](http://patterntap.com/)
* [Premium Pixels](http://www.premiumpixels.com/)

### Frameworks & Style Guides

* [Web Style Guide](http://webstyleguide.com/wsg3/index.html)
* [Bootstrap](http://getbootstrap.com/)
* [Foundation](http://foundation.zurb.com/)
* [Skeleton Framework](http://getskeleton.com/)
* [Google HTML/CSS Style Guide](https://google.github.io/styleguide/htmlcssguide.xml)
* [GitHub Styleguide](https://github.com/styleguide/)

### Icons

* [Helveticons](http://hlvticons.ch/) via Goodbye Horses
* [Ion Icons](http://ionicons.com/) via Ben Sperry
* [Fugue Icons](http://p.yusukekamiyamane.com/) via Yusuke Kamiyamane
* [famfamfam Silk Icons](http://www.famfamfam.com/lab/icons/silk/) via Mark James
* [Pictos](http://pictos.cc/) via Drew Wilson
* [Picons](http://picons.me/)
* [The Noun Project](http://thenounproject.com/)

### Miscellaneous

* [COLOURlovers](http://www.colourlovers.com/) — Color trends and palettes
* [ColorHexa](http://www.colorhexa.com/) — **Color encyclopedia**
* [Modernizr](http://modernizr.com/) — JavaScript feature detection library
* [jQuery](http://jquery.com/) — Feature-rich JavaScript library
* [Google Hosted Libraries](https://developers.google.com/speed/libraries/devguide) — Content distribution network for JavaScript libraries
* [Copy Paste Character](http://copypastecharacter.com/) — Copying the “hidden” characters