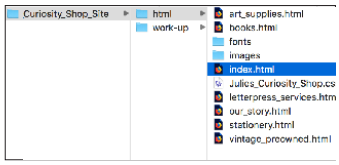


Example: necessary structure of site directory (folders/files)

FILE MANAGEMENT & SITE DIRECTORIES

Structure and keep all project folders / files where they will need to be from the very beginning of a project:

PROJECT FOLDER (WITH DESCRIPTIVE NAME) *Holds everything for each unique project (includes process folder).*

ROOT FOLDER CALLED "HTML" *Main folder that holds files for the actual site: an html file for each page, the css file, a folder of images used on the site, a folder for font files used on the site (if needed). Eventually gets uploaded to a server. The server will look for the same folder structure there as the files were structured on your computer.*

FIRST PAGE – ALWAYS CALLED "INDEX.HTML" *The server will look for the index.html file and use it as the home page.*

CSS: CASCADING STYLE SHEET CONCEPT

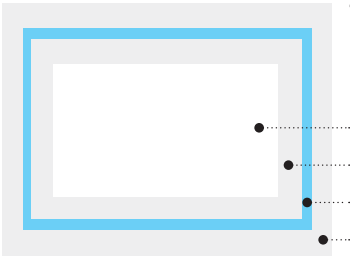
THE CASCADE The principle that styles can ripple to — or be inherited from — elements in a file, or to many associated files and their elements. Makes shaping and styling elements easier and more efficient.

Three levels of style sheets: browser (3), author (2), user (1)

Applying css to html documents: CSS is useless on its own; must be associated with HTML. Apply 3 ways:

- **inline** — quick-fix method, styles can occur straight in the body of an HTML doc; *discouraged*.
- **embedded** — where CSS rules are defined and live in the head of the HTML doc; fine to use for styles applying to only one page in a site.
- **external** — best way of attaching CSS to a multi-page site; a separate file linked in the html <head> tag.

BOX MODEL



"Every element on a web page is surrounded by a force field — a simple, multi-layered box that can be manipulated to create sophisticated effects." (Griffiths) At left is an exaggerated view of the "layers" surrounding any content item such as a box with text and/or images in it, a box with navigation, etc.:

CONTENT *This is the viewable text, images, movies, etc. inside of html 'containers.'*

PADDING *This is an invisible space cushion **inside** of the border. Can choose its size.*

BORDER *This is the outline of a box. It can be invisible, or visible (size and style).*

MARGIN *This is an invisible margin **outside** of the border. Can choose its size.*

SPECIFIC TAGS

Learn via many HTML and CSS cheat sheets available online (see: Camp Site > Basecamp > Week 2).

GENERALIST TAGS

BLOCK-LEVEL ELEMENTS html elements that are themselves, or contain, common chunks of info. They start and end with a hard return by default (example: h1–h6, p, and others).

DIV TAG A block-level html element; used to mark up a *'division'/section* of information so that it can pieces of info can be bundled together. Can also be used to separate chunks of information.

INLINE ELEMENT They stay part of the element in which they exist (examples: images, links, others...); no appearance of a hard return before or after.

SPAN TAGS A generic html element that can be used to tag untagged content with special classes or IDs (example: content in need of visual emphasis inside of a paragraph).

CUSTOM TAGS with CLASS and ID SELECTORS

HTML tags can be customized with the use of class and ID selectors in CSS, leading to more visual flexibility and uniqueness. Without them, you have a finite set of pre-existing tags that limit the range of design variety.

CLASS SELECTOR (.) A unique name you can give to an html element to style instances of it with css. Can be used many times on a single html page; can also be used across multiple pages in a site. The name of a class should reflect the function of the class. In CSS, the unique name is preceded by a "." [dot]. For example:

```
.SongTitle {  color: pink;
              font-weight: bold;  }
```

In the corresponding HMTL, the HMTL tag can be used to mark-up class selectors, as in:

```
<h3 class="SongTitle"> After Hours </h3>
```

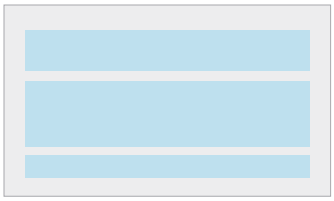
ID SELECTOR (#) A unique name you can give to an html element to style instances of it with css. Can be used only once on a single html page; but can be used across different pages in a site. The name of an ID should reflect the function of the ID . In CSS, the unique name is preceded by a "#" [hashtag]. For example:

```
#sub_navigation {   background-color: pink;
                    border: 3px solid red;
                    padding: 2em;   }
```

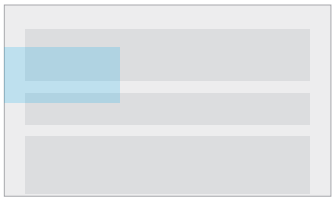
In the corresponding HTML, the mark-up identifies the ID selector by name, as in:

```
<div id="sub_navigation"> [content goes here…] </div>
```

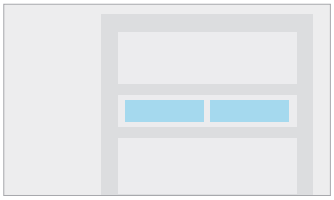
APPROACHES TO LAYOUT AND POSITIONING HTML ELEMENTS



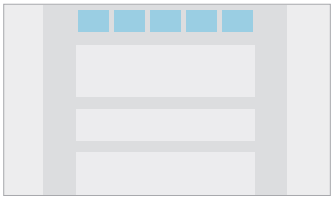
NORMAL FLOW / STATIC All block level elements appear vertically, one beneath the other. The order of the content appears in the order that it is typed in the html. The default positioning behavior. This is called "normal flow."



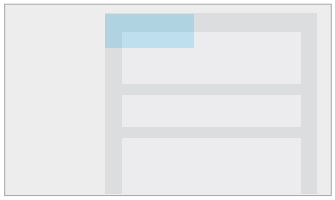
FIXED A form of absolute positioning that fixes the element in relation to the browser edges itself. These don't affect other elements' positions and don't move when the user scrolls up/down the page. Always stays.



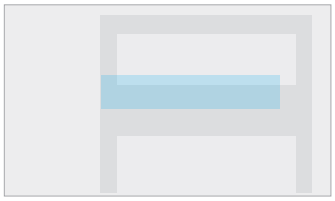
FLOAT Allows elements to remain part of the flow of a page while appearing side-by-side. Functions as a form of text wrapping. Used to be used for entire layouts.



FLEXBOX One-dimensional layout control; that is, good for organization of elements either horizontally or vertically. A good alternative to floats.



ABSOLUTE Removes an item from normal flow and positions it in relation to its containing element. Surrounding elements ignore the space that would have been occupied by it.



RELATIVE Offests an element relative to where it would have been in normal flow. Surrounding elements stay in their normal flow. Children of relative elements can be positioned with 'float' and 'clear.'

STICKY A combination of relative and fixed positioning: the element behaves as relative (scrolls with the content) until it reaches a specified threshold, when it then acts as fixed ("sticks" in relation to the browser).

CSS GRID ("GRID")
Two-dimensional layout control that enables the horizontal *and* vertical organization of elements across a true grid (use of columns ***and*** rows). Enormous layout potential with much less code than other methods.