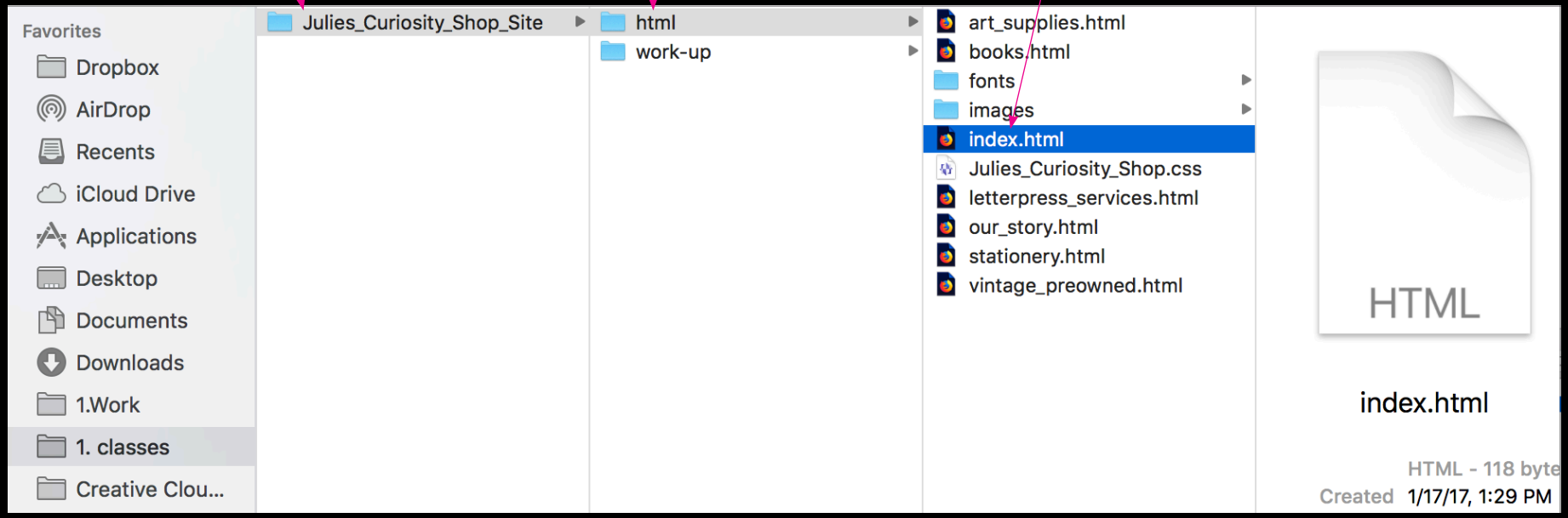
FILE MANAGEMENT:
Site Directories

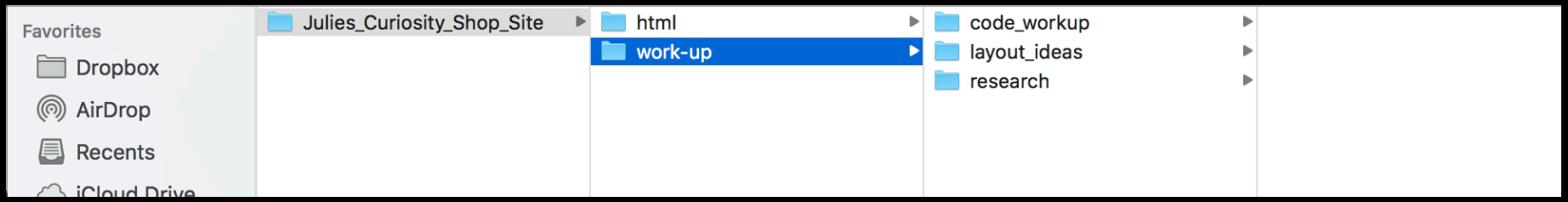
FOLDER + FILE STRUCTURE

PROJECT FOLDER
(ACTUAL NAME)

ROOT FOLDER
(ALWAYS "HTML")

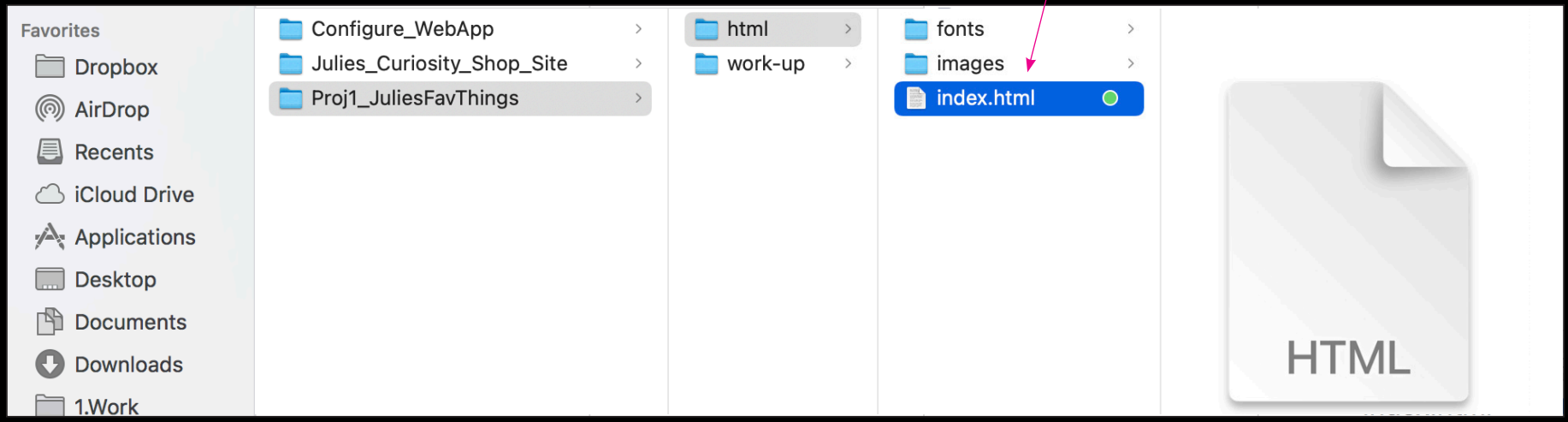
PAGE 1
(ALWAYS "INDEX.HTML")

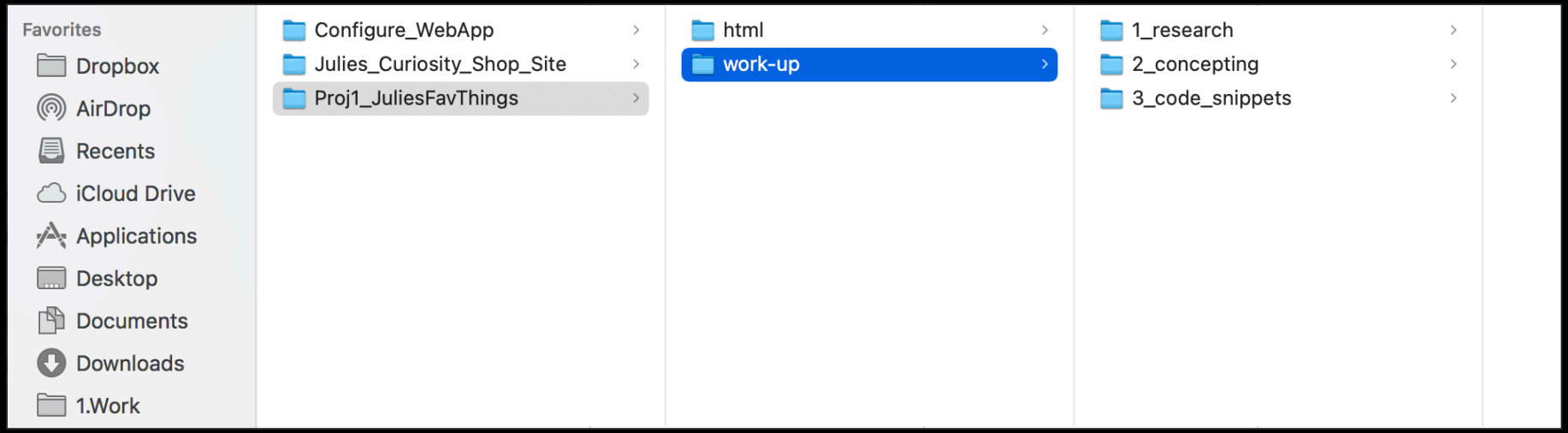




SINGLE-PAGER: FOLDER + FILE STRUCTURE

PAGE 1
(ALWAYS "INDEX.HTML")





HTML & CSS: *Layout Basics*

1: *Browser itself*

2: *Author/designer*

3: *User*

CONDITIONS:
THE CASCADE CONCEPT

INHERITANCE:

***Parent elements* &**
Child elements

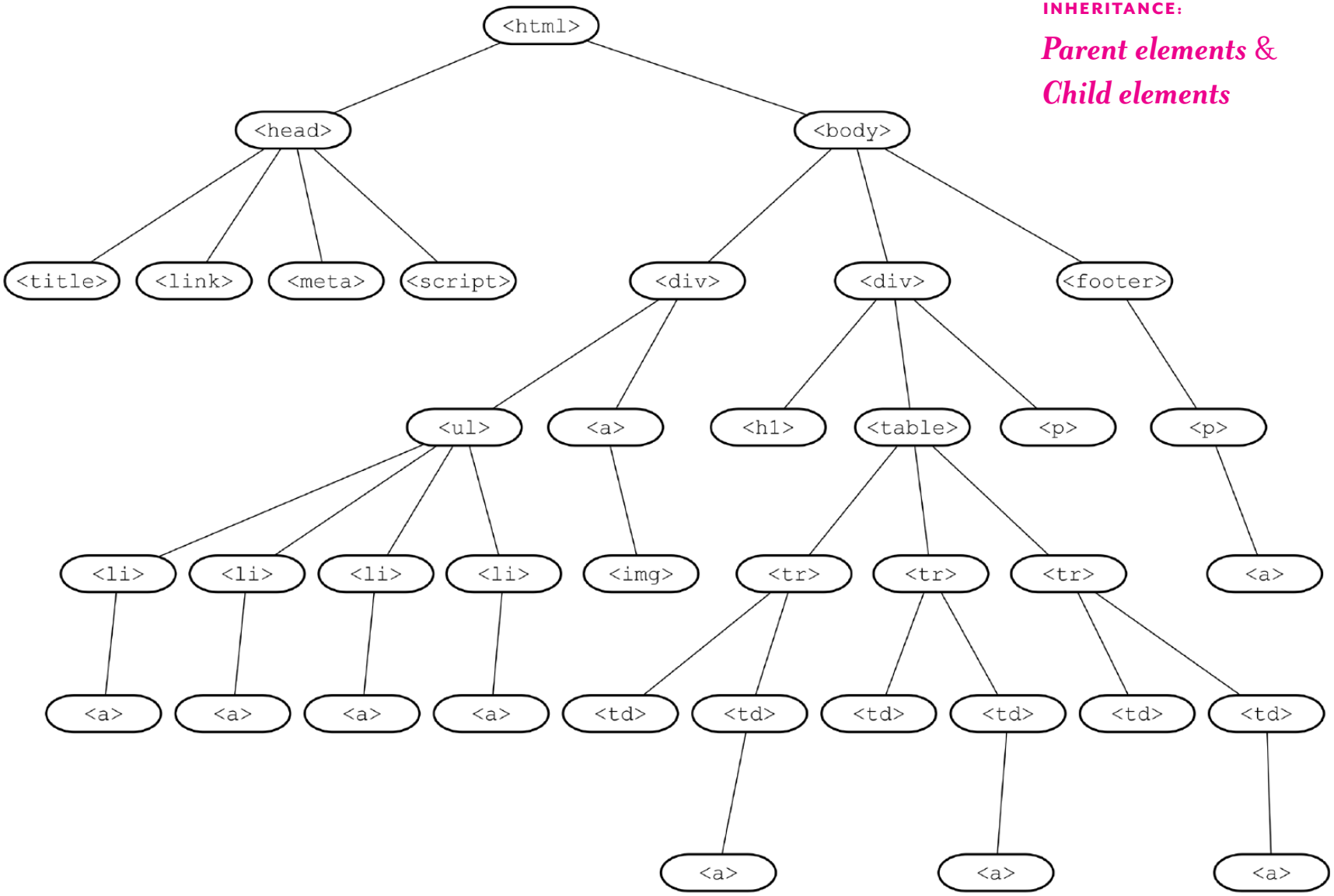
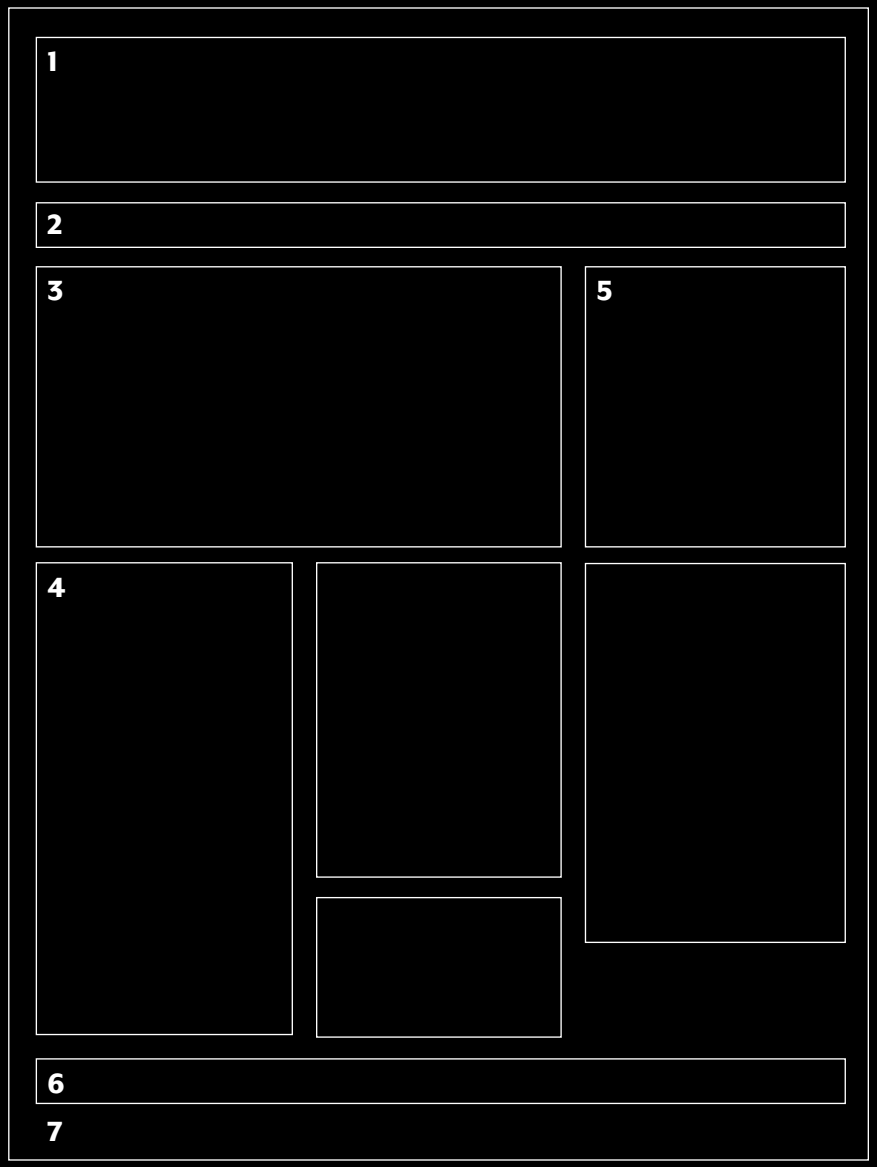


image: Photo by Simon Hurry on Unsplash

ANATOMY OF A WEB PAGE

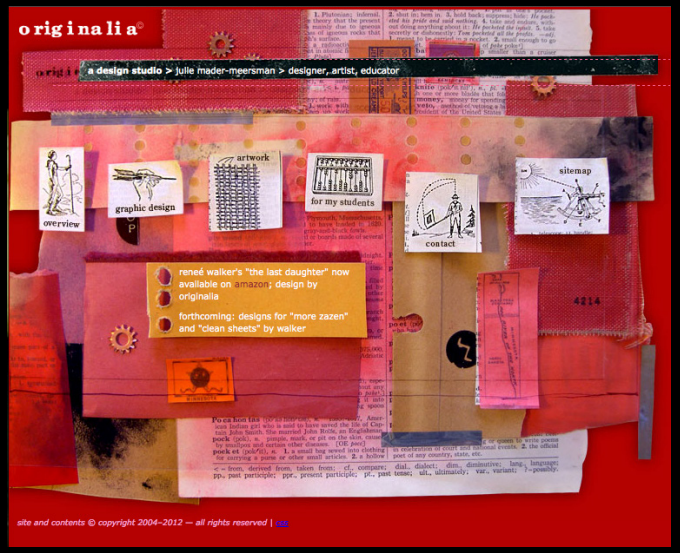


1. *Header*
2. *Navigation*
3. *Feature*
4. *Body/Content*
5. *Sidebar*
6. *Footer*
7. *Background*

APPROACHES TO WEB LAYOUT

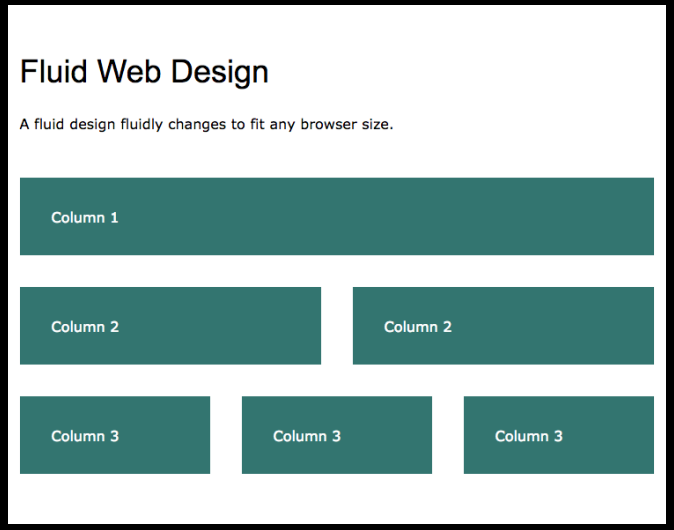
1) FIXED WIDTH

sizing is in pixels



2) FLUID / ELASTIC

sizing is %-based or em-based



3) RESPONSIVE

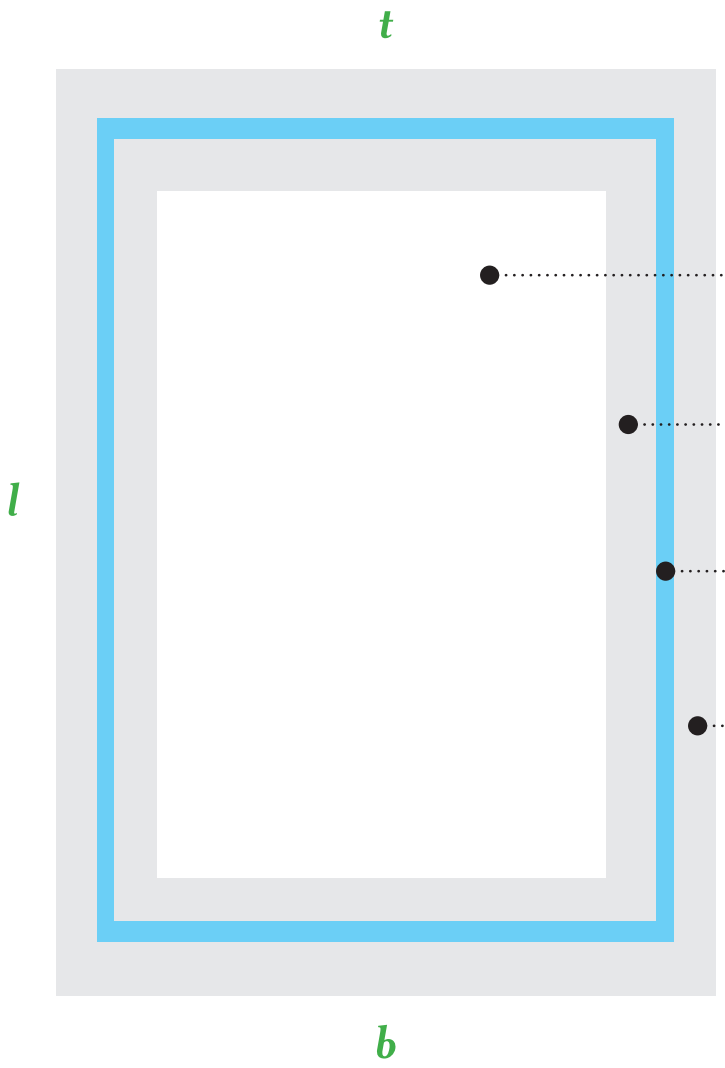
content first; fluid layout with flexible images, and media queries; sizing is %-based or em-based

4) INTRINSIC OR RWD "PLUS"

columns and rows w flexbox and css grid; sizing uses fr units; media queries as needed

"Every element [i.e.,tag] on a web page is surrounded by a force field — a simple, multi-layered box that can be manipulated to create sophisticated effects." — PATRICK GRIFFITHS

THE BOX MODEL: AN EXAGGERATED VIEW



CONTENT *The viewable text, images, lists, links, etc. inside of an html element*

PADDING *An invisible space cushion **inside** of the border. Can choose its size on any edge.*

BORDER *The outline of a box; can be visible/invisible. Can choose size, style and color.*

MARGIN *An invisible space **outside** of the border. Can choose its size on any edge.*

ORDER *Values can be assigned to any edge individually in clockwise order (top, right, bottom, left).*

THE BOX MODEL:
EXPLAINED BY WEBFLOW

The box model for beginners web design tutorial

BLOCK-LEVEL AND INLINE LEMENTS

BLOCK LEVEL HTML ELEMENTS

– "stack" vertically by default

- header
- h1
- p
- footer

Always begin on a new line. Run edge to edge (full width) across a page or parent element (L to R). Takes up as much room as its content needs. Tend to be "big chunks."

EXAMPLES:

divs
H1–H6 tags, P tags
header tags
footer tags
etc.

INLINE HTML ELEMENTS

– stay within chunks by default

- h1
- p Hi . .hyperlink. . can help you with your banking
- footer

Do not automatically appear on a new line or have any spatial breaks around them, by default.

EXAMPLES:

links
image tags
span tags

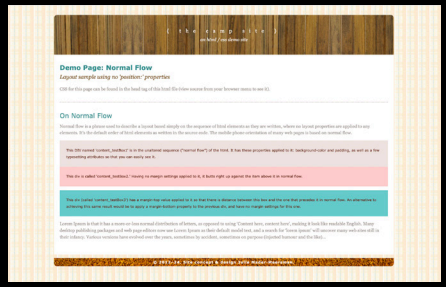
=

NORMAL FLOW

– default behavior / appearance of html

- [logo + navigation]
- ***Welcome banking customer***
- Hello… Griffin G. can help you with your banking inquiry today.
- [credits + contact info]

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.



HTML5 & CSS3 CHEAT SHEETS

image: veign.com

CLASSES & IDs:
Full Customization

CLASS & ID SELECTORS: ANALOGIES

Classes call up *the same type of thing*:

KIDS

CATS

BIKES

IDs call up one, specific thing:

STACEY

MARC

HEATHER

images: Generated with Adobe Firefly

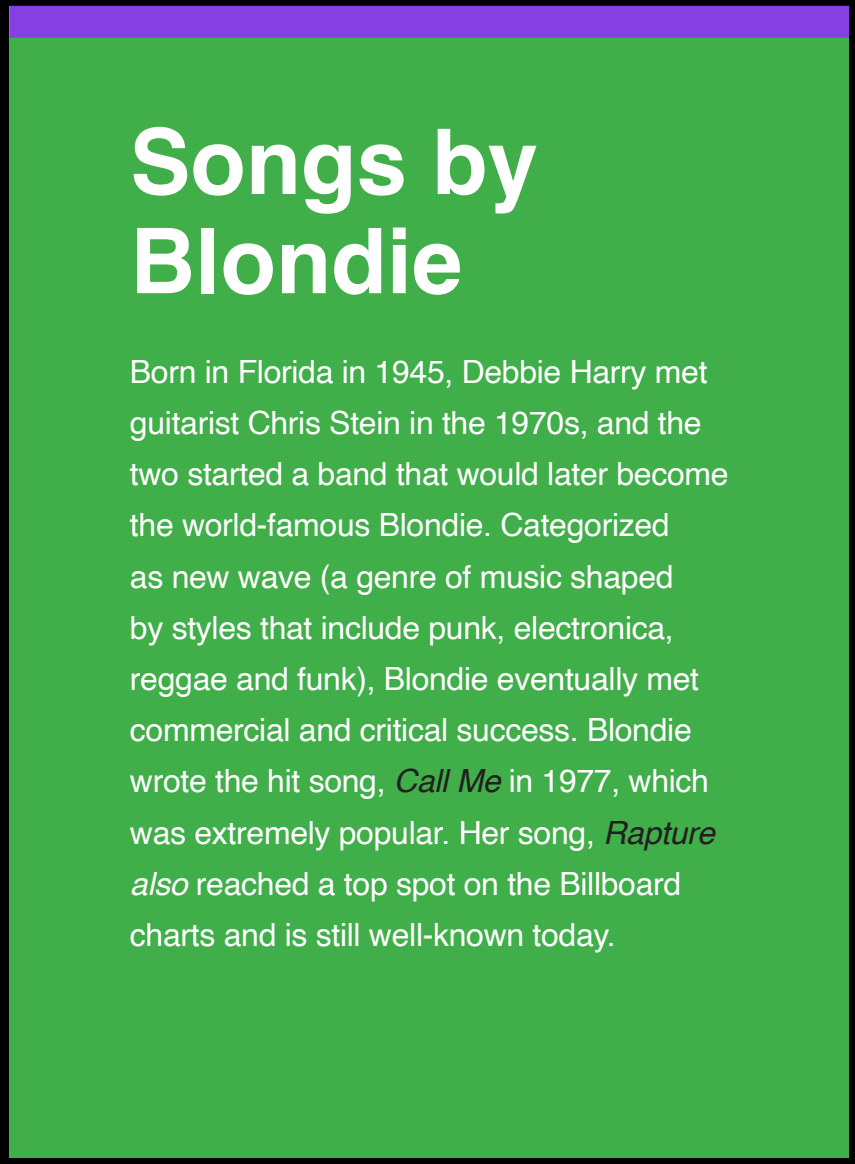
CUSTOM CONTROL: CLASS & ID SELECTORS

HTML

```
<div id="BlondieProfile">
  <h2> Songs by Blondie </h2>
  <p>...Blondie wrote the hit song,
     <span class="SongTitle"> Call Me </span>
     in 1977, which was extremely... </p>
</div>
```

CSS

```
#BlondieProfile {
     width: 100%;
     background-color: green;
     border-top: 10px solid purple;   }

h2 {  font-family: Helvetica;
     font-size: 2em; }

p {   line-height: 1.5em;  }

.SongTitle {
     font-style: italic;  }
     color: black;  }
```



CUSTOM CONTROL: CLASS & ID SELECTORS

HTML

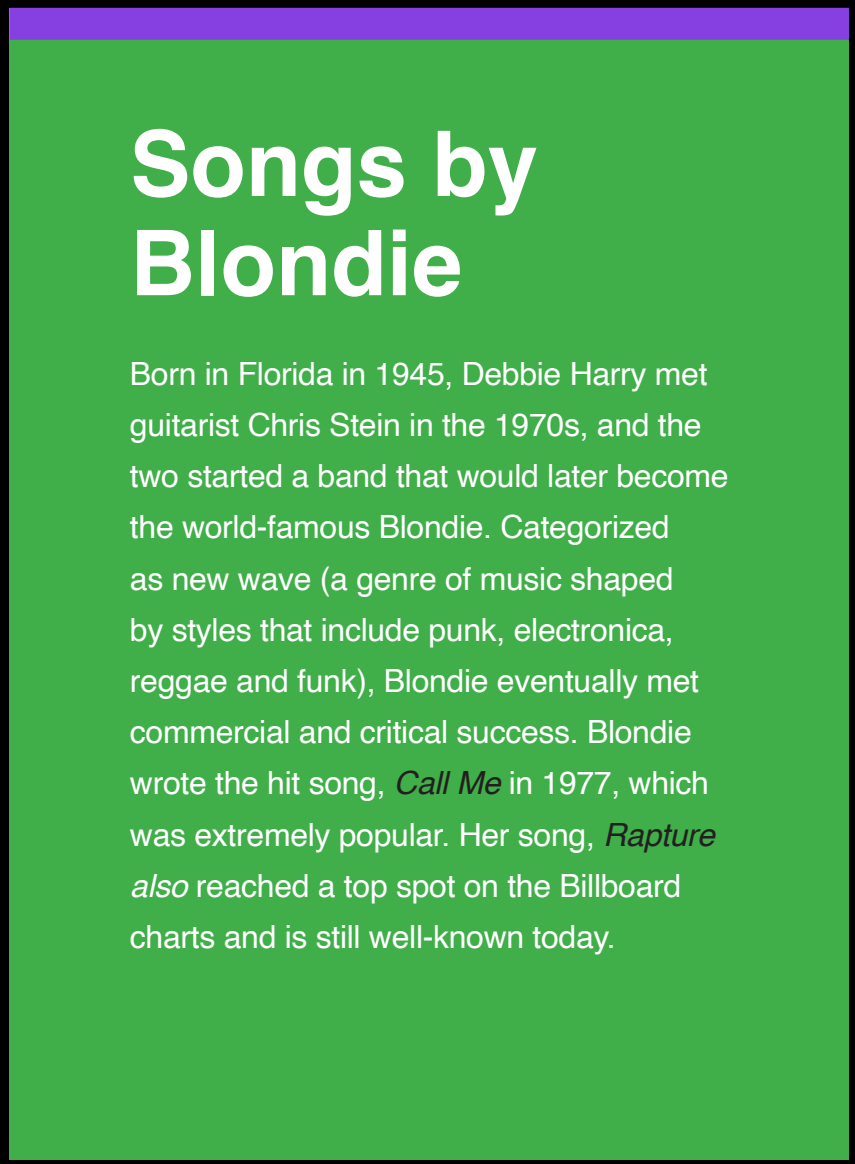
```
<div id="BlondieProfile">
  <h2> Songs by Blondie </h2>
  <p> ...Blondie wrote the hit song,
     <span class="SongTitle"> Call Me </span>
     in 1977, which was extremely... </p>
</div>
```

CSS

```
#BlondieProfile {
     width: 100%;
     background-color: green;
     border-top: 10px solid purple;   }

h2 {  font-family: Helvetica;
     font-size: 2em; }

p {   line-height: 1.5em;  }

.SongTitle {
     font-style: italic;  }
     color: black;  }
```

[hashtag = id]

[dot = class]

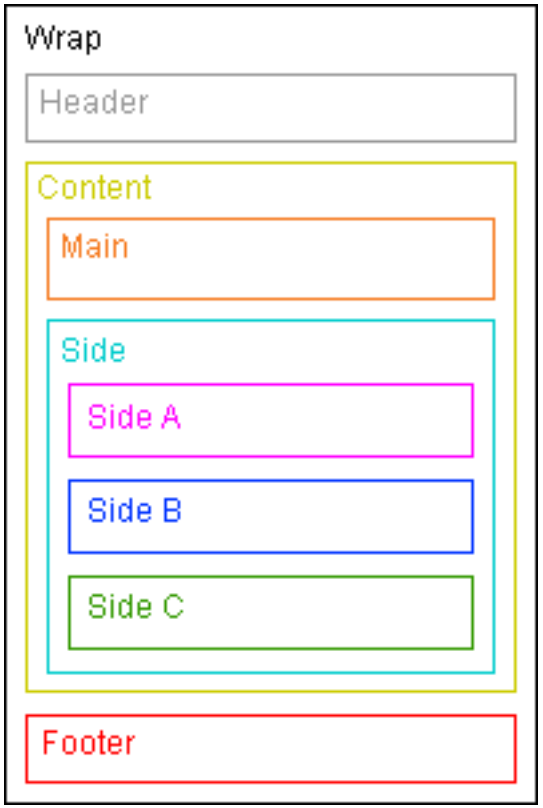


DIV STRUCTURE USING IDS

THIS HTML CODE:

```
<div id="Wrap">
   <div id="Header">
   </div> <!-- closes Header-->
   <div id="Content">
      <div id="Main">
      </div> <!-- closes Main -->
      <div id="Side">
         <div id="Side_A">
         </div> <!-- closes Side A -->
         <div id="Side_B">
         </div> <!-- closes Side B-->
         <div id="Side_C">
         </div> <!-- closes Side C-->
      </div> <!-- closes Side -->
   </div> <!-- closes Content -->
   <div id="footer">
   </div> <!-- closes footer -->
</div>  <!-- closes wrap-->
```

DISPLAYS LIKE THIS:



DIVS HOLD CHUNKS TOGETHER LIKE:

GIFT BASKET CELLOPHANE:

NESTING DOLLS:

http://matthewjamestaylor.com/demos/ipad-css-layout/

XHTML & HTML5 MARKUP STYLES

XHTML

HTML

```
<div id="header"> [content here] </div>
<div id="nav"> [content here] </div>
<div id="section"> [content here] </div>
<div id="article"> [content here] </div>
<div id="footer"> [content here] </div>
```

CSS

```
#header {
          padding: 1em; }
#nav {
          padding: 1em; }
#section {
          padding: 1em; }
#article {
          padding: 1em; }
#footer {
          padding: 1em; }
```

HTML5

HTML

```
<header> [content here] </header>
<nav> [content here] </nav>
<section> [content here] </section>
<article> [content here] </article>
<footer> [content here] </footer>
```

CSS

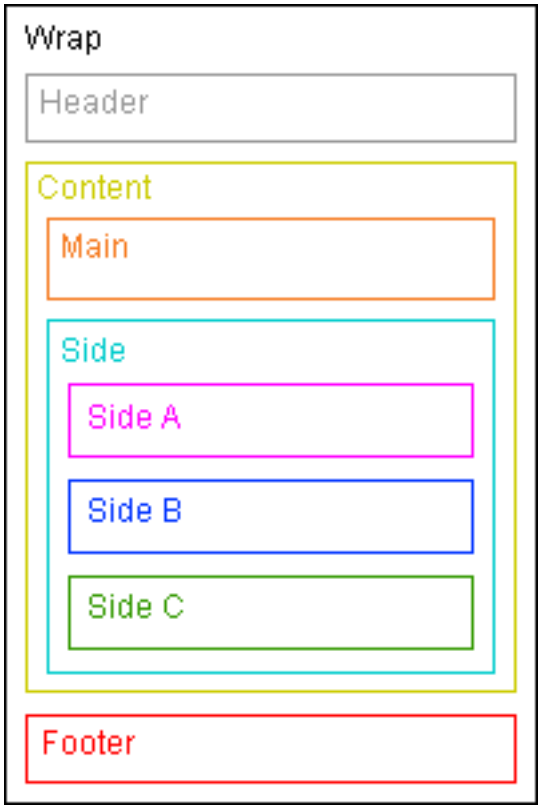
```
header {
          padding: 1em; }
nav {
          padding: 1em; }
section {
          padding: 1em; }
article {
          padding: 1em; }
footer {
          padding: 1em; }
```

DIV STRUCTURE USING HTML5

THIS HTML CODE:

```
<div id="Wrap">
   <header> </header>
   <div id="Content">
      <section> </section>
      <aside>
         <div id="Aside_A">
         </div><!--closes Aside A-->
         <div id="Aside_B">
         </div> <!-- closes Aside B-->
         <div id="Aside_C">
         </div> <!-- closes Aside C-->
      </aside>
   </div> <!-- closes Content -->
   <footer> </footer>
</div>  <!-- closes wrap-->
```

DISPLAYS LIKE THIS:



DIVS HOLD CHUNKS TOGETHER LIKE:

GIFT BASKET CELLOPHANE:

NESTING DOLLS:

edited from http://matthewjamestaylor.com/demos/ipad-css-layout/

Positioning

NORMAL FLOW

NORMAL FLOW

– default behavior / appearance of html

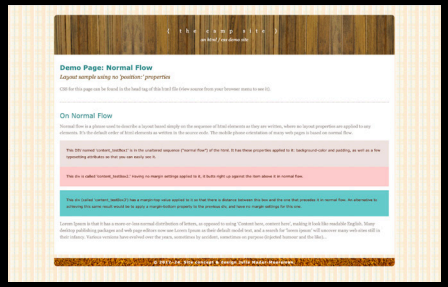
[logo + navigation]

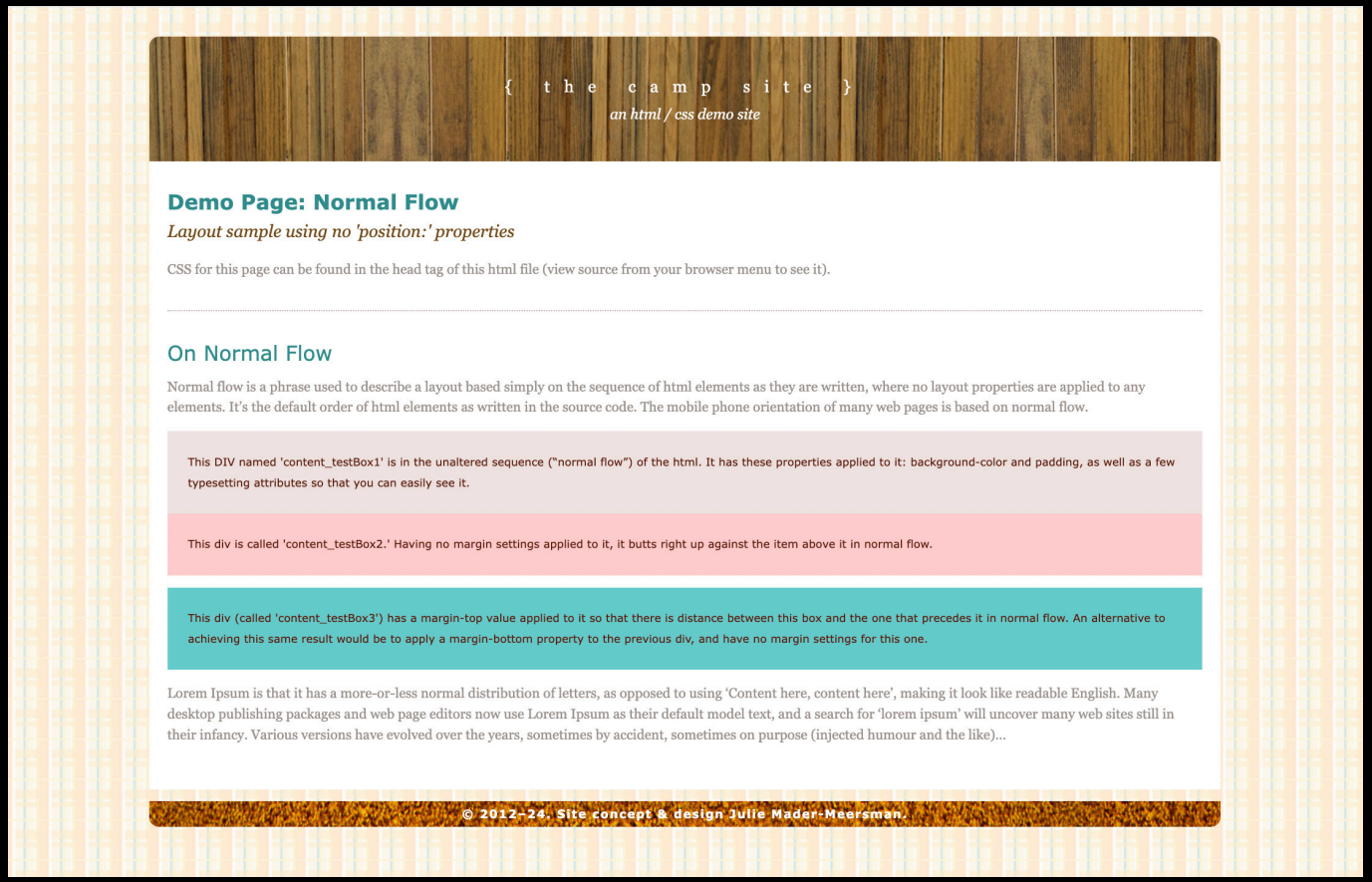
Welcome banking customer

Hello… Griffin G. can help you with your banking inquiry today.

[credits + contact info]

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.

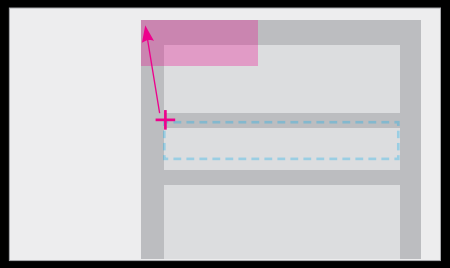




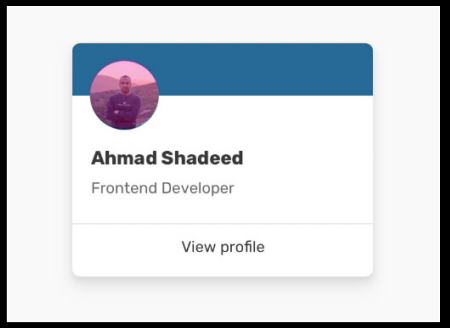
CSS POSITIONING: PAST PREFERENCES

ABSOLUTE

`position: absolute;`

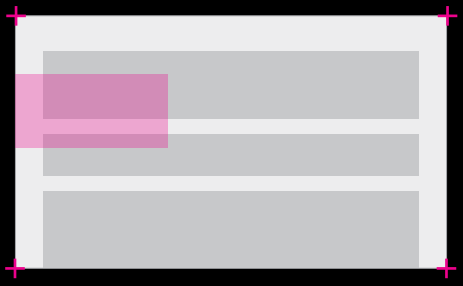


Removes an item from normal flow and positions it in relation to its containing element. Surrounding elements ignore the space that it would have occupied.

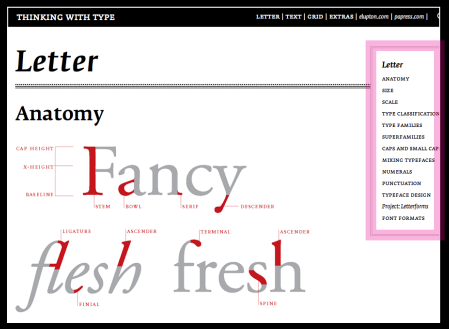


FIXED

`position: fixed;`

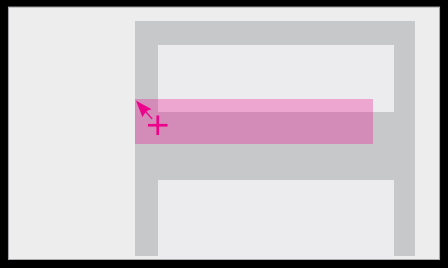


A form of absolute positioning that fixes the element in relation to the browser edges itself. These don't affect other elements' positions and don't move when the user scrolls up/down the page. Always stays.

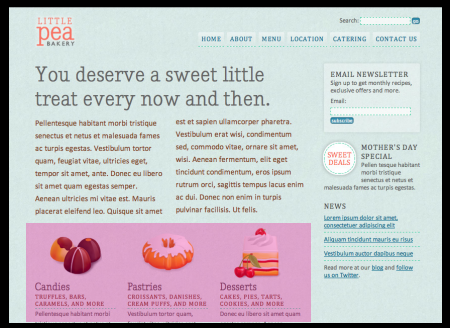


RELATIVE

`position: relative;`

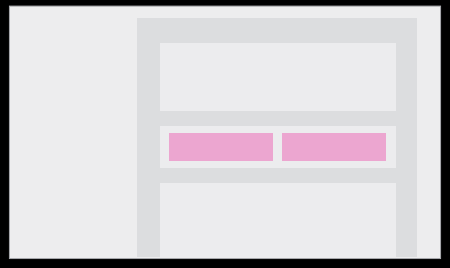


Offests an element relative to where it would have been in normal flow. Surrounding elements stay in their normal flow. Children of relative elements can be positioned with 'float' and 'clear' and can have a z-index.

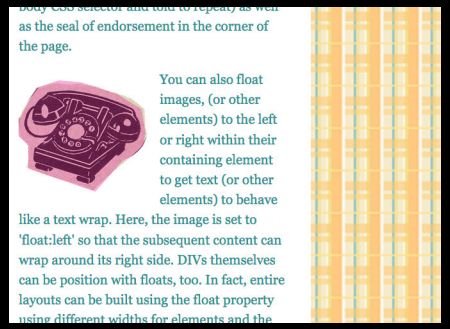


FLOAT

`float:left;` or `float:right;`



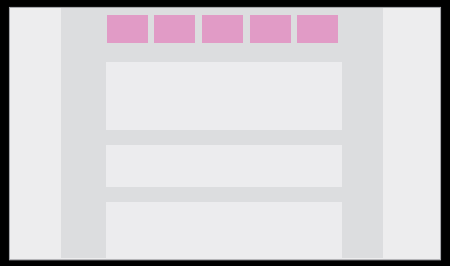
Allows elements to remain part of the flow of a page while appearing side-by-side. Functions as a form of text wrapping. Used to be used for entire layouts.



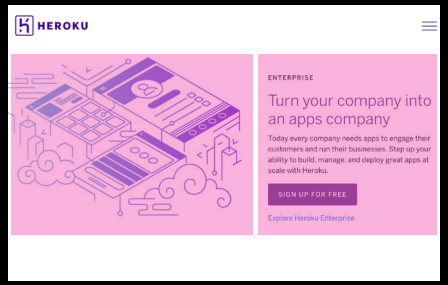
CSS POSITIONING: MODERN STANDARDS

FLEXBOX

```
display: flex;
```

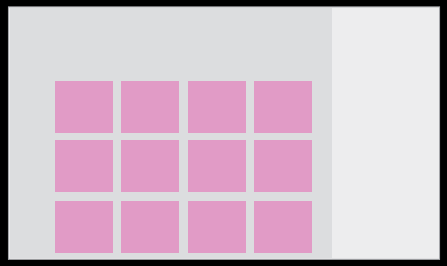


One-dimensional layout control; that is, good for organization of elements either horizontally or vertically. A good alternative to floats.

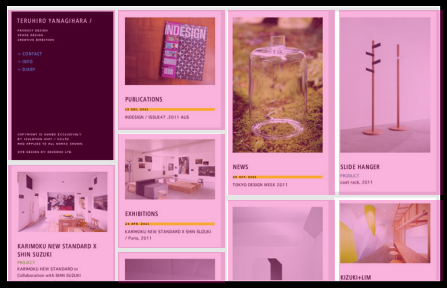


CSS GRID / "GRID"

```
display: grid;
```



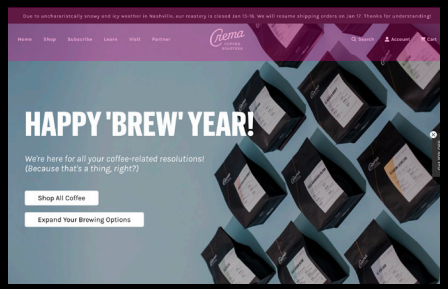
*Two-dimensional layout control that enables the horizontal *and* vertical organization of elements across a true grid (use of columns and rows). Enormous layout potential with much less code than other methods.*



STICKY

```
position: sticky;
```

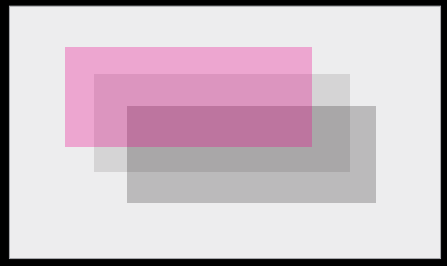
A combination of relative and fixed positioning: the element behaves as relative (scrolls with the content) until it reaches a specified threshold, when it then acts as fixed ("sticks" to the browser).



Z-INDEX

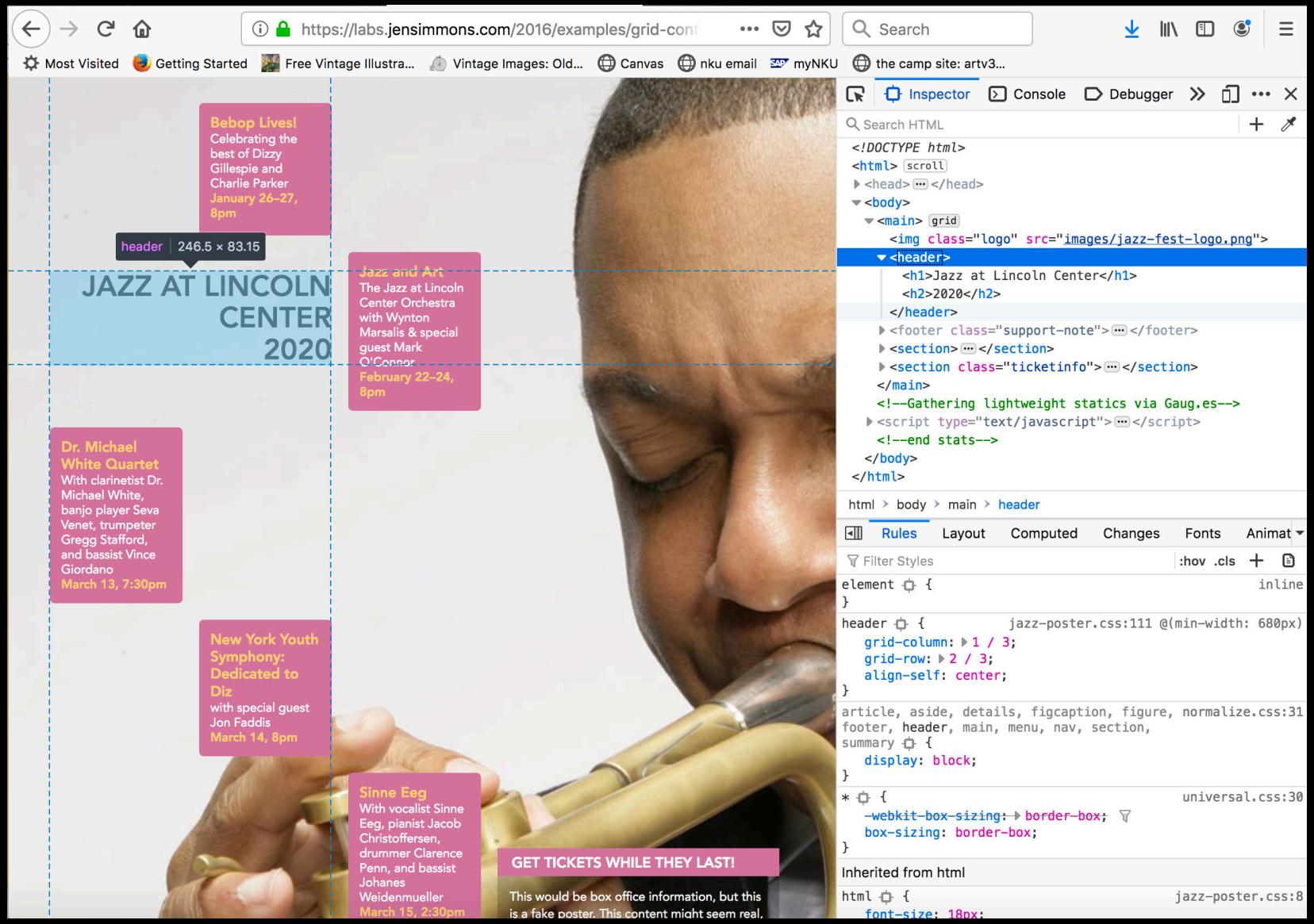
```
z-index: 50;
```

Set parent to position: relative;



The place of an element in the page depth. Allows control of layering. All elements exist in the order they are written in html, unless otherwise specified by the z-index value. Higher numbers are close to the front of the screen.

VIEW SOURCE CODE +
INSPECT ELEMENT



POSITIONING OPTIONS DEMONSTRATED