





**FILE MANAGEMENT:**

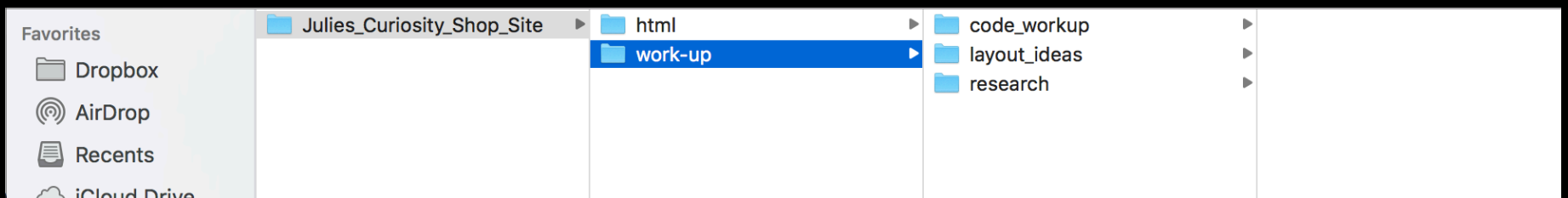
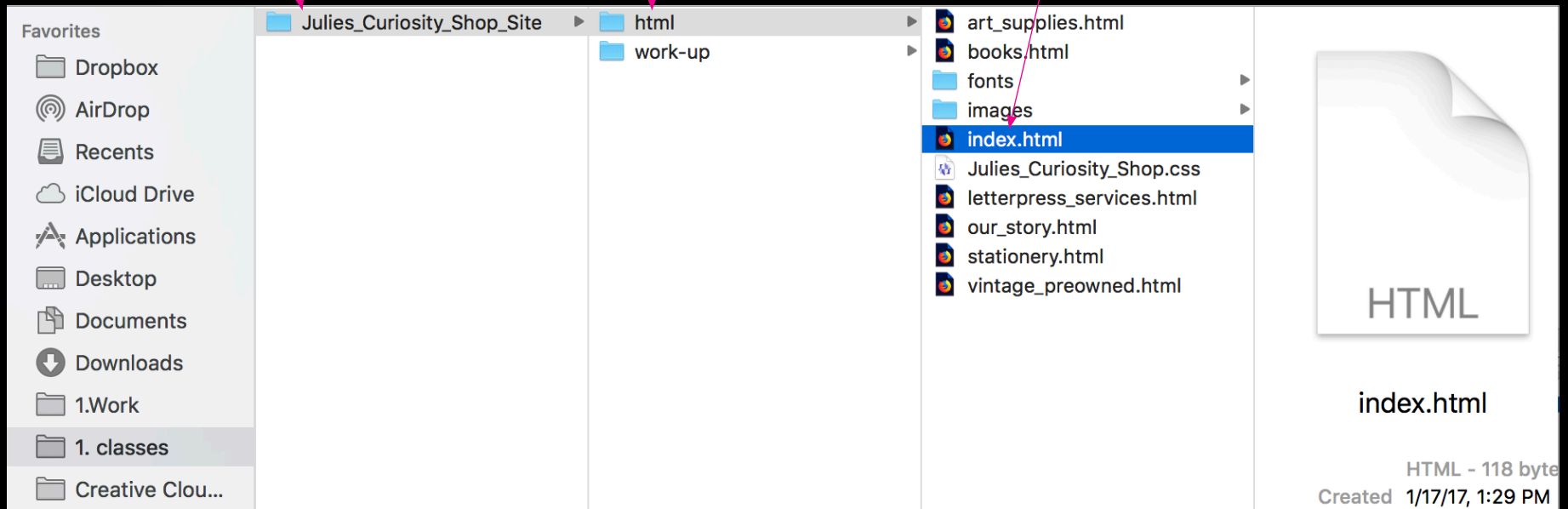
*Site Directories*

# FOLDER + FILE STRUCTURE

**PROJECT FOLDER**  
(ACTUAL NAME)

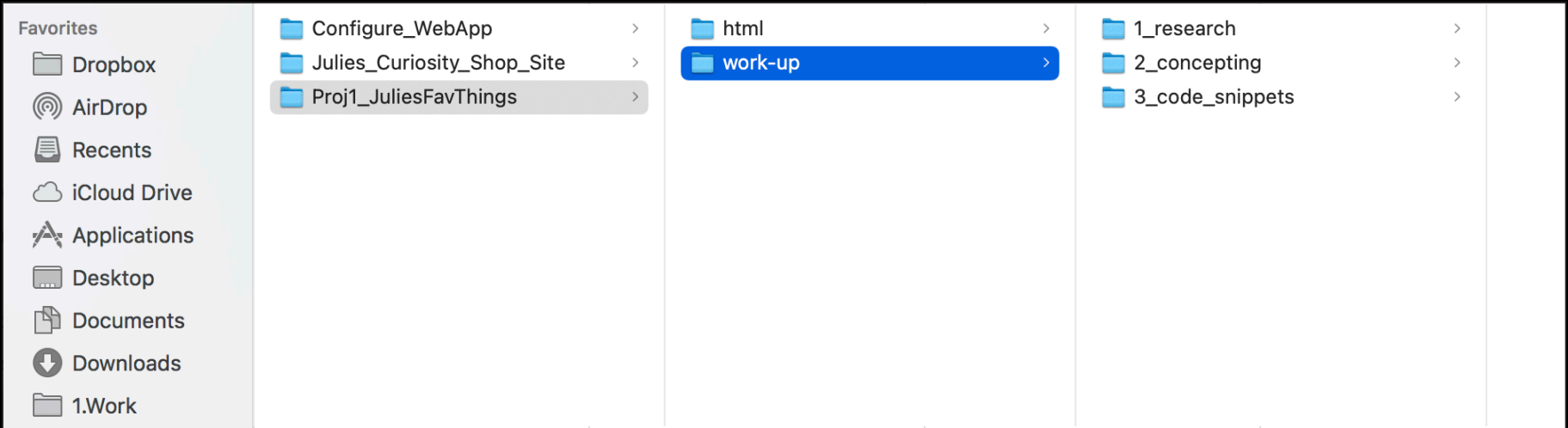
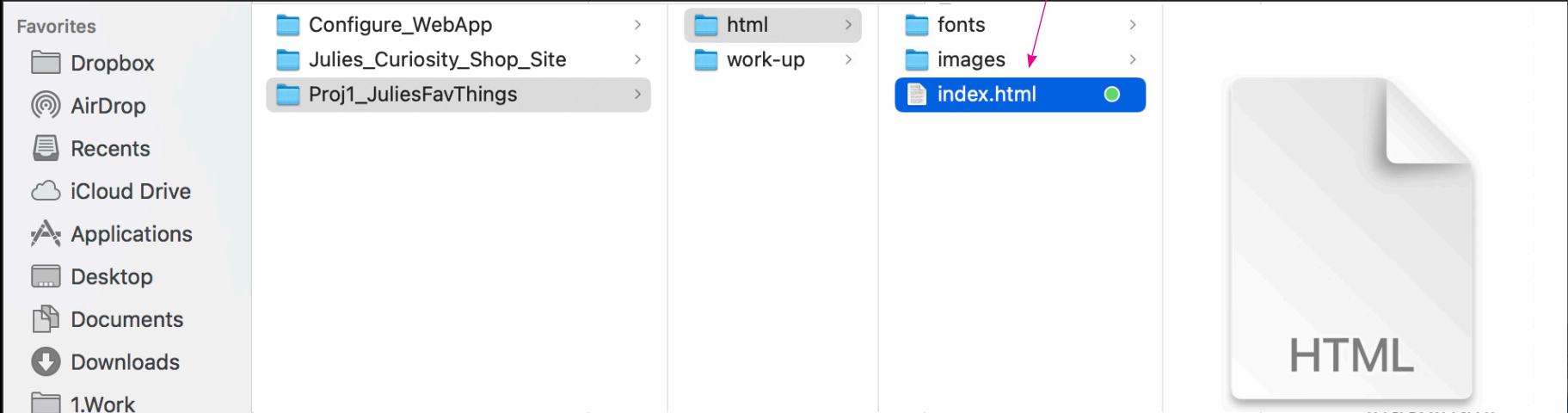
**ROOT FOLDER**  
(ALWAYS "HTML")

**PAGE 1**  
(ALWAYS "INDEX.HTML")



SINGLE-PAGER: FOLDER  
+ FILE STRUCTURE

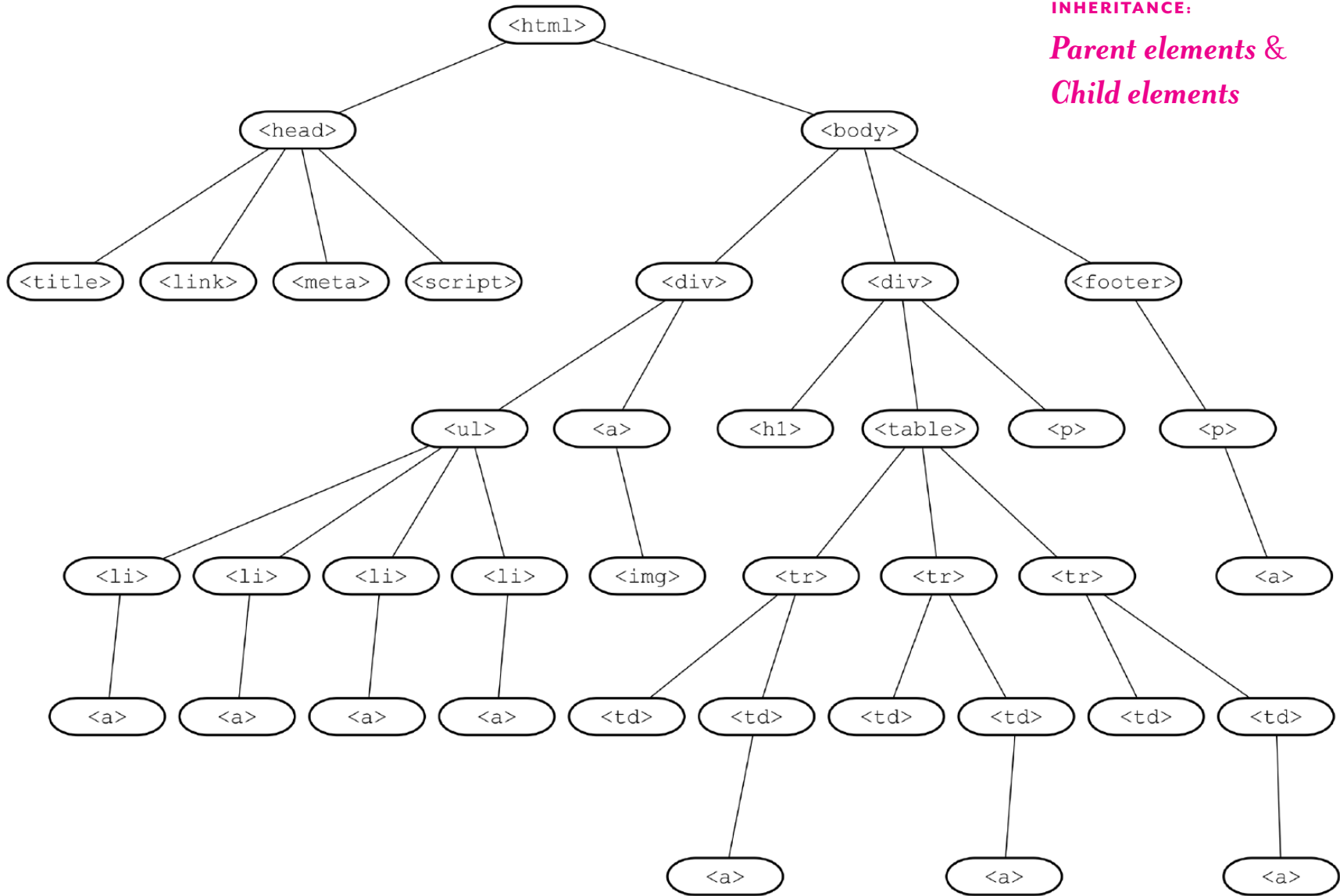
PAGE 1  
(ALWAYS "INDEX.HTML")



# HTML & CSS: *Layout Basics*

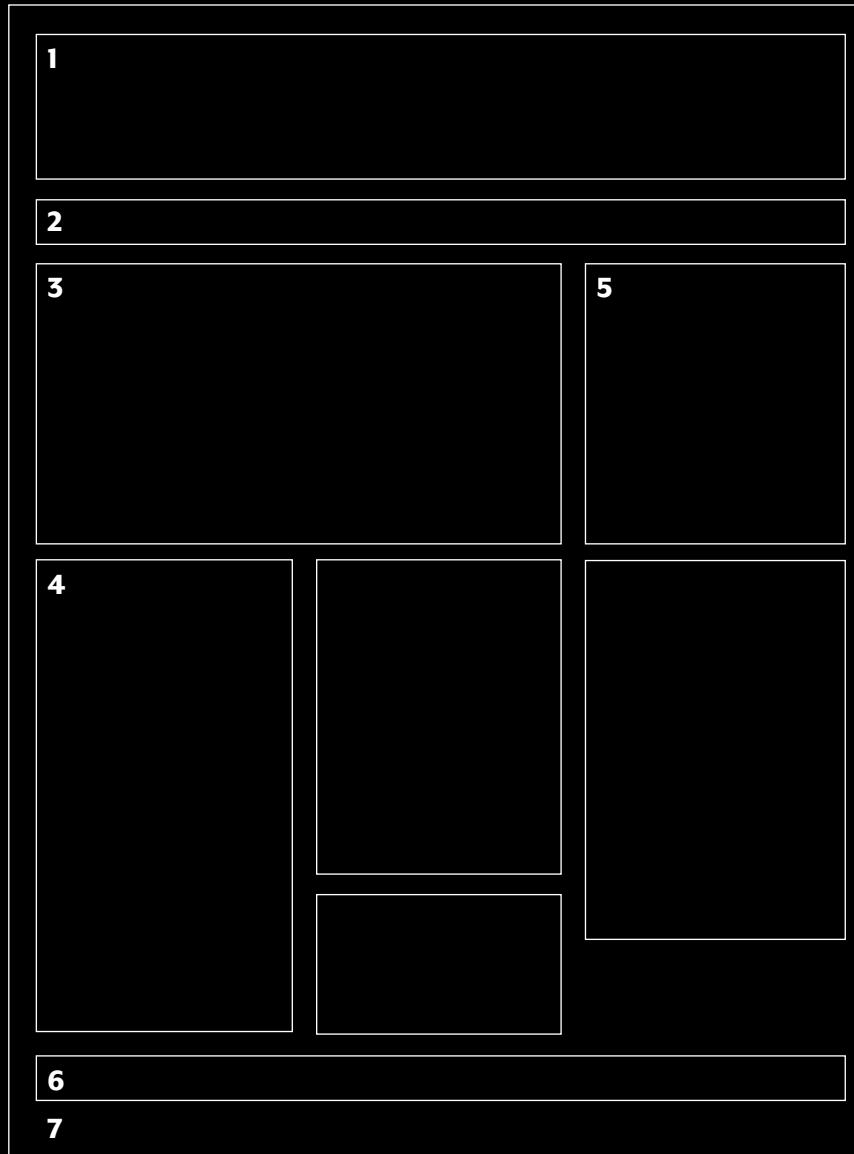
- 1: *Browser itself*
- 2: *Author/designer*
- 3: *User*

# CONDITIONS: THE CASCADE CONCEPT



**INHERITANCE:**  
*Parent elements &  
Child elements*

## ANATOMY OF A WEB PAGE

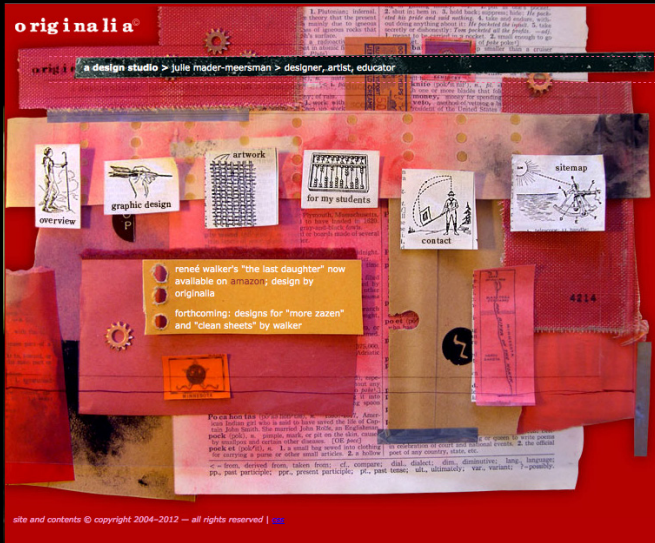


1. *Header*
2. *Navigation*
3. *Feature*
4. *Body/Content*
5. *Sidebar*
6. *Footer*
7. *Background*

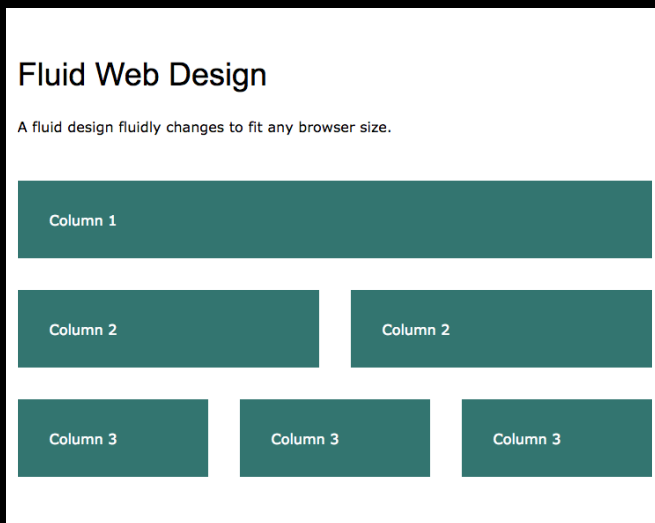


# APPROACHES TO WEB LAYOUT

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

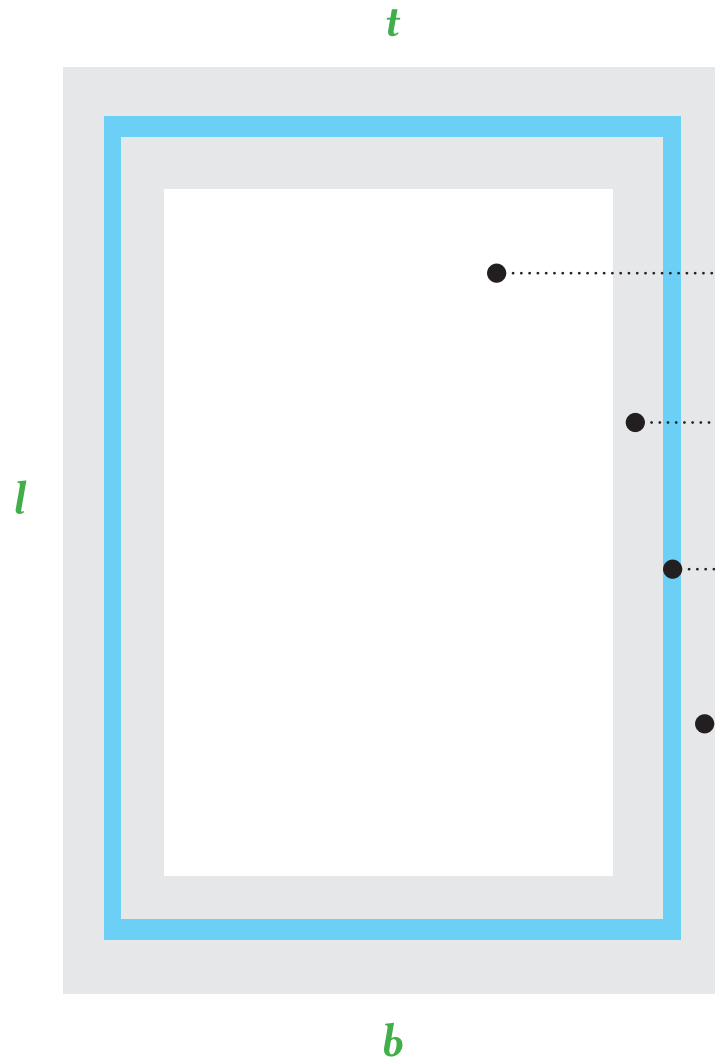


4) **INTRINSIC OR RWD "PLUS"**  
columns and rows w  
flexbox and css grid;  
sizing uses fr units;  
media queries as  
needed



*“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

INTRO



0:01 / 1:54



The box model for beginners web design tutorial



# BLOCK-LEVEL AND INLINE ELEMENTS

## 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 . <u>hyperlink</u> . 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:

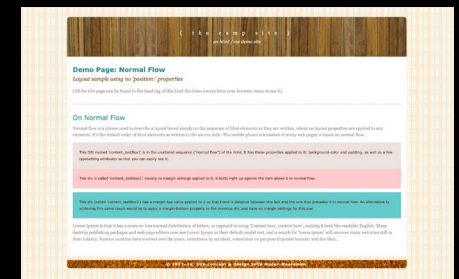
links  
image tags  
span tags

## NORMAL FLOW

– default behavior /  
appearance of html

[logo + navigation]
<i>Welcome banking customer</i>
Hello... <u>Griffin G.</u> 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.



**HTML**



# THE MEGA CHEAT SHEET

**TAGS - EVENT ATTRIBUTES -  
MOBILE - BROWSERS - CANVAS**

by **Jamie Spencer** @

make a  **websitehub.com**

**CSS**



# The Ultimate CSS CHEAT SHEET

[Including CSS3 Tags]

With billions of websites and counting, yours needs to stand out. CSS is the style sheet language used to make your website one-in-a-billion. Ready to learn and utilize it? Here's a cheat sheet to help you remember all the different elements at your disposal.

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





## 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; }
```

# Songs by Blondie

Born in Florida in 1945, Debbie Harry met guitarist Chris Stein in the 1970s, and the two started a band that would later become the world-famous Blondie. Categorized as new wave (a genre of music shaped by styles that include punk, electronica, reggae and funk), Blondie eventually met commercial and critical success. Blondie wrote the hit song, *Call Me* in 1977, which was extremely popular. Her song, *Rapture* also reached a top spot on the Billboard charts and is still well-known today.

## 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 {           [hashtag = id]
  width: 100%;
  background-color: green;
  border-top: 10px solid purple; }
```

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

```
p { line-height: 1.5em; }
```

```
→ .SongTitle {                [dot = class]
  font-style: italic; }
  color: black; }
```

# Songs by Blondie

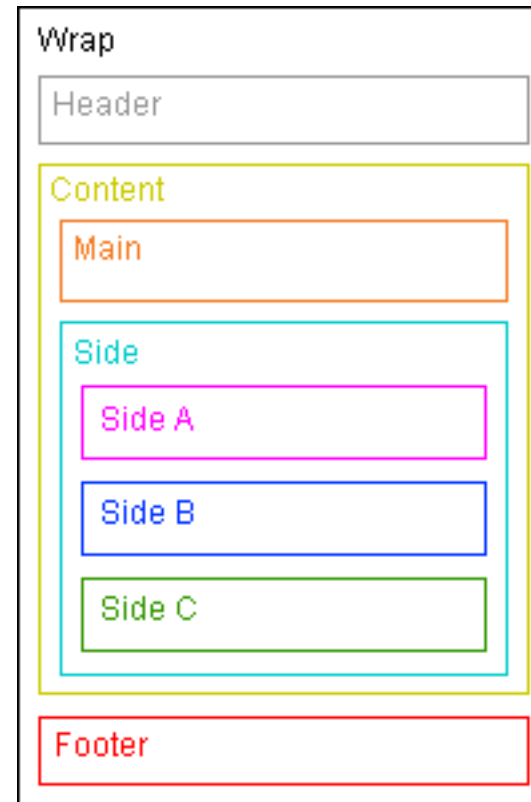
Born in Florida in 1945, Debbie Harry met guitarist Chris Stein in the 1970s, and the two started a band that would later become the world-famous Blondie. Categorized as new wave (a genre of music shaped by styles that include punk, electronica, reggae and funk), Blondie eventually met commercial and critical success. Blondie wrote the hit song, *Call Me* in 1977, which was extremely popular. Her song, *Rapture* also reached a top spot on the Billboard charts and is still well-known today.

## 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:



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; }
```

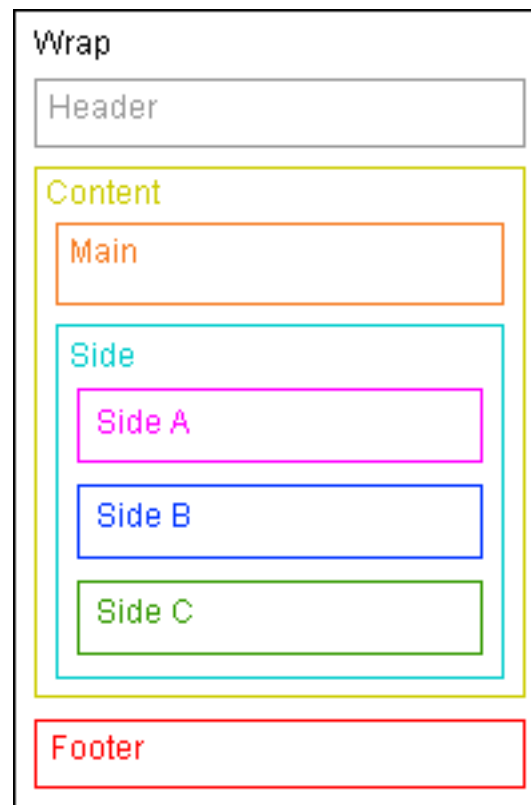


# DIV STRUCTURE USING HTML5

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:



*Positioning*

# NORMAL FLOW

## NORMAL FLOW

- default behavior /  
appearance of html

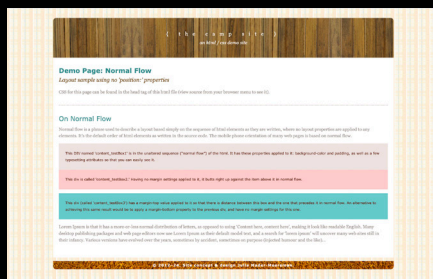
[logo + navigation]

*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.*



{ the camp site }  
an html / css demo site

### Demo Page: Normal Flow

Layout sample using no 'position:' properties

CSS for this page can be found in the head tag of this html file (view source from your browser menu to see it).

---

### On Normal Flow

Normal flow is a phrase used to describe a layout based simply on the sequence of html elements as they are written, where no layout properties are applied to any elements. It's the default order of html elements as written in the source code. The mobile phone orientation of many web pages is based on normal flow.

This DIV named 'content\_testBox1' is in the unaltered sequence ("normal flow") of the html. It has these properties applied to it: background-color and padding, as well as a few typesetting attributes so that you can easily see it.

This div is called 'content\_testBox2.' Having no margin settings applied to it, it butts right up against the item above it in normal flow.

This div (called 'content\_testBox3') has a margin-top value applied to it so that there is distance between this box and the one that precedes it in normal flow. An alternative to achieving this same result would be to apply a margin-bottom property to the previous div, and have no margin settings for this one.

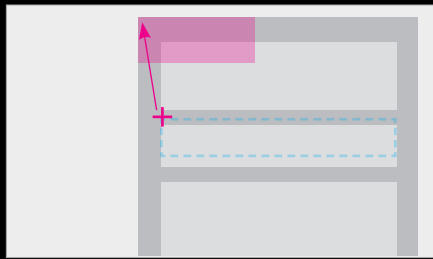
Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like)...

© 2012-24. Site concept & design Julie Mader-Meersman.

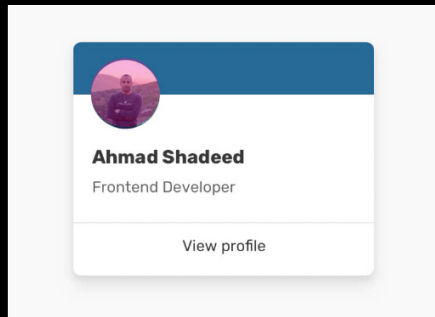
# CSS POSITIONING: PAST PREFERENCES

## ABSOLUTE

`position: absolute;`

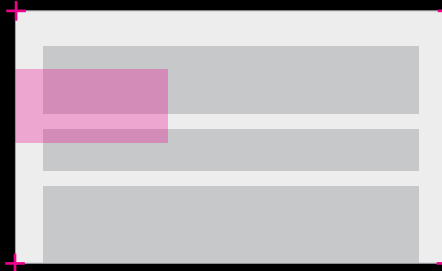


Removes an item from normal flow and positions it in relation to its containing element. Surrounding elements ignore the space that it would have occupied.



## FIXED

`position: fixed;`

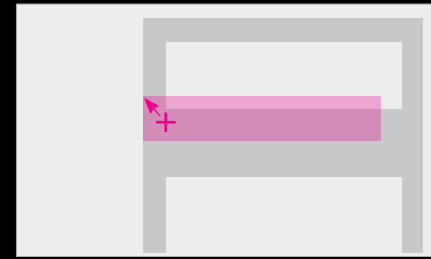


A form of absolute positioning that fixes the element in relation to the browser edges itself. These don't affect other elements' positions and don't move when the user scrolls up/down the page. Always stays.

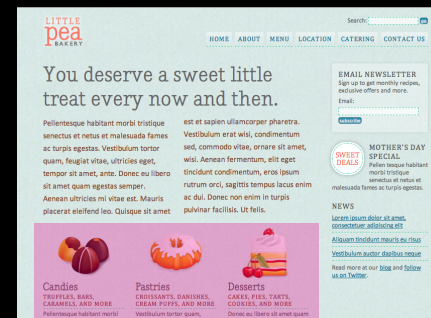


## RELATIVE

`position: relative;`

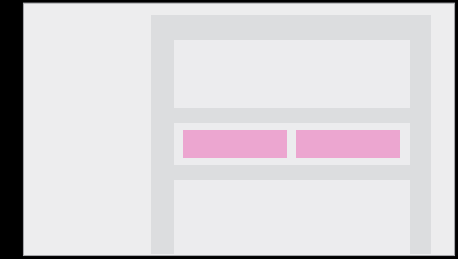


Offsets an element relative to where it would have been in normal flow. Surrounding elements stay in their normal flow. Children of relative elements can be positioned with 'float' and 'clear' and can have a z-index.

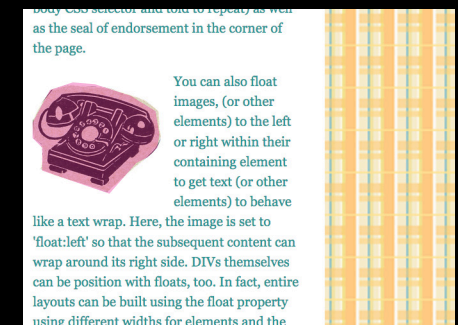


## FLOAT

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



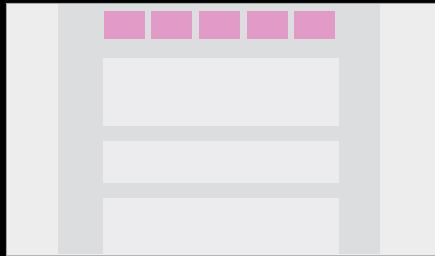
Allows elements to remain part of the flow of a page while appearing side-by-side. Functions as a form of text wrapping. Used to be used for entire layouts.



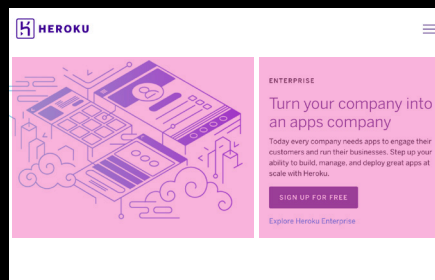
# CSS POSITIONING: MODERN STANDARDS

## FLEXBOX

`display: flex;`



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

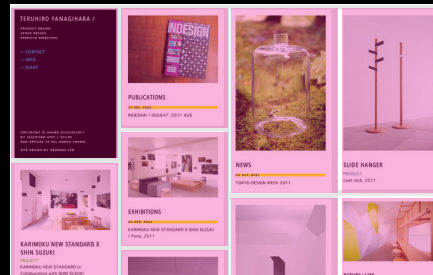


## CSS GRID / "GRID"

`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.*

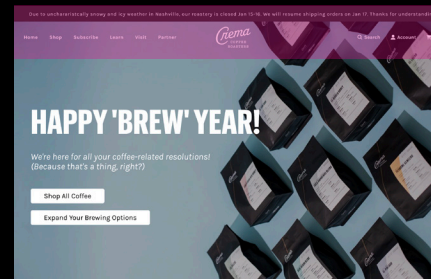


## STICKY

`position: sticky;`



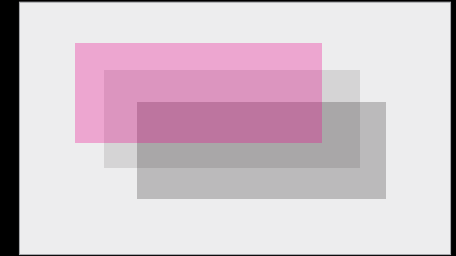
*A combination of relative and fixed positioning: the element behaves as relative (scrolls with the content) until it reaches a specified threshold, when it then acts as fixed ("sticks" to the browser).*



## 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

The screenshot shows a web browser displaying a jazz poster for Lincoln Center 2020. The poster features a large image of a man playing a trumpet. Overlaid on the poster is a grid layout with several text boxes containing event information:

- Bebop Lives!** Celebrating the best of Dizzy Gillespie and Charlie Parker. January 26-27, 8pm.
- Jazz and Art** The Jazz at Lincoln Center Orchestra with Wynton Marsalis & special guest Mark O'Connor. February 22-24, 8pm.
- Dr. Michael White Quartet** With clarinetist Dr. Michael White, banjo player Seva Venet, trumpeter Gregg Stafford, and bassist Vince Giordano. March 13, 7:30pm.
- New York Youth Symphony: Dedicated to Diz** with special guest Jon Faddis. March 14, 8pm.
- Sinne Eeg** With vocalist Sinne Eeg, pianist Jacob Christoffersen, drummer Clarence Penn, and bassist Johanes Weidenmueller. March 15, 2:30pm.

The main title of the poster is "JAZZ AT LINCOLN CENTER 2020". At the bottom, there is a call to action: "GET TICKETS WHILE THEY LAST!" followed by a note: "This would be box office information, but this is a fake poster. This content might seem real."

The browser's developer tools are open, showing the HTML structure. The selected element is the header, which contains the following code:

```
<h1>Jazz at Lincoln Center</h1>
<h2>2020</h2>
</header>
```

The CSS rules for the header are:

```
header {
  grid-column: 1 / 3;
  grid-row: 2 / 3;
  align-self: center;
}
```



POSITIONING OPTIONS  
DEMONSTRATED

{ the camp site }

*an html / css demo site*

{ base camp }

{ basic code syntax }

{ layout }

{ text and links }

{ images }

{ responsive web }

{ navigation }

{ css3 }

{ resources }

