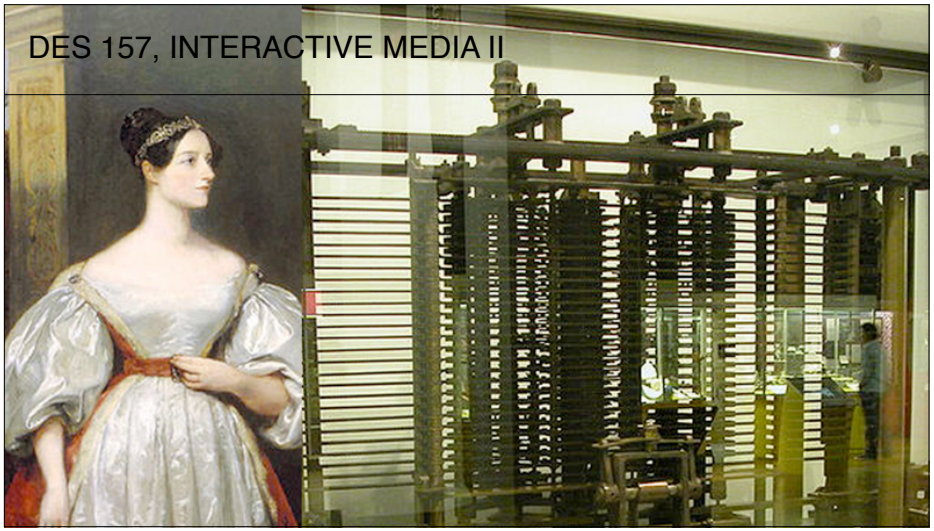


Gina Bloom
Associate Professor of English, Univ. of California, Davis [website]
Project Director, *Play the Knave*, a video game about Shakespeare performance

** Follow us on Twitter @PlayTheKnave **

Game Design Intern
We are searching for an intern interested in graphic and interactive design to beautify the interface for a videogame called *Play the Knave* (http://playtheknave.org). This is a motion capture game in which users customize options and then perform in scenes from Shakespeare's plays. Requirements: creativity, good design sensibilities, and experience with html, javascript, and css.

The intern would be working with faculty, graduate students, and undergraduates from a range of fields (including English, Computer Science, Theater, and Sociology) in the ModLab (https://modlab.ucdavis.edu/). The student would receive course credit units or Transcript Notation through the Career Center. This is a great opportunity for someone interested in building a portfolio for a career in game design, graphic design, or digital interface design.

today

intro, syllabus, review html5/css3

surveys

st 1: any portal in a storm

thursday

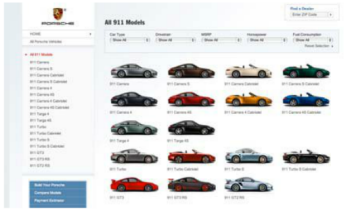
st 1: sketches and html/css due at the start of class

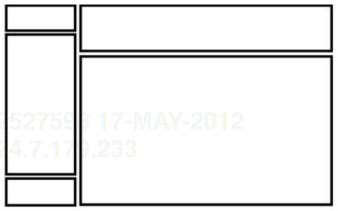
sign up for github before class

always bring sketchbooks to class

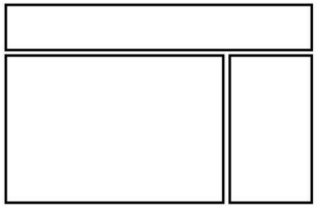
interface design current trends

left-column nav

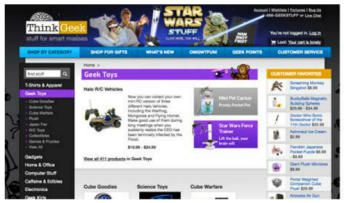


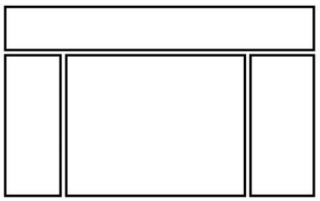


right-column nav



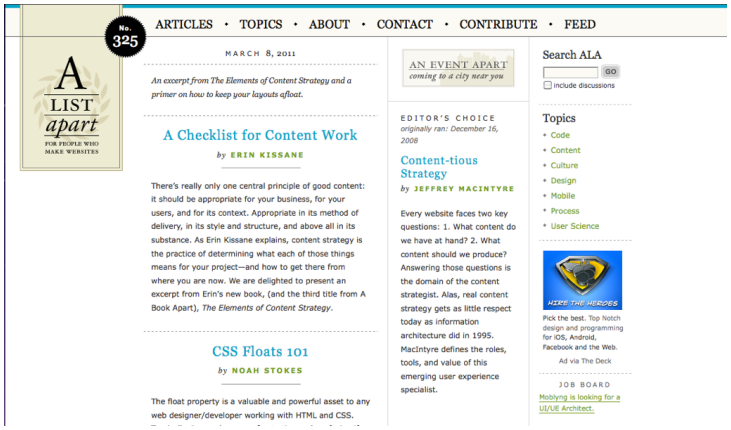
three-column nav



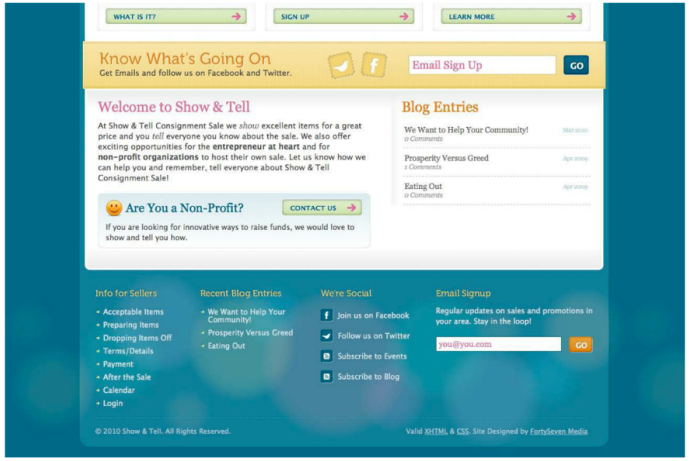


magazine style

above the fold

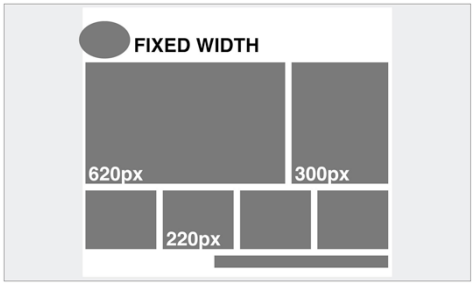


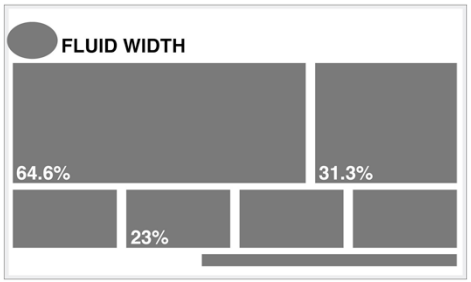
expansive footers

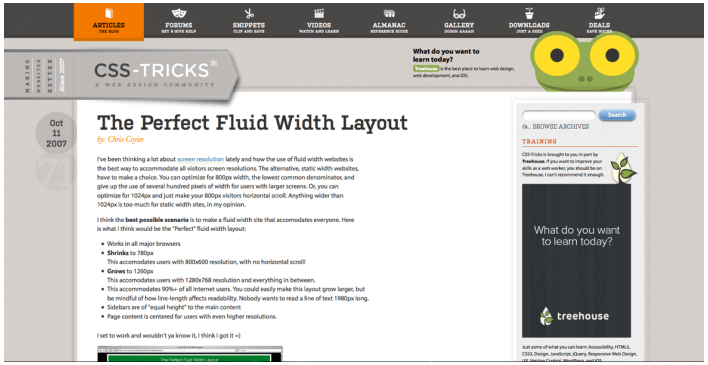


parallax

http://johnpolacek.github.io/scrollorama/hyperlink



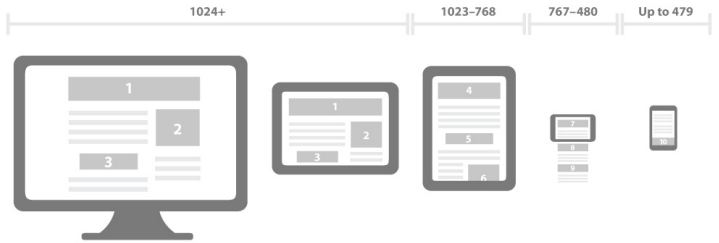




http://css-tricks.com/the-perfect-fluid-width-layout/

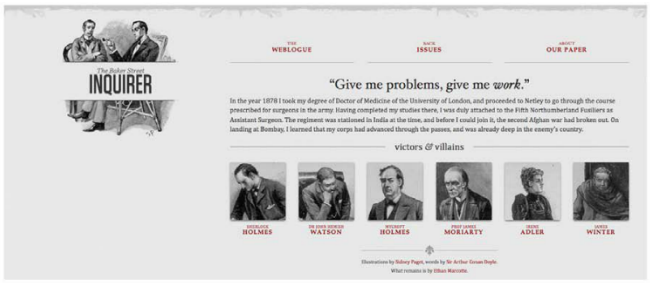
	Pros	Cons
Fixed width	■ gives designer more control over how an image floated within the content will look ■ allows for planned whitespace ■ improves readability with narrower text blocks	■ can appear dwarfed in large browser windows ■ takes control away from the user
Liquid width	■ adapts to most screen resolutions and devices ■ reduces user scrolling	■ challenging to read when text is spanning a wide distance ■ harder to execute successfully ■ limits or imposes on whitespace

A new approach...Responsive design



responsive design: mobile and desktop

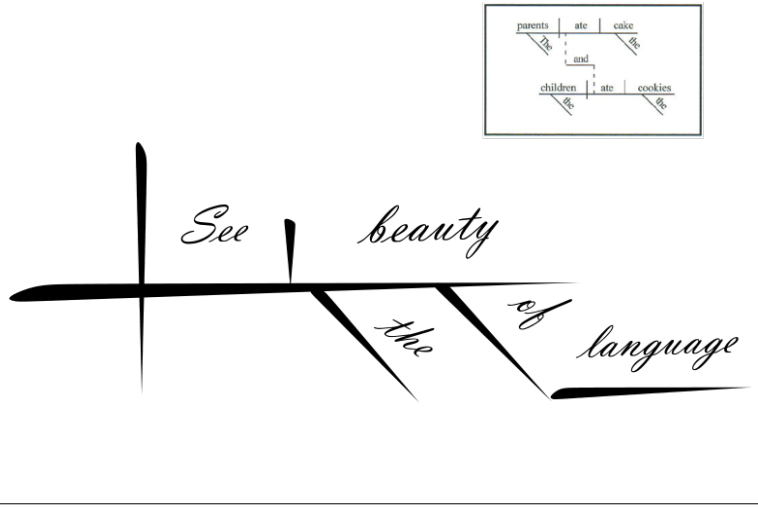
responsive design: jumbotron viewport



the following is a combination of Patrice Ehlert's slides (made in communication with David Hurwich and Bill Mead and me) with my own

overview html

semantic markup / web standards



naming files

no spaces

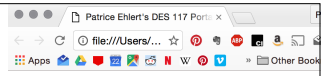
all lower case

dashes or camelCase to separate words

index.html is always the filename of the first page

example

```
<h1>United States History</h1>
    <p>Some introductory text</p>

    <h2>American Revolution</h2>
        <p>Some paragraphs about this topic</p>

    <h2>Early Years of the Republic</h2>
        <p>Some paragraphs about this topic</p>

    <h2>The 19th Century</h2>
        <p>Some paragraphs about this topic</p>

        <h3>Civil War</h3>
            <p>Some paragraphs about this topic</p>

        <h2>The 20th Century</h2>
            <p>Some paragraphs about this topic</p>
```



United States History

Some introductory text

American Revolution

Some paragraphs about this topic

Early Years of the Republic

Some paragraphs about this topic

The 19th Century

Some paragraphs about this topic

Civil War

Some paragraphs about this topic

The 20th Century

Some paragraphs about this topic

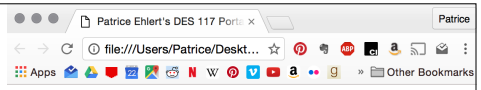
types of html tags

structural

phrasing

example

```
<p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Donec finibus lorem non nulla porttitor,
vitae fringilla nunc fringilla. Sed tincidunt pharetra
commodo.</p>

<p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Donec finibus lorem non nulla porttitor,
<strong>vitae fringilla</strong> nunc fringilla. Sed
tincidunt pharetra commodo.</p>
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec finibus lorem non nulla porttitor, vitae fringilla nunc fringilla. Sed tincidunt pharetra commodo.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec finibus lorem non nulla porttitor, **vitae fringilla** nunc fringilla. Sed tincidunt pharetra commodo.

structural tags

<header>
<main>
<section>
<article>
<aside>
<nav>
<footer>

<h1–h6>
<p>

<blockquote>

phrasing tags

<a>
<img>

closing tags

```
<head>
  <title>Hello world</title>
  <meta charset="utf-8">
</head>
```

self-closing tags

```
<body>
  <img src="images/banner.png" height="200" width="800" alt="gradient">
</body>
```

attributes and values

<element attributename="attributevalue”>

<img src="images/banner.png" height="200" width="800” alt=”banner”>

nesting

```
<body>
  <img src="images/banner.png" height="200" width="800" alt="gradient">
  <p>Hello world!</p>
</body>
```

<head>

not visible to users

define title, load style sheets, load JavaScript files

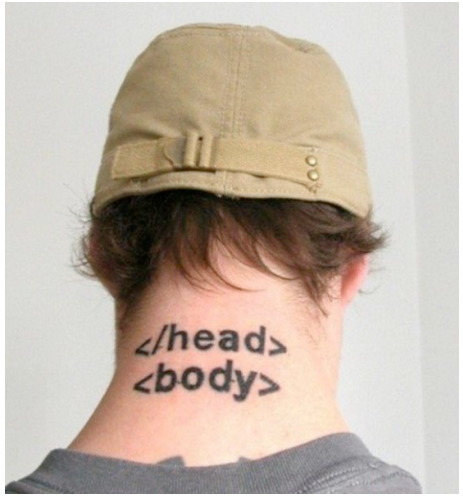
<body>

encloses page's content (text, images, forms, audio, etc.)

visible in browser window

head + body

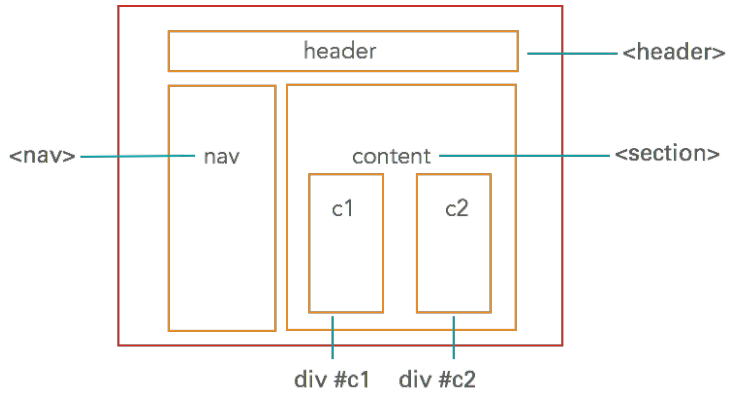
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
          <link href="assets/stylesheets/des-117.css" rel="stylesheet">
     <link href="https://fonts.googleapis.com/css?family=Cabin" rel="stylesheet">
</head>

    <body>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>

    </body>
  </html>
```



main, article, aside

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>

    <body>
      <main>
        <article>
          <h1>Foobar</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </article>
      </main>

      <aside>
        <p>Nam vitae tellus vitae erat ornare posuere at et purus..</p>
      </aside>

    </body>
  </html>
```

wireframe to layout



document object model

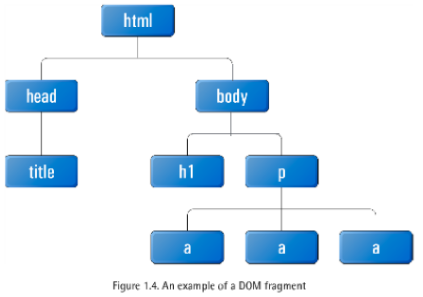


Figure 1.4. An example of a DOM fragment

From jQuery: Novice to Ninja

links

phrasing (or inline) elements by default

must include an href attribute (hypertext reference)

links can link to . . .

an html page on another website

an html page on the same website

a section within the same page

a media file or some other file to be downloaded.

nothing (a link can be empty)

a link's content can be . . .

text

an image

another element

external link

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>

    <body>

<p>
  <a href=”http://www.foobar.com” rel=”external”>Foobar</a>
</p>

    </body>
  </html>
```

internal link

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>

    <body>

<p>
  <a href=”about.html”>About Me</a>
</p>

    </body>
  </html>
```

link to another directory

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>

    <body>

<p>
  <a href=”products/cheese.html”>Cheese</a>
</p>

    </body>
  </html>
```

link to a directory one level up

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>

    <body>

<p>
  <a href="../index.html">Home</a>
</p>

    </body>
  </html>
```

empty link

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>

    <body>

<p>
  <a href="#">Foobar</a>
</p>

    </body>
  </html>
```

images

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
  </head>

    <body>

<p>
  <img src="images/foobar.png" height="200" width="400" alt="foo">
</p>

    </body>
  </html>
```

jpg or jpeg

lossy compression
24-bit color
good for photographs

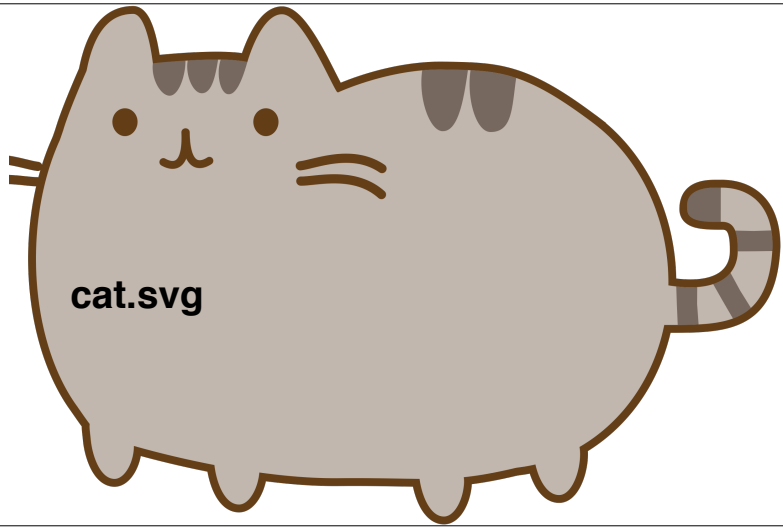
FILTHY BRITCHES

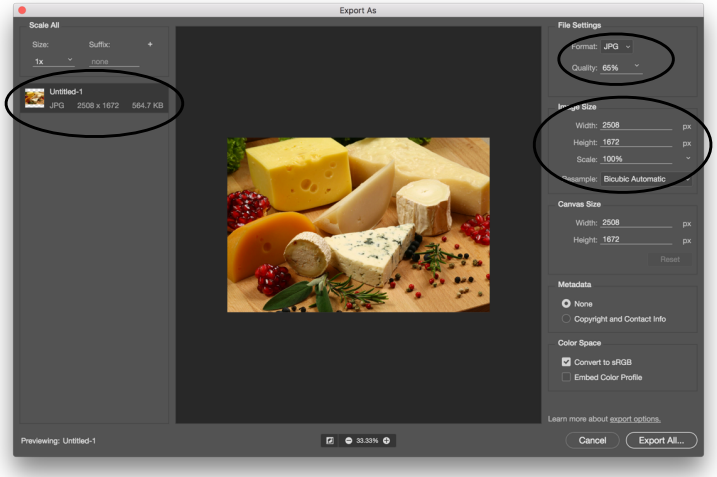
png

24-bit color
option for 8-bit transparency
good for photographs and graphics

gif

8-bit color (no continuous tone)
smaller files
option for transparency
good for computer generated graphics
can be animated!





copyright

when in doubt, create your own images

always create your own images for this class

ultimately, learn to search for images with particular licensing

best practices

kiss (keep it simple…)

keep code “lean and mean”

try not to repeat yourself

type from the outside in

1. <h1></h1>
2. <h1>Here is my header</h1>
3. <h1 attr="value">Here is my header</h1>

properly indent your code

```
<section>
        <article>
                <p>Here is a paragraph in an article, which is in a section.</p>
    <article/>
</section>
```

use atom-beautify

html comments

```
<!-- ==== START MAIN CONTENT ==== -->

  <main>
    <h1>The Little Mermaid</h1>
    <p>Far out in the ocean, the water is as blue as the <em>prettiest</em> cornflower.</p>
    <img src="images/little-mermaid.jpg"/>
  </main>

<!-- end main content -->

<!-- ==== START ASIDE ==== -->

  <aside>
    <a href="#">All about Hans Christian Andersen</a>
  </aside>

<!-- end aside -->
```

css

roles of html + css

html is for structure, css is for style

if it's about content, it's html. If it's about presentation, it's css

css location

where css resides

external

apply to many pages at once, update many pages at once

internal

overrides external

local

overrides internal

linking to a css stylesheet

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>The Little Mermaid</title>
    <link href="stylesheets/little-mermaid.css" rel="stylesheet">
  </head>

    <body>
    </body>
  </html>
```

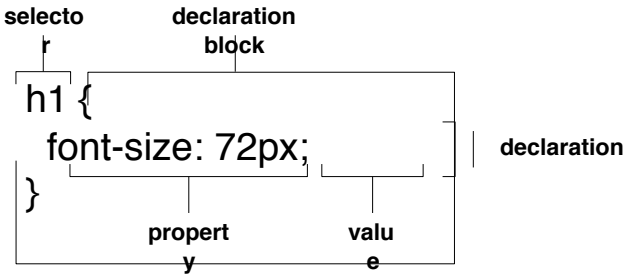
index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>The Little Mermaid</title>
    <link href="stylesheets/little-mermaid.css" rel="stylesheet">
  </head>

  <body>
    <h1>The Little Mermaid</h1>
    <p>Far out in the ocean, where the water is as blue as the
prettiest cornflower, and as clear as crystal, it is very, very deep.</
p>
    <img src="images/little-mermaid.jpg"/>
  </body>
</html>
```

little-mermaid.css

```
h1 {
  font-size: 72px;
  font-color: #2f4f4f;
  font-family: georgia;
}

p {
  font-size: 11px;
  font-color: #696969;
  font-family: verdana;
}

img {
  border: 1px solid black;
}
```

css style rules



selectors

- type selectors
- contextual selectors
- pseudo class selectors
- attribute selectors
- class or id selectors

type selectors

```
h1 {
  font-size: 72px;
  font-color: #2f4f4f;
  font-family: georgia;
}

p {
  font-size: 11px;
  font-color: #696969;
  font-family: verdana;
}

img {
  border: 1px solid black;
}
```

contextual selectors

selects elements based on their position in the “tree”

pinpoints elements based on their ancestors, parent or siblings

descendant selector

```
p em{
  background-color: black;
  font-family: creepster;
  font-size: 16px;
}
```

pseudo class selectors

can select the first, last, or any specific child element

can select different states of an element

useful for lists and links

selecting the first child

```
ul li:first-child {
  background-color: black;
  font-family: creepster;
  font-size: 16px;
}
```

selecting the last child

```
ul li:last-child {
  background-color: black;
  font-family: creepster;
  font-size: 16px;
}
```

selecting another child

```
ul li:nth-child(2) {
  background-color: black;
  font-family: creepster;
  font-size: 16px;
}
```

selecting link pseudo classes

```
a:link {
  color: ghostwhite;
}

a:visited {
  color: midnightblue;
}

a:focus {
  color: black;
}

a:hover {
  color: orange;
}

a:active {
  color: darkgray;
}
```

lvha

```
a:link {
  color: ghostwhite;
}

a:visited {
  color: midnightblue;
}

a:hover {
  color: orange;
}

a:active {
  color: darkgray;
}
```

attribute selectors

selects elements that have certain attributes or attribute values

numerous ways to specify

attribute selectors

```
img[alt] {
  background-color: ghostwhite;
}

img[alt=”davidpumpkins”] {
  background-color: ghostwhite;
}

img[alt-=”davidpumpkins”]{
  background-color: ghostwhite;
}

img[altl=”davidpumpkins”]{
  background-color: ghostwhite;
}
```

```
img[alt^=”davidpumpkins”]{
  background-color: ghostwhite;
}

img[alt$=”davidpumpkins”]{
  background-color: ghostwhite;
}

img[alt*=”davidpumpkins”]{
  background-color: ghostwhite;
}
```

selecting multiple elements

```
ul li:last-child, img[alt*=”davidpumpkins”], a {
  background-color: black;
  font-family: creepster;
  font-size: 16px;
}
```

classes and id’s

```
<!-- ==== START MAIN CONTENT ==== -->

<main>
  <h1 class=”story-title”>The Big Toe</h1>
  <p id=”paragraph1”>One day, a boy was digging in his garden, when he saw <em>a big toe</
em> sticking out of the ground.</p>
  <p id=”paragraph2”>He tried to pick it up but it was stuck.</p>
  <img class=”illustration” src=”images/big-toe.jpg”/>
</main>

<!-- end main content -->
```

classes

start with a . (as in .bigText)

can be applied to multiple elements

more than one class can be applied to a single element

can be applied to different types of elements

assigning class names

```
<article class="scary-story">
  <h1 class="story-title">The Big Toe</h1>
  <p>One day, a boy was digging in his garden, when he saw a big toe sticking out of the
ground.</p>
  <img class="illustration" src="images/big-toe.jpg" alt="A Big Toe"/>
</article>
```

assigning class names

```
<article class="scary-story">
  <h1 class="story-title">The Big Toe</h1>
  <p>One day, a boy was digging in his garden, when he saw a big toe sticking out of the
ground.</p>
  <img class="illustration" src="images/big-toe.jpg" alt="A Big Toe"/>
</article>

<article class="scary-story">
  <h1 class="story-title">The Thing</h1>
  <p>Trevor looked up and let out a scream of terror.</p>
  <img class="illustration" src="images/the-thing.jpg" alt="The Thing"/>
</article>
```

class selectors

```
.scary-story {
  border: 1px solid black;
}

.story-title {
  font-size: 66px;
  color: #666;
  font-family: creepster;
}
```

id's

start with # (as in #column1)

id's can be used only once per page—must be unique

an element can have only one ID

useful for layout

useful for linking sections of pages, and applying js

assigning id's

```
<ul class="story-index">
  <li id="story1">The Big Toe</li>
  <li id="story2">The Thing</li>
  <li id="story3">The Red Spot</li>
</ul>
```

id selectors

```
#story1 {
  color: midnightblue;
}

#story2 {
  color: ghostwhite;
}

#story3 {
  color: orange;
}
```

values

values can be:

words (bold, normal)

numerical measurement (in, cm, mm, pt, pc, px, em, ex, %)

color (hex, numeric value, or %)

color values

color keywords (color: blue)

hexadecimal (color: #0000ff)

short hex (color: #00f)

rgb (color: rgb(0, 0, 100))

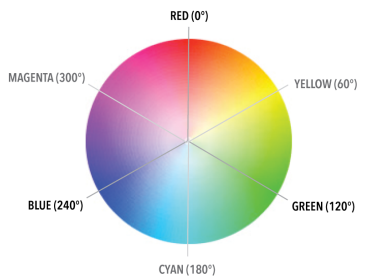
rgba (color: rgb(0, 0, 100, .75))

hsl (color: hsl(282, 100%, 25%))

hsla (color: hsl(282, 100%, 25%, .75))

hsl

hsl(282, 100%, 25%)



web fonts

allows css to link to a font on a server

much more variety

some free, some not

google fonts, adobe edge web fonts, monotype library, etc.

font control

font-family: Font name or generic names, such as serif, sans-serif, monospace, fantasy, cursive

font-size: can be absolute, relative, or keyword (small, x-small, large, etc)

font-style: italic, normal, oblique

font-weight: normal and bold are really the only ones that work

font-variant: small-caps or normal

text control

text-indent: indents first line (positive or negative)

letter-spacing: tracking

word-spacing: space between words

text-decoration: underline, overline, strikethrough, blink

text-align: left, center, right

line-height: leading

text-transform: uppercase, lowercase, capitalize, none

vertical-align: moves text up or down with reference to baseline

the box model

you'll decide if you want lots of space after each <h1>

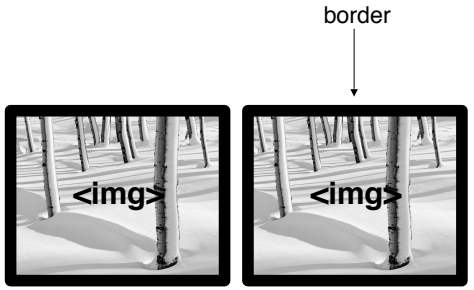
or if you want hardly any space at all
between two lines of text

or if you want every paragraph to be indented by 15 pixels

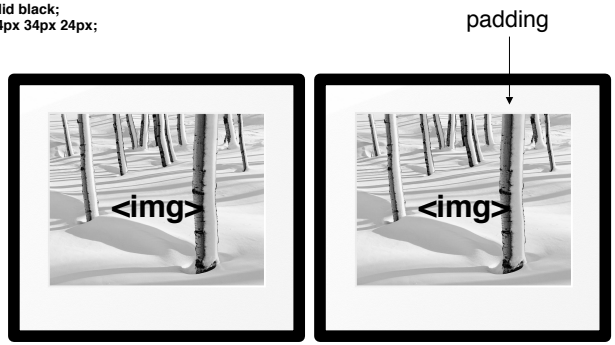
and have a line height of 35 pixels

the box model:

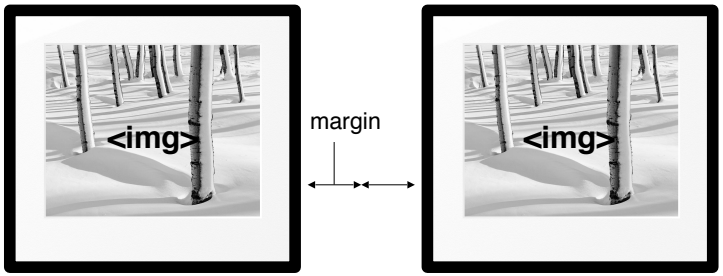

```
img {
border: 12 px solid black;
}
```

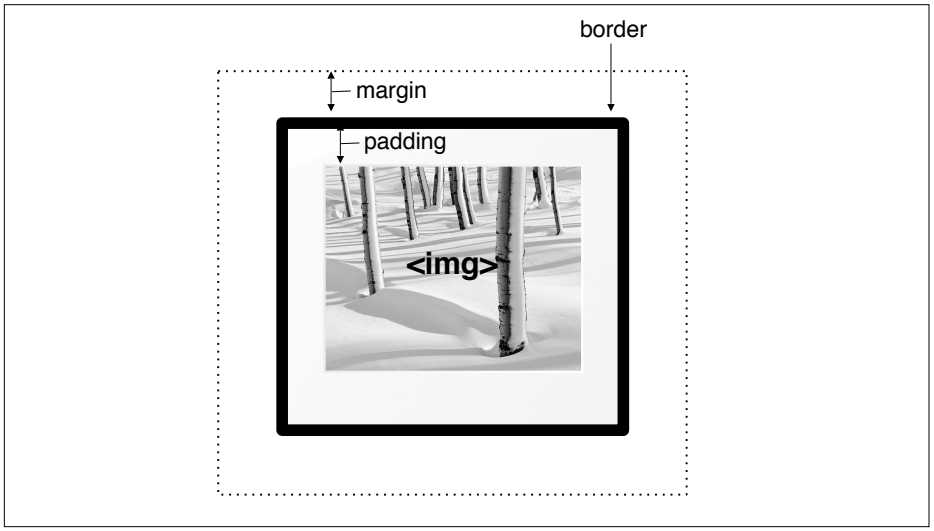


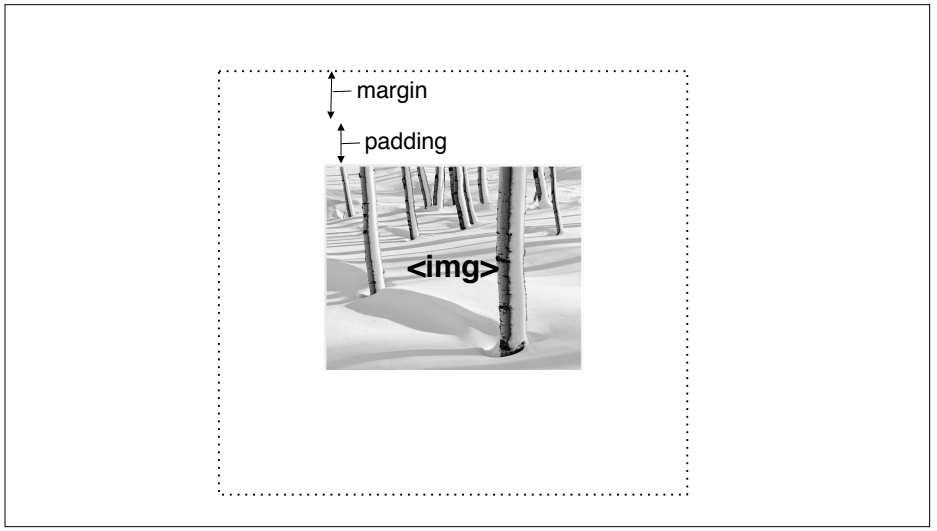

```
img {
border: 12 px solid black;
padding: 24px 24px 34px 24px;
}
```

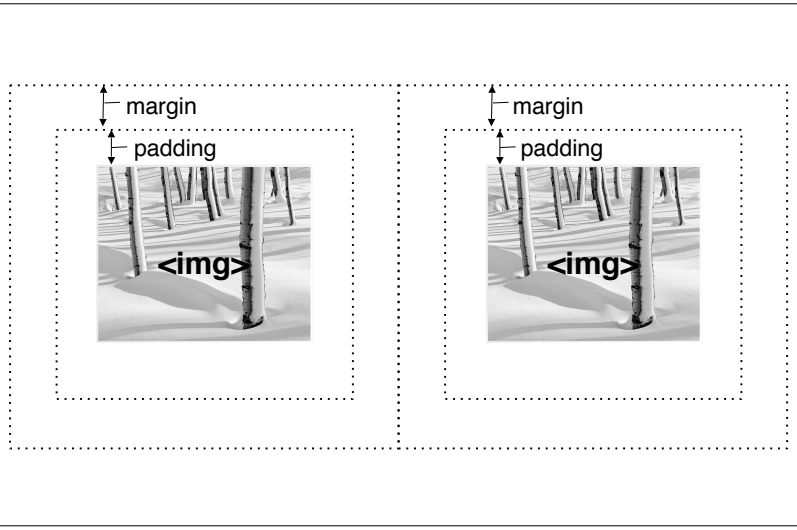


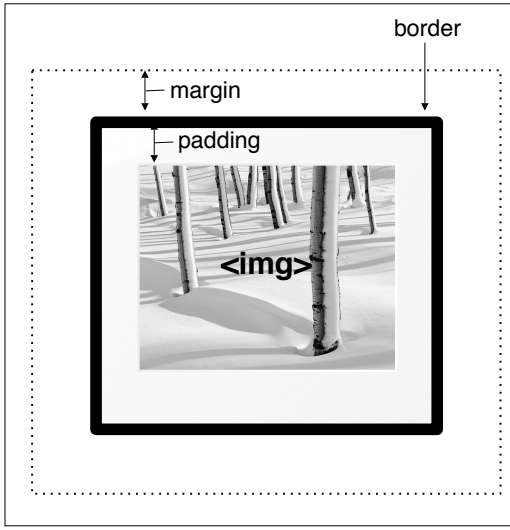

```
img {
border: 12px solid black;
padding: 24px 24px 34px 24px;
margin: 24px;
}
```











element width +
right padding width +
left padding width +
right border width +
left border width+
right margin width+
left margin width
=
computed width

css positioning

the document flow

the arrangement of page elements as defined by
positioning statements and the order of html statements

how different elements take up space and
arrange themselves around each other

display property values

block
inline
inline block

block-level elements

a stack of boxes
take up entire width of parent container (body, section, etc.)
always on their own line
can have a width and height
<p>, <h1-h6>, <ul>, <ol>, etc.

inline elements

not on a new line

occupies space bounded by element tags

cannot have a width or height

, , <small>, etc.

inline block elements

not on a new line

occupies space bounded by the element tags

can have a width and height

any block or inline element can be set to display: inline-block

position property values

static (default)

relative (to where it would be)

absolute (out of flow)

fixed (does not scroll with page)

use with: top, bottom, left, right

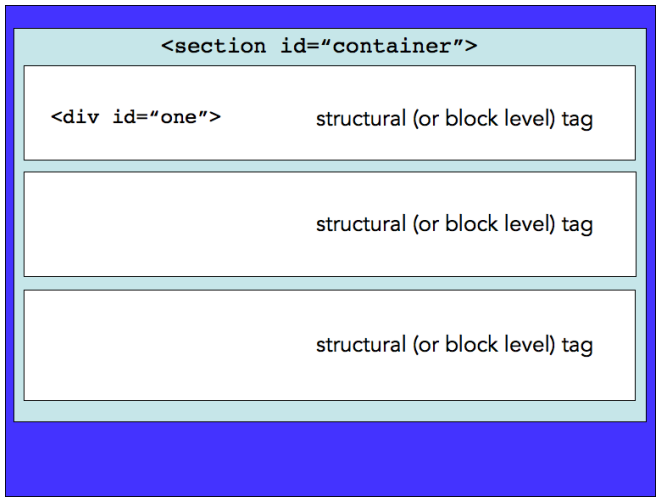
```
h1{
  position: absolute;
  top: 1in;
}
```

floats

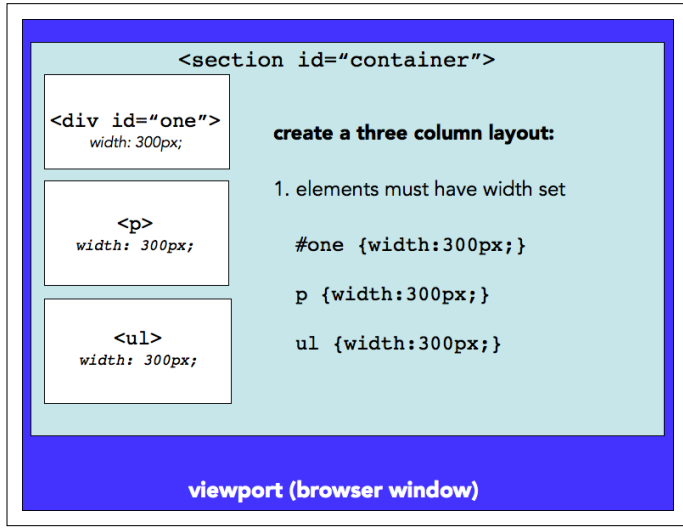
float: left, right

clear: left, right, both, none

.

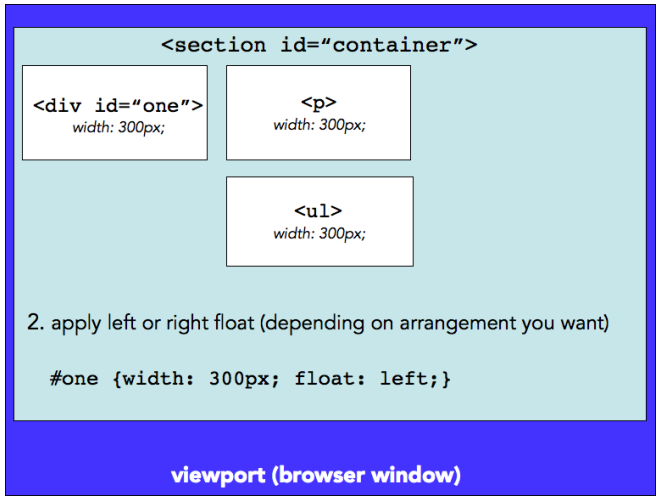

<section id="container">

structural (or block level) tag
structural (or block level) tag
structural (or block level) tag


<section id="container">

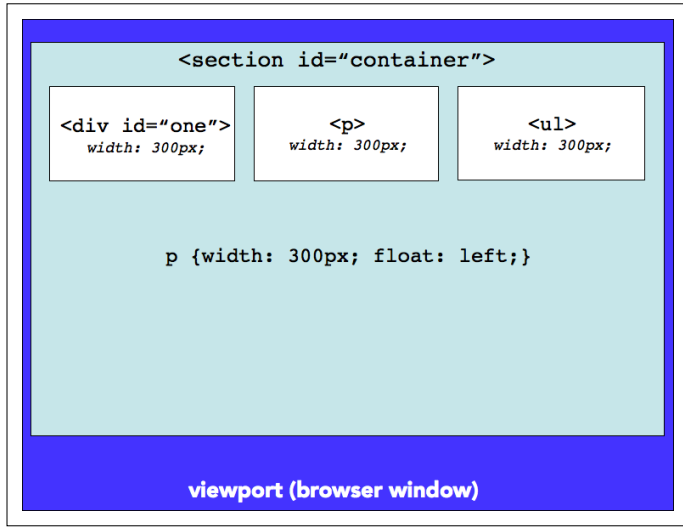
width: 300px;
<p>
width: 300px;

width: 300px;
create a three column layout:
1. elements must have width set
#one {width:300px;}
p {width:300px;}
ul {width:300px;}
viewport (browser window)


<section id="container">

width: 300px;
<p>
width: 300px;

width: 300px;
2. apply left or right float (depending on arrangement you want)
#one {width: 300px; float: left;}
viewport (browser window)


<section id="container">

width: 300px;
<p>
width: 300px;

width: 300px;
p {width: 300px; float: left;}
viewport (browser window)

layout suggestions

- divide into logical sections
- use header tags to identify and prioritize
- use divs to subdivide

.

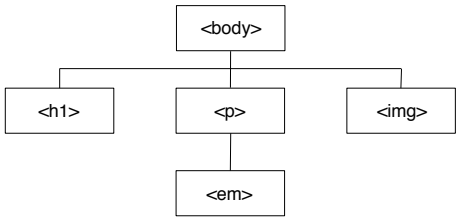
inheritance

inheritance

```
<body>
  <h1>The Little Mermaid</h1>
  <p>Far out in the ocean, the water is as blue as the <em>prettiest</em> cornflower.</p>
  <img src="images/little-mermaid.jpg"/>
</body>
```

inheritance and the document-object-model

```
<body>
  <h1>The Little Mermaid</h1>
  <p>Far out in the ocean, the water is as blue as the <em>prettiest</em> cornflower.</p>
  <img src="images/little-mermaid.jpg"/>
</body>
```



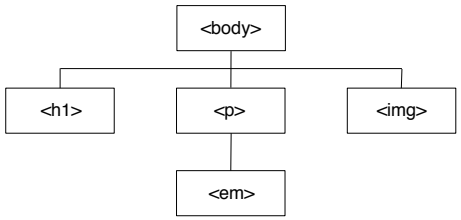
inheritance

index.html

```
<body>
  <h1>The Little Mermaid</h1>
  <p>Far out in the ocean, the water is as blue as the
<em>prettiest</em> cornflower.</p>
  <img src="images/little-mermaid.jpg"/>
</body>
```

little-mermaid.css

```
body {
  font-family: tahoma;
}
```



inherited

color
font
font-family
font-weight
letter-spacing
text-align
list-style

not inherited

border
float
height
width
margin
padding
z-index

best practice

apply a style rule to the highest possible node up the tree

don't repeat yourself if you don't have to

no

```
h1 {
  font-size: 36px;
  font-color: #2f4f4f;
  font-family: georgia;
}

p {
  font-size: 36px;
  font-color: #2f4f4f;
  font-family: georgia;
}

a {
  font-size: 36px;
  font-color: #2f4f4f;
  font-family: georgia;
}
```

yes

```
body {
  font-size: 36px;
  font-color: #2f4f4f;
  font-family: georgia;
}
```

the cascade

the cascade

1. specificity
2. order
3. importance

specificity

```
p {
  font-family: verdana;
}

p em {
  font-family: helvetica;
}

footer p em {
  font-family: tahoma;
}
```

order

```
footer p em {
  font-family: tahoma;
  font-family: georgia;
  font-family: helvetica;
}
```

importance

```
footer p em {
  font-family: tahoma;
  font-family: georgia !important;
  font-family: helvetica;
}
```

css comments

```
/* HEADER
-------------------------*/

h1 {
  font-size: 72px;
  font-color: #2f4f4f;
  font-family: georgia;
}

/* MAIN CONTENT
-------------------------*/

p {
  font-size: 11px;
  font-color: #696969;
  font-family: verdana;
}

img {
  border: 1px solid black;
}
```

css reset

css reset

- defines new default CSS styles
- smooths out styles across browsers
- keeps us from having to remember all default browser styles!
- http://meyerweb.com/eric/tools/css/reset/

validation

validate your files!

or you won't get a grade

```
<a href="http://validator.w3.org/check?uri=referer">
<img src="images/html5.png" width="48" height="48" alt="link to html5 validation"></a>

<a href="http://jigsaw.w3.org/css-validator/check/referer">
<img src="images/css3.png" width="48" height="48" alt="link to css3 validation"></a>
```

learn from others

in any browser: view > page source

troubleshooting

validate your code

print your code and read away from the computer

rubber ducky method

post it to the Discussions feature of Canvas & email glenda and Spencer

file structure