

CSS:
part one

cascading

style

sheets

CSS

PURPOSE:

HTML tells us what to display

CSS tells us how

CSS developed to make content separate from display

SYNTAX

Simple text file

Contains rules for displaying your HTML

css: types of styles

INLINE STYLES

defined within an individual tag

with a lot of content starts to become inefficient

INTERNAL STYLES

defined at the top of each individual page

with a lot of pages, starts to become inefficient

EXTERNAL STYLES

defined once for your whole site

all of your HTML pages link to the same file

extremely efficient (and its what we'll do in this class)

css: benefits of external stylesheet

central definition of visual style

make one change, the entire site is updated

can be reused on any number of pages

Developers and designers can work independently

Same content can be restyled for lots of different media

web

cell phone

print

screen reader

css: characteristics

CASCADING

style rules are applied in order

last definition takes precedence over the first

ORDER OF PRIORITY

- 1 BROWSER default ← *lowest priority*
- 2 EXTERNAL styles
- 3 INTERNAL styles
- 4 INLINE styles ← *highest priority*

linking html to css:

Only change in HTML is to add a line at the top

```
<link rel="stylesheet" type="text/css" href="styles.css" />
```

Tells browser to include a style sheet called styles.css

You can name the stylesheet whatever you want

Include just before the </head> tag

making your first file:

- 1 Create a text file called **styles.css** in your web folder.
- 2 Add a line to your HTML, right before the `</head>` tag:

```
<link rel="stylesheet" type="text/css" href="styles.css" />
```

- 3 Start editing **styles.css**.

before we begin

PREPARE YOURSELF:

lots and lots of style properties

don't worry about memorizing them!

keep reference open while coding

GOOD RESOURCES (linked on course website)

<http://www.w3schools.com>

CSS Pocket Reference (O'Reilly, \$10)

css: syntax

```
selector {  
  
    property: value;  
    property: value;  
    ...  
}
```

selector: what you are defining

{ and **}**: define beginning and end of rules for that selector

property: set of display properties you can change (in documentation)

value: what you're changing it to

semicolon: ends each line

css: syntax

SELECTORS:

HTML tag you want to make a rule for

```
body {
```

```
  property: value;
```

```
}
```

```
p {
```

```
  property: value;
```

```
}
```

css: syntax

You can define multiple tags at once:

```
h1, h2, h3 {
```

```
  property: value;
```

```
}
```

```
p {
```

```
  property: value;
```

```
}
```

css: syntax

Organize your CSS file with comments:

```
/* headers */
```

```
h1, h2, h3 {
```

```
  property: value;
```

```
}
```

```
/* paragraph styles */
```

```
p {
```

```
  property: value;
```

```
}
```

css: color

RGB (Red, Green, Blue)

Values 0 to 255

additive color

black: 0,0,0

white: 255,255,255

HEXADECIMAL

Encoded value of RGB

values from 00 to FF

black: #000000

white: #FFFFFF

css: color

Some predefined colors can be called by name in your css file
best not to rely on them

some are useful:

white (255,255,255)

black (0,0,0)

gray/grey (128,128,128)

red (255,0,0)

green (0,255,0)

blue (0,0,255)

css: color

color: property

refers to text color only

not background or border

can be defined in rgb, hex or by name

```
p {  
  color: rgb(255,255,255);      /* white */  
  color: #FFFFFF;              /* white */  
  color: white;                /* white */  
}
```

css: fonts

Browsers use fonts installed on the users computer:

- Different for every user

- Differs from Mac to PC

FONT STACK:

- Giving the browser a prioritized list of fonts

- Browser will try the list in order until it finds a font on the users system

- If it can't find anything, it will use the browsers default type face

css: font-family

Start with most specific requests:

Georgia, Arial, Helvetica, Verdana, Times New Roman

Finish with most generic:

serif, sans-serif

```
p {  
  font-family: Helvetica, Verdana, Arial, sans-serif;  
  font-family: Georgia, "Times New Roman", Times, serif;  
}
```

css: units of measurement

PERCENTAGE (%)

defined with %

relative to the page

PIXEL (px)

dot on computer screen

not consistent: effect can vary with different monitor resolutions

EM (em)

relative measurement equal to the current type size

allows for whole page to resize in proportion

css: font-size

Tip:

Adjust the whole page's font size in the body tag

Make specific adjustments farther down the hierarchy

```
body {
```

```
  font-size: 1em;
```

```
}
```

```
p {
```

```
  font-size: .8em;
```

```
}
```

css: more font properties

Lots of other things about fonts that you can control:

```
p {  
  font-family: Courier, Georgia, Times, serif;  
  font-size: 1.5em;  
  font-weight: bold;  
  font-style: italic;  
}
```

css: text

CSS lets you do other basic formatting

Define basic treatments like color, alignment, word wrap

Some issue of support across browsers - test carefully!

```
p {  
  text-align: right;  
  text-transform: uppercase;  
  text-indent: 10px;  
  text-decoration: underline;  
}
```

css: typography (sort of)

CSS offers some recognizable typographic functions:

```
p {  
  font-family: Courier, Georgia, Times, serif;  
  font-size: 1em;  
  line-height: 1.4em;  
  letter-spacing: 1px;  
}
```

css: links

Links are a special case because they have states (pseudo-classes):

LINK

in effect when you load the page

VISITED

in effect if the link is stored in browser history

ACTIVE

in effect while mouse button is pressed on a link

HOVER

in effect while mouse is over link but not clicking

css: links

This link changes when you hover over it:

```
a {  
  color: white;  
  font-size: 1em;  
}  
  
a:hover {  
  color: green;  
  font-size: 1.3em;  
}
```

css: background color

background-color: property

sets the background of any html element

set different backgrounds for each

```
p {  
    background-color: rgb(255,0,0);           /* white */  
    background-color: #FFFFFF;              /* white */  
    background-color: white;                /* white */  
}
```

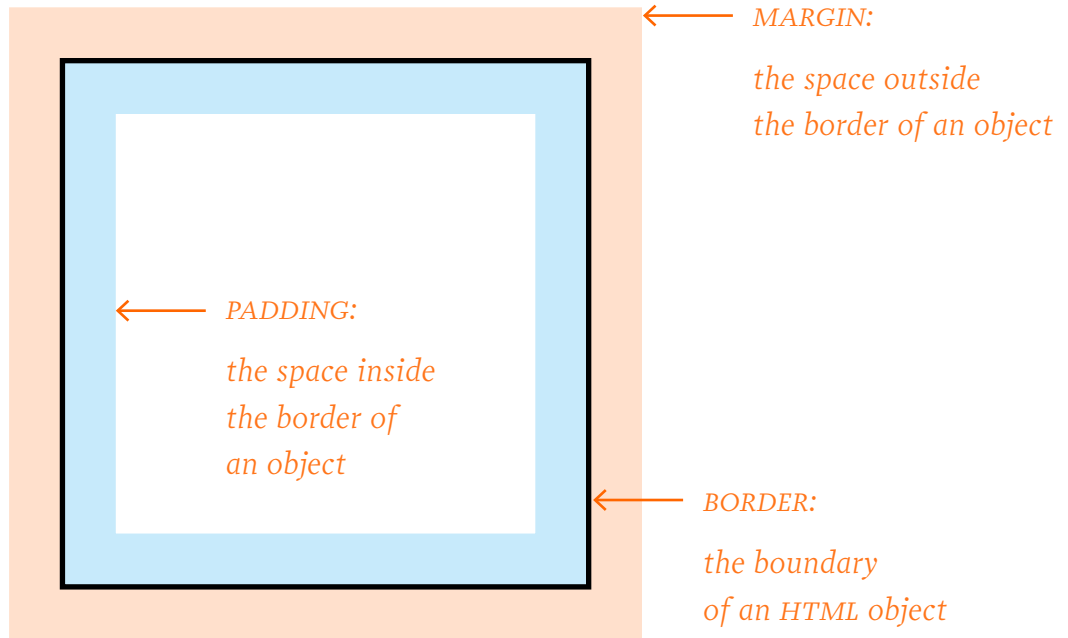
css: more backgrounds

Set the background of any selector

Can be color, image, combination of both

```
body {  
  background-color: rgb(255,255,255);  
  background-image: url(images/imagename.jpg);  
  background-repeat: no-repeat;  
  background-position: center center;  
}
```

css: anatomy of an html tag



css: borders

spell it all out, or define with shorthand

```
p {
```

```
  border-left-color: red;
```

```
  border-left-width: 1px;
```

```
  border-left-style: dashed;
```

← *spelled out, one line for each property of each border*

```
  border-left: 1px dashed red;
```

← *shorthand for each side*

```
  border: 1px dashed red;
```

← *shorthand for all borders*

```
}
```

css: margins

Space added outside of the border.

```
p {
```

```
margin-top: 0px;
```

```
margin-right: 5px;
```

```
margin-bottom: 10px;
```

```
margin-left: 15px;
```

← *spelled out, one
line for each side*

```
margin: 0px 5px 10px 15px;
```

```
}
```

← *shorthand for all sides
order is top, right, bottom, left*

css: padding

space inside the border

doesn't change the location of the border

constricts text/objects inside the border

```
p {
```

```
padding-top: 0px;
```

```
padding-right: 5px;
```

```
padding-bottom: 10px;
```

```
padding-left: 15px;
```

← *spelled out, one
line for each side*

```
padding: 0px 5px 10px 15px;
```

← *shorthand for all sides*

```
}
```

css: dimension

You can give anything a width and a height:

```
p {  
  width: 200px;  
  height: 200px;  
  
  /* tell the browser what to do with any overflow text! */  
  overflow: auto;  
}
```

css: all together, now.

Using color, fonts, borders, margins and padding, you can do a lot:

```
p {  
  font-family: Georgia, serif;  
  font-size: 1em;  
  color: #999999;  
  background-color: #333333;  
  border: 1px solid black;  
  margin: 10px;  
  padding: 10px;  
}
```

in class:

Using color, fonts, borders, margins, and padding,
create a style sheet for your process pages!