HTML & CSS Level 1: Week 4

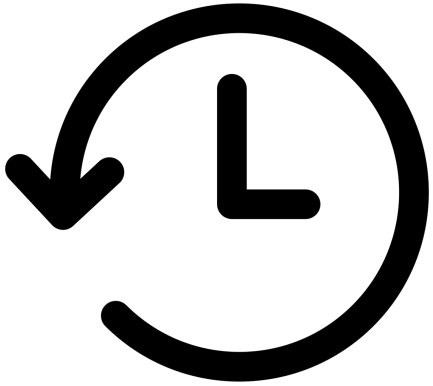
March 3, 2015

Instructor: Devon Persing

This week

- CSS abbreviations
- Pseudo-selectors
- The CSS box model
- Positioning content with CSS
- Browser reset styles

Review!



History designed by Ema Dimitrova from the Noun Project

Last time

- Used Google Fonts
- Used class and id selectors to style content
- Styled backgrounds of containers along their x and y axes

<> {} `class` and `id`

- **`class`** and **`id`** attributes can be added to any HTML element
- **Classes** are for multiple things on a page
- **IDs** are for individual things on a page
- You can make up whatever **`class`** and **`id`** values you want!

<> `class` attributes in HTML

- Classes can be shared by multiple elements on a page

```
<h1 class="kittens">...</h1>

<span class="kittens">...</span>
```

- Elements can have multiple classes

```
<div class="kittens puppies">...</div>

<div class="kittens puppies birds">...</div>
```

{ } `class` selectors in CSS

- Start with a **period** (`.`)
- Can style any element with the class

```
.kittens { color: #000000; }
```

- Or can be used to style only a specific **type** of element with the class

```
h3.kittens { color: #000000; }
```

- More specific than an HTML type selector

<> `id` attributes

- IDs *cannot* be shared by multiple elements on a single page
- Elements *cannot* have multiple IDs

```
<div id="kittens">...</div>

<div id="puppies">...</div>

<div id="birds">...</div>
```

{ } `id` selectors in CSS

- Start with a **hash/pound sign** (**#**)
- Can style the single element with the ID

```
#kittens { color: #000000; }
```

- More specific than a class selector

Mixing `class` and `id` attributes

- Elements can have **`id`** and **`class`** attributes at the same time

```
<div id="kittens">...</div>

<div id="puppies" class="small floppy">...</div>

<div id="birds" class="small feathery">...</div>
```

- ID selector styles can be used to override class selector styles

Be thoughtful in your selectors

- Recommended order of attack:
  a. Type selectors
  b. Class selectors
  c. Descendent selectors
  d. ID selectors
- If you overuse IDs in your styles, **you're going to have a bad time**

Award designed by Luc Poupard from the Noun Project

{ } Styling a background image

- The property is **`background-image`**
- The value is a **URL where an image lives**

```
.kittens {
  background-image: url("img/kittens.jpg");
}
```

{ } Styling a background image

- **`background-repeat`**: repeat horizontally or vertically, or not at all
- **`background-position`**: Start at the left or right, top or bottom, and center or not
- **`background-attachment`**: Does it stick or scroll
- **`background-size`**: How much of the container does it cover

{ } The magical image background

```
/* make a full-sized, fixed image background that covers
the whole container */
.magic-bg {
  background-image: url("img.png");
  background-repeat: no-repeat;
  background-position: center center;
  background-attachment: fixed;
  background-size: 100%;
}
```

CSS pseudo-classes and pseudo-elements

{ }

{ } Pseudo-classes are conditional

- **Pseudo-classes** are added to a selector to add conditional styles to an element
- Most commonly used to style **states** of **<a>** elements and form elements

```
a:link { /* the default state of a link */ }

a:visited { /* a link that's been clicked */ }

a:hover { /* a link that has a mouse hover */ }

a:focus { /* a link that has keyboard focus */ }

a:active { /* a link that is being clicked */ }
```

{ } `:hover` versus `:focus`

- **`:hover`** is for a link or other interactive element that has a **mouse hover**
- **`:focus`** is for a link or other interactive element that has **keyboard focus**
- Browsers have their own default **`:focus`** styles for **accessibility**

```
a:hover, a:focus {
/* it's good practice to style them together! */
}
```

{ } :hover for other elements

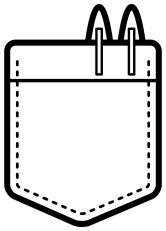
- **:hover** can be used to style hover states for some non-interactive elements to create a more dynamic experience

```
div { /* a div with a background... */
    background: #99ff66;
}

div:hover { /* ...could have another on hover */
    background: #ff6600;
}
```

{ } Some nifty pseudo-things

- `:first-letter` styles the first letter of a block of text
- `:first-child` and `:last-child` style the first and last children of a parent
- `:nth-child()` can be used to style even or odd children, or do some math to style every 5th, etc.
- `::selection` styles text that is selected by the user

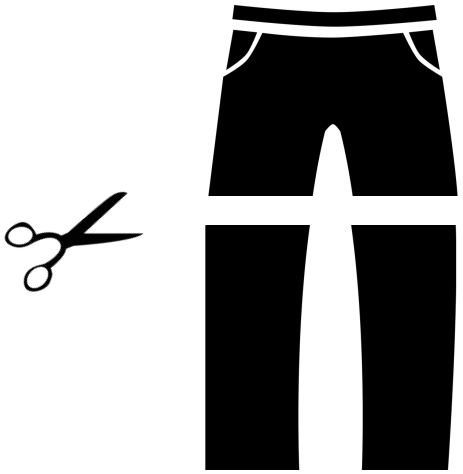


CSS selectors are evolving

- **Pseudo-classes**, **pseudo-elements**, **combinators**, and **attribute selectors** create extremely targeted ways to style content that degrade gracefully in older browsers
- To learn more of these techniques: **http://www.quirksmode.org/css/selectors/**

Pocket designed by Michela Tannoia from the Noun Project

CSS abbreviations



Pants designed by Pham Thi Dieu Linh from the Noun Project
Scissors designed by Kelly Ness from the Noun Project

{ } Abbreviated hex colors

```
color: #333333;
/* becomes */
color: #333;


color: #aa0099;
/* becomes */
color: #a09;
```

{ } Abbreviated font styles

```
font-style: italic;
font-variant: small-caps;
font-weight: bold;
font-size: 1em;
line-height: 1.5em;
font-family: Helvetica, sans-serif;

/* becomes */

font: italic small-caps bold 1em/1.5em
Arial, Helvetica, sans-serif;

/* font-size & font-family are required! */
```

{ } CSS `background` abbreviations

- **`background-color`** and **`background-image`** and can be combined into **`background`**
- Color is always listed first, then the image

```
body {
background: #eee url("img/kitten.jpg");
}
```

{ } More background

- You can also add all of your other **background-** styles:

```
/* a div with a light gray background, and a background
image that doesn't repeat and is positioned in the
bottom right */

div {
   background: #eee url("img/kitten.jpg")
no-repeat bottom right;
}
```

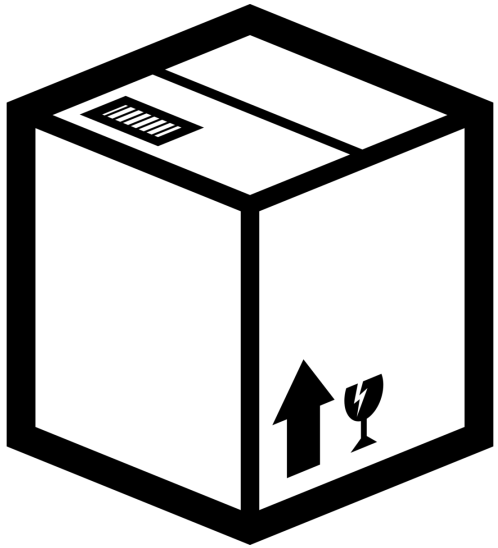
{ } Unordered list item styles

- You can shorten your list styles to just **`list-style`**

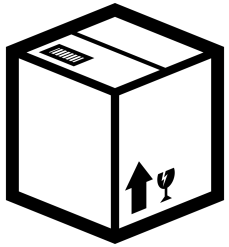
```
ul li {
   list-style-type: disc;
   list-style-position: inside;
}

ul li {
   list-style: disc inside;
}
```

The CSS box model

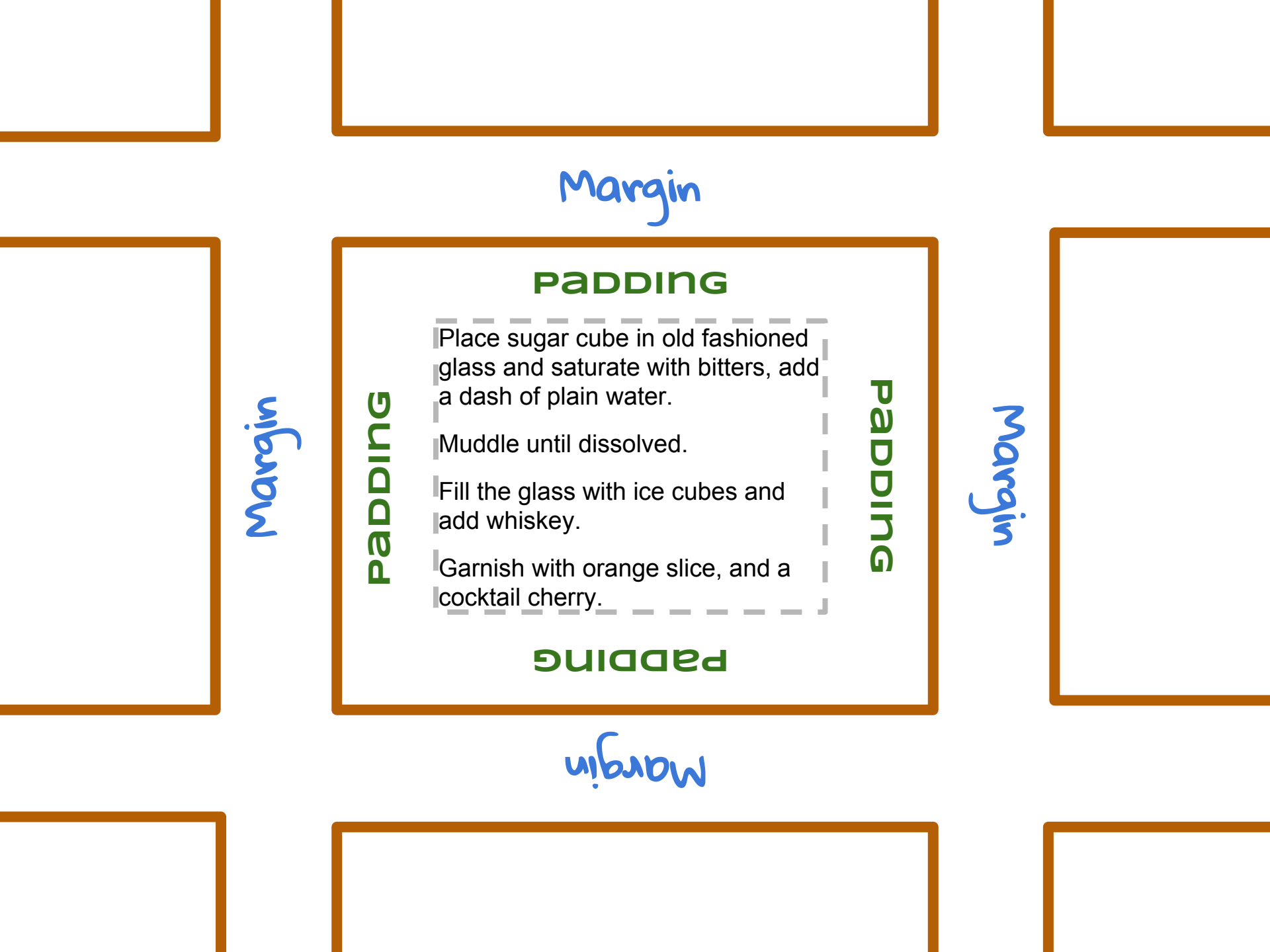


Box designed by Cornelius Danger from the Noun Project



A CSS box model metaphor

- **Content**: stuff in the box
- **Padding**: bubble wrap & packing peanuts
- **Border**: the sides of the box
- **Margin:** space between multiple boxes
- In general, the box model works for **block** and **inline-block** elements


Margin
Padding
Place sugar cube in old fashioned glass and saturate with bitters, add a dash of plain water.
Muddle until dissolved.
Fill the glass with ice cubes and add whiskey.
Garnish with orange slice, and a cocktail cherry.
Padding
Padding
Margin
Margin
Margin
Margin

{ } Box model content

- By default, content helps determines the default **`width`** and **`height`** of the element's box
- Defaults for block elements can be overridden with CSS

```
div { /* px, em, %, auto, etc. */
   width: 400px;
   height: 200px;
}
```

{ } Box model `padding`

- Creates space between content and the **`border`** for readability

```
padding-top: 20px;
padding-right: 20px;
padding-bottom: 40px;
padding-left: 40px;
```

{ } padding abbreviated

```
padding-top: 20px;
padding-right: 30px;
padding-bottom: 40px;
padding-left: 50px;

/* abbreviations for boxes go clockwise! */

padding: 20px 30px 40px 50px;
```

{ } padding abbreviated further

```
/* top/bottom and left/right match? */
padding-top: 20px;
padding-right: 40px;
padding-bottom: 20px;
padding-left: 40px;
/* combine them! */
padding: 20px 40px;
```

{ } padding abbr. even further!

```
/* all match? */

padding-top: 20px;
padding-right: 20px;
padding-bottom: 20px;
padding-left: 20px;

/* combine them even more! */

padding: 20px;
```

{ } Box model `border`

- Goes around the edge of the element
- Default **`width`** is **`0`** for most elements
- Borders can have **`color`** and **`style`** too

```
border-width: 20px;
border-style: dotted;
border-color: #ff0000;

/* border-width for the bottom edge only */

border-bottom-width: 4px;

/* border-color for the left edge only */

border-left-color: #ff0000;
```

{ } border abbreviations

```
border-top-width: 4px;
border-right-width: 3px;
border-bottom-width: 4px;
border-left-width: 3px;
border-style: solid;
border-color: #a00;
/* becomes */
border: 4px 3px solid #a00;
```

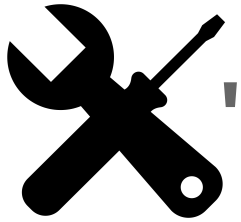
{ } Box model `margin`

- Goes outside the **`border`**
- Creates space between boxes
- Gets abbreviated the same way
- **Can be negative** to shift elements

```
margin-top: -20px;
margin-right: 40px;
```

{ } With `background`

- **`background`** styles fill both the **content** and the **`padding`** of elements
- **background-position** can also be negative!
- Use **`background-position`** and the box model properties to arrange your background images in cool ways



"Seeing" the box model

1. Open your Web Inspector (right click in the browser and choose "Inspect Element")
2. Hover your mouse over a line of code within the **`<body>`**
3. See different colors to denote different parts of the box

Toolset designed by Calvin Goodman from the Noun Project

{ } What is up with my box sizes?

- How containers take up space is dictated by the **`box-sizing`** property
- The default value for **`box-sizing`** is **`content-box`**

```
box-sizing: content-box;
```

- This means that **`width`** and **`height`** include only the content by default

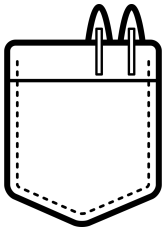
{ } `border-box` to the rescue

- Changing **`content-box`** to **`border-box`** makes it so that the width and height include the border and padding

```
html {
  box-sizing: border-box;
}
*, *:before, *:after {
  box-sizing: inherit;
}
```

Intro to layouts

Layout designed by Yamini Chandra from the Noun Project



A brief history of web layouts

- Before CSS, we used **`<table>`** elements to make layouts :(
- With CSS we can use a variety of properties to arrange elements on the screen by adjusting the flow of the page
  - **Pros:** Any content can be displayed anywhere!
  - **Cons:** Any content can be displayed anywhere!

{ } Setup: Centering our page

- Most websites sit in the middle!
- To do this, give your **<body>** (or another container that wraps the whole page):
  - a **width** value
  - a left *and* right **margin** value of **auto**

```
body {
   width: 960px;
   margin: 0px auto;
}
```

{ } Layout properties

- **`display`**: for dictating how elements behave within the box model
- **`float`**: for moving elements around within the page flow
- **`position`**: for moving elements in and out of the page flow altogether

{ } The `display` property

- Remember block, inline, and inline-block elements?
- You can roll your own with the **`display`** property
- The most common ones are:
  - **`display: block;`**
  - **`display: inline;`**
  - **`display: inline-block;`**

{ } Why use `display`?

- Make a link look like a button
- Add padding and margins to a "naturally" inline element like a **`<span>`**
- Make a list of navigation links horizontal
- Any use cases to keep style and content separate

{ } inline-block example

- Make a list of navigation links horizontal and look button-y

```
.nav li { /* for positioning */
   display: inline-block;
   vertical-align: top;
}

.nav li a { /* for button-y-ness */
   display: block;
   padding: 20px;
   background: #5fc09a;
}
```

{ } Floating down the river

- Our page is a flowing river of HTML elements
- Elements can be floated on the river to allow other elements to flow around them
- Most common way to make a multi-column layout

{ } `float` property

- Easiest way to offset content like images, pullquotes, or other elements within the flow of a document
- Requires that an element have **`display: block;`**
- Has three values: **`left`**, **`right`**, **`none`**

```
.titanic {
    float: none;
}
```

{ } The simplest column layout

1. Have a parent element (like **`<body>`**) and give it a width value:
2. Give elements you want to be columns **`width`** values that add up to the parent's **`width`** (child A + child B = parent)
3. Make your columns **`float: left;`**
4. Voila!

{ } `clear` property

- HTML river showing up in weird places?
- The **`clear`** property fixes **`float`** and also has three values:

```
clear: left;
clear: right;
clear: both;
```

{ } self-clearing floats

- One of the tricky things about floats is when to stop floating
- You can use a pseudo-element on the parent of the floated elements to create a "self-clearing" float

```
.parent:after {
  content: "";
  display: table; /* we'll talk about this shortly! */
  clear: both;
}
```

{ } Using the `position` property

- The **`position`** property lets us arrange elements:
  - In relation to the flow (**`relative`**)
  - In a very specific place outside of the flow or within a **`relative`** element (**`absolute`**)
  - In relation to the browser window (**`fixed`**)
- How **position** is applied depends on to where the element is in the flow by default

{ } Tweaking the `position`

- We can dictate where elements go down to the pixel
- **`left`**, **`right`**, **`top`** and **`bottom`** + or - pixels between positioned elements and their containers

```
div {
    position: absolute;
    right: -10px;
    top: 30px;
}
```

{ } Using `position: fixed;`

- **`Position: fixed;`** is a way to make content "stick" to the browser window, regardless of where the user scrolls
- Commonly used to make headers, navigation, or footers that follow the page as it scrolls

Resetting browser default styles

Browser defaults can be a pain

- Every browser has slightly different styles
- Different types of elements get different font sizes, line-height, padding, margins, etc.
- Tweaking styles for individual types of elements is time consuming

A blank slate

- Reset styles strip out browser default and let us make our own defaults
- We'll use the canonical reset stylesheet: **http://meyerweb.com/eric/tools/css/reset/**

Two ways to add reset styles

- **Method one:** Put reset styles into **their own .css file** and load it before your existing stylesheet

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

- **Method two:** Put reset styles **into the top** of your existing stylesheet

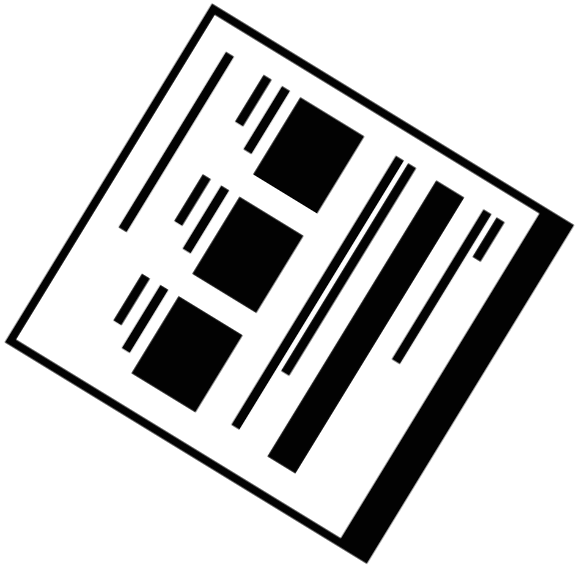
Method one: Separate styles

1. Copy reset styles
2. Paste into a new blank document in your text editor
3. Save your styles as a different .css file (e.g., **`reset.css`**) in your css folder
4. Add a link to the reset stylesheet in **`<head>`**, *before* your existing styles

Method two: One stylesheet

1. Copy reset styles
2. Paste at the very top of your existing stylesheet (e.g., **styles.css**) so they load first
3. Save your stylesheet

(ノ^_^)ノ



"Homework"

- Use a reset stylesheet
- Style your boxes!
- Practice making a multi-column layout
- Practice abbreviating your CSS
- *HTML5 for Web Designers*: Ch. 13-15

Next time

- New and old HTML elements
- Mobile-friendly layouts
- iframes and embedded media
- Next steps and resources for learning
- Related topics and lingering questions

Questions? Comments?

- Visit dpersing.github.io/svc for:
  - Class slides
  - Code samples
  - Resources
- Email me: dep@dpersing.com
- Tweet at me: @devonpersing