HTML & CSS Level 1: Week 4

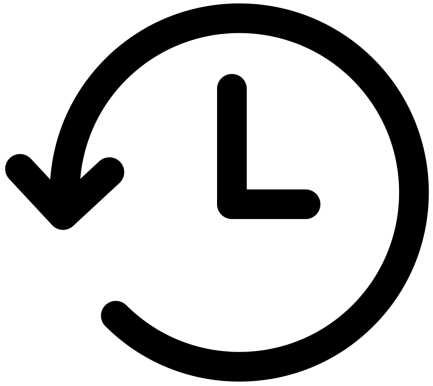
May 26, 2015

Instructor: Devon Persing

This week

- CSS abbreviations
- Using classes and ids for styles
- Pseudo-selectors
- Backgrounds with images, opacity, and gradients

Review!



History designed by Ema Dimitrova from the Noun Project

Last time

- Used Google Fonts
- Block, inline, and inline-block elements
- The box model
- Data tables

{ } Web fonts

- Web fonts let us style sites with **fonts that users may not have** on their own device
- Web font services **licence fonts for online use** specifically
- Files are either:
  - hosted by a service
  - served with your pages

<> Block and inline elements

Block elements:

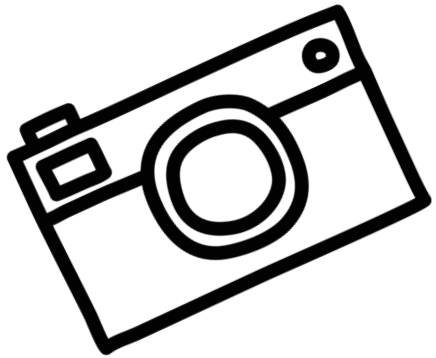
- Create **linebreaks**
- **Take up "space"** on the page

Inline elements:

- **Don't** create linebreaks
- **Flow within other content** on the page

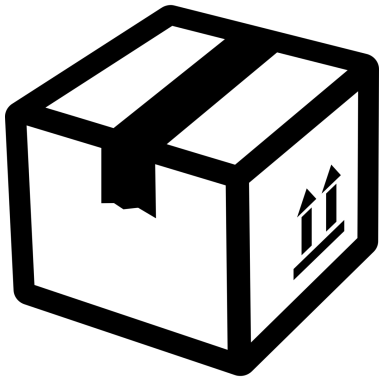
<> The rare inline-block element

- **Inline-block** elements behave a bit like both block and inline elements:
  - Take up height and width like block elements
  - Flow with content around them
- So far we know `img` elements



<> `<div>` elements

- **`div`** elements are **generic block** elements
- Used to **create sections or groupings** in HTML pages for layout and style
- Function **like a box** to put content (or other **`div`** elements) inside
- Have heights and widths



Box designed by Mourad Mokrane from the Noun Project

<> `<span>` elements

- **`span`** elements are **generic inline** elements
- Can **nest inside** other block or inline elements
- Used to **style other inline content** or **content inside block elements**
- **Flow** with the content around them

<> More inline elements

- **`em`** elements are used to show the equivalent of *spoken emphasis*
- **`strong`** elements are used to show **importance in context**

```
<p>"Oh, great. Someone ate <em>my only clean socks</em>."
</p>

<p>"Was it <strong>the cat</strong>?"</p>

<p>"No, it was <strong>the dog</strong>."</p>
```

{ } Width and height

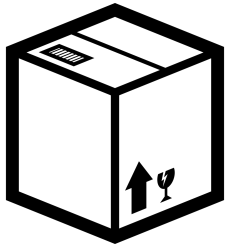
- Some elements have width and height by default
- You can set the width and height of images with HTML attributes:

```
<img src="example.jpg" alt="" width="
300px" height="200px">
```

- But it's recommended to adjust them with CSS:

```
img { width: 300px; height: 200px;}
```

```
img { width: 300px; height: auto;}
```



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

{ } 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;
```

{ } 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;
```

{ } Box model `margin`

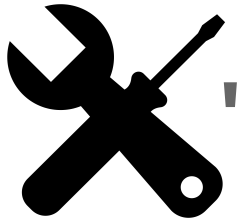
- Goes outside the **`border`**
- Creates space between boxes
- **Can be negative** to shift elements

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

{ } `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;
}
```



"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 and values to denote different parts of the box

Toolset designed by Calvin Goodman from the Noun Project

<> Basic table elements

- **`<table>`** wraps all **table** elements
- **`<tr>`** creates a **row** of table cells
- **`<th>`** creates a **table header** cell for a column *or* a row
- **`<td>`** creates a regular **table data** cell within a row

{ } Styling table elements

- All of our box model styles can be applied!
- Some additional styles for cells:
  - **`border-spacing`** puts space between cells

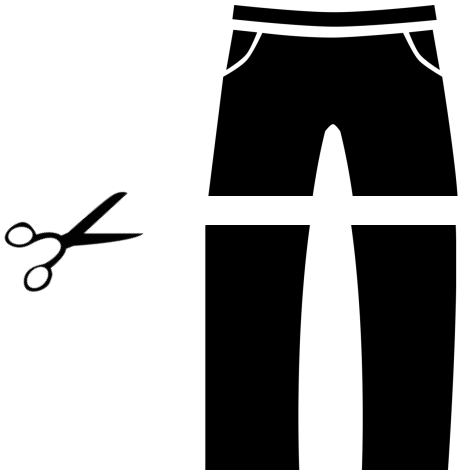
    ```
    table { border-spacing: 4px; }
    ```

  - **`border-collapse`** makes cell borders overlap or sit up against each other

    ```
    table { border-collapse: collapse; }
    ```

Questions?

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
Helvetica, sans-serif;

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

{ } 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;
}
```

{ } 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;
```

{ } border abbreviations

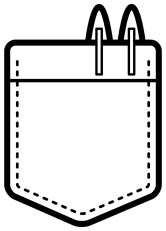
```
border-top-width: 4px;
border-right-width: 3px;
border-bottom-width: 4px;
border-left-width: 3px;
border-style: solid;
border-color: #a00;

/* becomes */

border: 4px 3px solid #a00;
```

{ } margin abbreviated

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

/* margin works just like padding! */
margin: 20px 30px 40px 50px;

margin: 20px 40px;

margin: 20px;
```

Using `class` and `id` selectors

Identification Badge designed by Michela Tannoia from the Noun Project



Combining concepts

- **Week 1:** HTML elements can have attributes
- **Week 2:** HTML element names can be used as CSS selectors (type selectors)
- **Week 4:** HTML attributes can *also* be used as CSS selectors

Pocket designed by Michela Tannoia from the Noun Project

<>/{} `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 single, unique 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; }
```

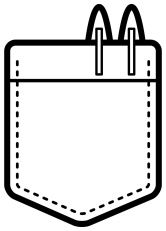
- More specific than a class selector

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



Other uses for `ids`

- Pre-HTML5, used to "label" major areas of the page

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

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

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

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

- Used to make on-page links

```
<a href="#header">Go back to the top</a>
```

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

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

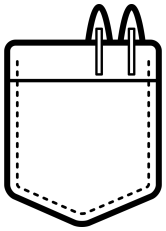
```
div { /* a div with a background... */
    background: #99ff66;
}

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

{ } `:before` and `:after`

- **`:before`** is a pseudo-element before an element
- **`:after`** is a pseudo-element after an element
- We used these in our **`border-box`** reset
- These can be manipulated to simplify border box handling, layouts, add transparent background images to containers, and more

{ } Some more 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

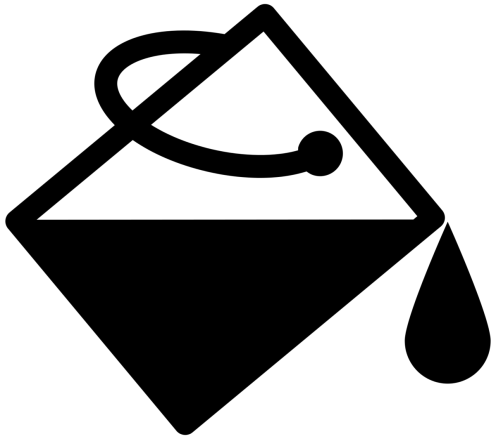


CSS selectors are evolving

- **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

More background styles



Paint Can designed by Alex Valdivia from the Noun Project

{ } background-color review

```
.block {
   color: #000000;
   text-align: center;
   background-color: #bc7384; /* for IE8 */
   background-color: rgb(188,115,132);
}
```

I'm covering up a kitten.
:(

{ } Transparent background-color

```
.block {
   /* text is black and centered */
   color: #000000;
   text-align: center;
   background-color: #bc7384; /* for IE8 */
   background-color: rgba(188,115,132,0.5);
}
```

I'm partially covering up a kitten. :|

{ } 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");
}
```

{ } CSS `background` abbreviations

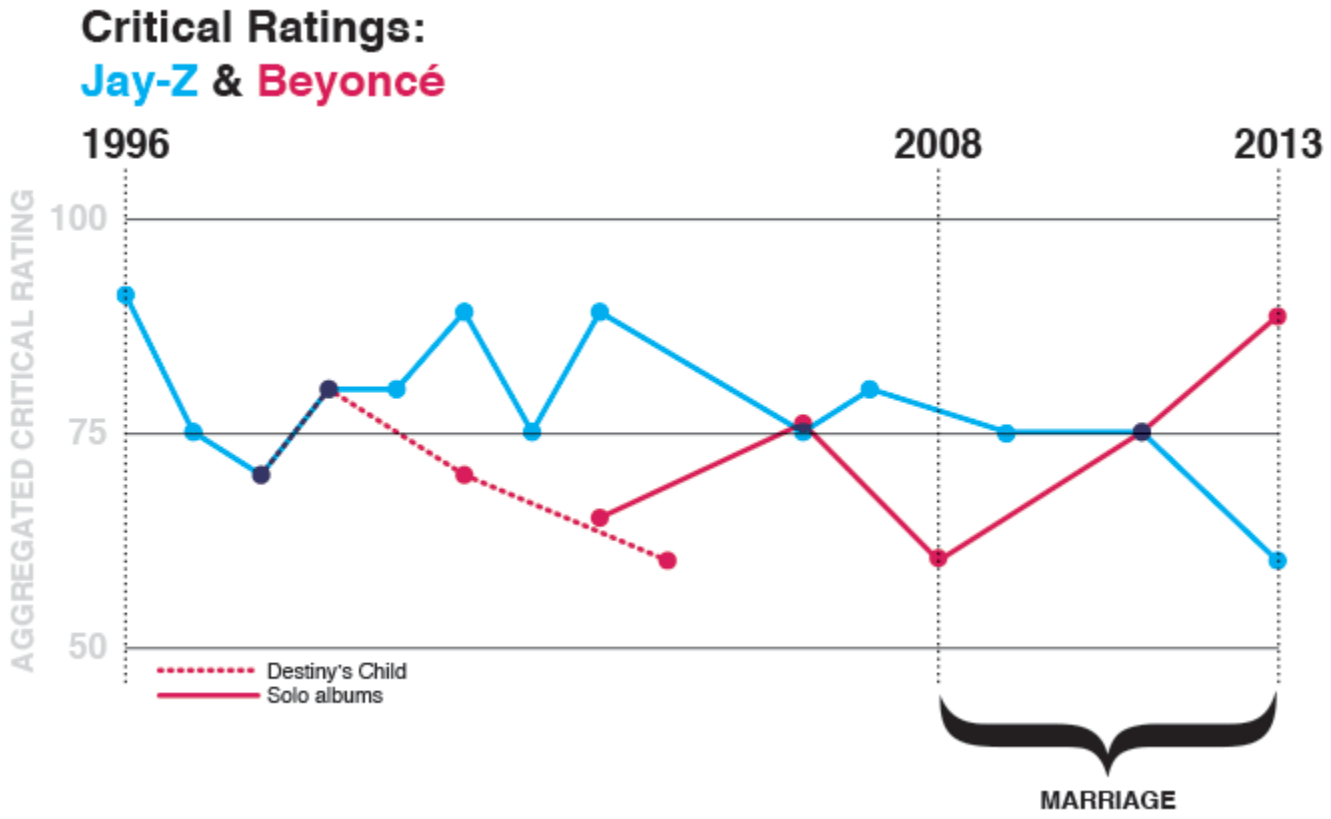
- **`background-color`** and **`background-image`** and can be combined into **`background`**
- Color is always listed first, then the image

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

{ } 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?

{ } Left & right, top & bottom



y-axis

x-axis

Graph by Dorothy Gambrell at Very Small Array

{ } Repeating a background

```
/* repeat the background horizontally */
background-repeat: repeat-x;

/* repeat the background vertically */
background-repeat: repeat-y;

/* don't repeat the background */
background-repeat: no-repeat;
```

{ } Positioning a background

- **`background-position`** values include both the x-axis and y-axis
- x-axis first, y-axis second
- Can be **`left`**/**`right`** **`top`**/**`bottom`** *or* any measurement (pixels, %, ems, etc.)

```
/* position a background in the left top corner */
background-position: left top;
```

{ } Positioning a background ex.

```
/* position a background in the left top corner */
background-position: left top;

/* positioning in the right bottom corner */
background-position: right bottom;

/* position on the left centered vertically */
background-position: left center;

/* position an image completely centered */
background-position: center;
```

{ } More background

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

```
/* 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;
}
```

{ } 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

{ } Background gradients

- Use the x/y axes, **`background-image`**, and colors (text, hex, rgb, rgba) to create opaque and transparent background gradients
- Gradients can be really complex, but we'll try some simple two-color ones
- Check out the CSS-Tricks article on CSS gradients for lots of examples

{ } Background gradients example

```
/* linear two-color gradient that goes from black to white
from top to bottom */
.gradient {
    background-color: black; /* for old browsers */
    background-image: linear-gradient(black, white);
}
```

{ } Using the axes

```
/* left to right */
.gradient {
    background-color: black; /* for old browsers */
    background-image: linear-gradient(to right, black,
    white);
}
/* toward the top right corner */
.gradient {
    background-color: black; /* for old browsers */
    background-image: linear-gradient(to top right, black,
    white);
}
```

{ } Background attachment

```
/* have the background scroll (the default) */
background-attachment: scroll;

/* have the background stick regardless of scrolling */
background-attachment: fixed;
```

...and some others

{ } The magical image background

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

"Homework"

- Practice abbreviating your CSS
- Add focus and hover states for your links
- Add classes and ids to your styles to make different elements unique or to style them in similar ways
- Add fancy backgrounds to your containers
- *HTML & CSS for Web Designers*: Ch. 13-14

Next time

- Browser styles
- Fixed and flexible 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