HTML & CSS: LEVEL 1

Instructor: Beck Johnson

Week 2

SESSION OVERVIEW

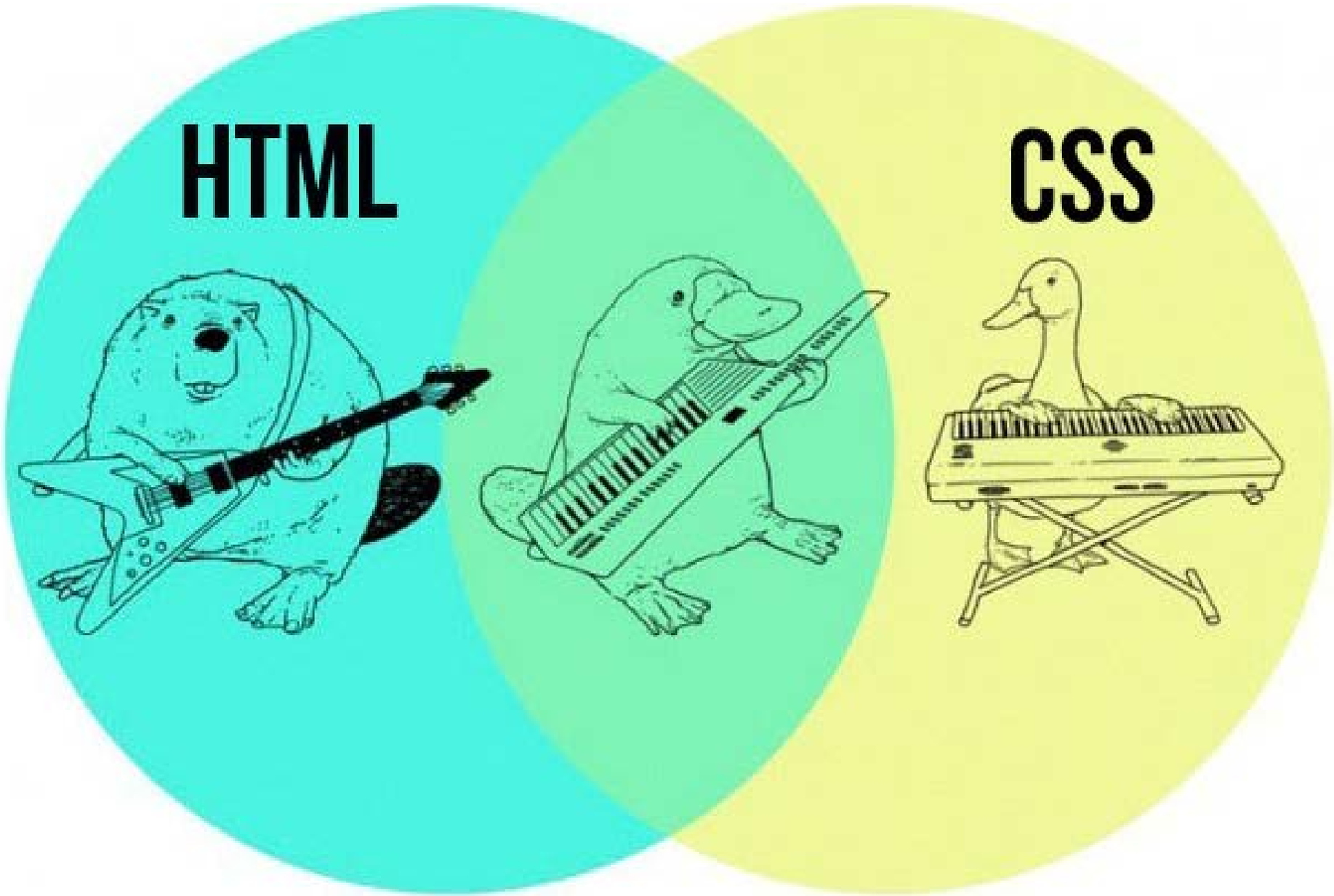
- Week 1 review and questions
- Image format overview
- Optimizing web images
- More CSS

REVIEW!

REVIEW: WEBPAGE COMPONENTS

- **HTML** structures and organizes content
- **CSS** stylizes the content and creates layout
- **Javascript** adds interactivity

HTML + CSS = WEBPAGE



REVIEW: HTML DOCUMENTS

- `<!DOCTYPE html>` tells the browser it's serving an HTML file using HTML5 standards
- `<html>` wraps the whole document
- `<head>` wraps the metadata and styles
- `<body>` wraps the visible content
- Most HTML elements have **opening** and **closing tags**, and some have **attributes**

REVIEW: LAYOUT ELEMENTS

- `<header>` wraps header content
- `<footer>` wraps footer content
- `<nav>` indicates that everything inside is related to navigation
- `<section>` is used to define content sections
    - Often sections have their own heading

REVIEW: HTML CONTENT

- **Headings** create an header/outline

  `<h1>...<h6>`

- **Paragraphs** and **lists** structure text

  `<p>`
  `<ul>`
  `<ol>`

- **Images** and **links** both require **attributes** to work

IMAGES

```
<img src="kitten.jpg" alt="Cute kitten" />
```

- Does not have a closing tag (“self-closing”)
- Two required `attributes`:
  - **src** is where the file lives (local or external)
  - **alt** is a description of the image (used for screen readers, search engines, etc)

LINKS

```
<a href="http://google.com">Google</a>
```

- Creates a link to other pages or websites
- The `href` attribute says where the link should go
- Anything inside `<a>` tags is clickable

REVIEW: ANATOMY OF A CSS RULE

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

- `selector` is the **thing** you want to style
- `property` is the **attribute** you want to style
- `value` is how you want to style it
- Values always end in semicolons (`;`)

REVIEW: EXAMPLE CSS RULE

```
p { color: blue; }
```

- `selector` is `p` (all `<p>` tags in the HTML)
- `property` is `color`
- `value` is `blue` (many color names are supported, or use the hex code `#0000ff`)

{} REVIEW: COMMON FONT PROPERTIES

`font-style:` normal, *italic* or *oblique*

`font-weight:` normal, **bold**, or values of 100, 200, etc (depending on the typeface)

`font-family:` the name of a typeface installed on the user's computer

`line-height:` a number followed by a measurement of the height of a line of that element

`font-size:` a number followed by a measurement of the height of that element's text

{} REVIEW: COLORS

`color:` changes the color of text

`background-color:` sets the background color of an element

Color `value` can be set using **names**, **HEX, RGB,** or **RGBA**

- Color name: `white`
- Hex: `#ffffff`
- RGB: `rgb(255, 255, 255)`
- RGBA: `rgba(255, 255, 255, 0.8)`

QUESTIONS?

WEB IMAGES

WEB-READY IMAGES

- **Minimize** file sizes to help load times in browser
- **Optimizes** images for RGB displays with correct **resolution** for browsers
- **Flattens** layers and removes metadata from graphics

WEB IMAGE TYPES

JPG/JPEG

- Created by the “Joint Photographic Experts Group”
- Millions of colors
- Uses a compression algorithm called **lossy**

GIF

- Stands for “Graphics Interchange Format”
- 256 colors max - fewer colors mean a smaller file
- Animation and off-on transparency

PNG

- Stands for "Portable Network Graphics”
- Millions of colors
- Full alpha transparency

JPGS

JPG PROS:

- small file size
- rich colors

JPG CONS:

- image distortion
- no transparency

JPGS

GIFS

GIF PROS:

- small file size
- transparency
- animations

GIF CONS:

- few colors

PNGS

PNG PROS:

- any amount of transparency
- best image quality

PNG CONS:

- large file size
- IE 7&8 don't support transparency

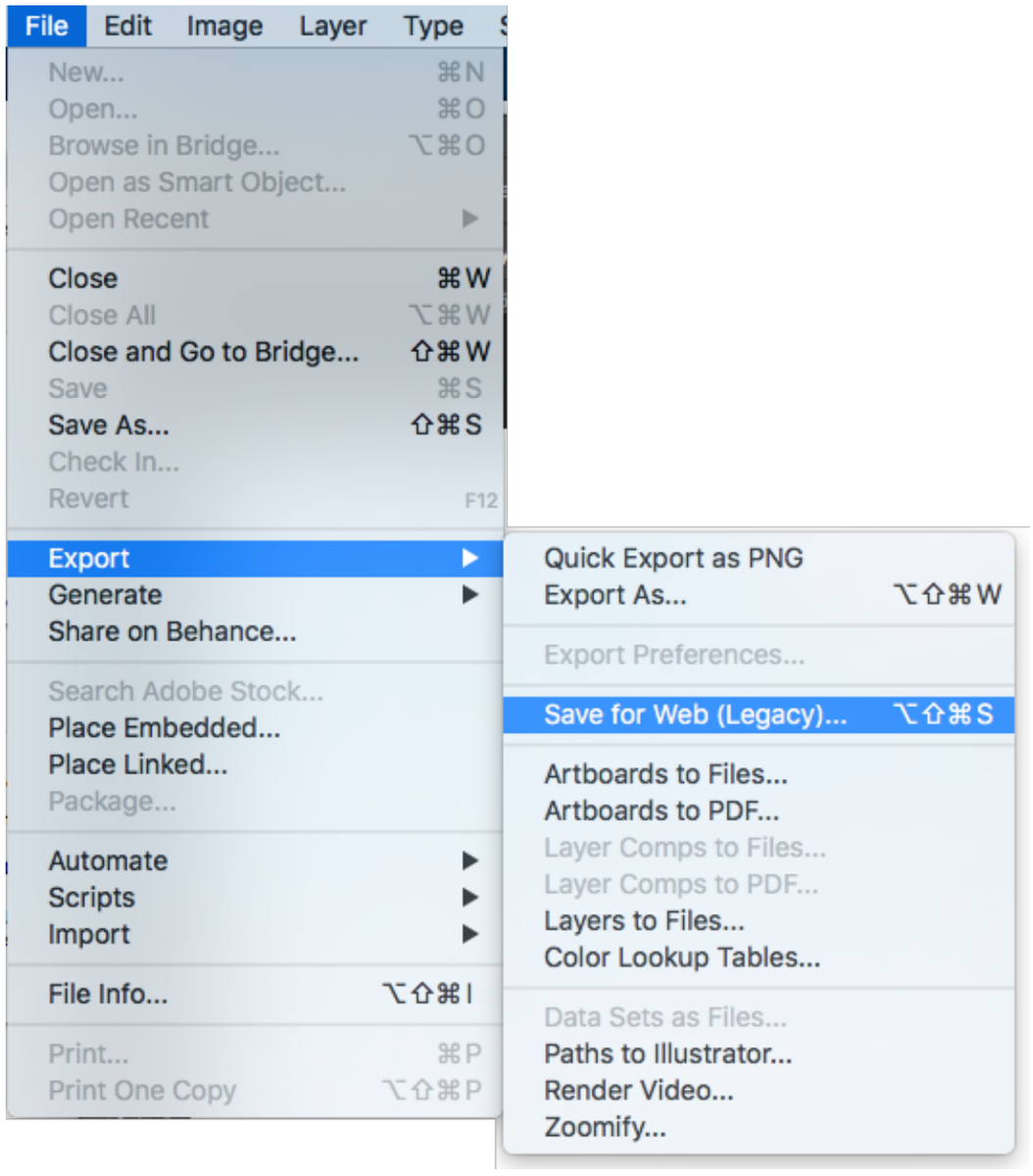
BEST USES:

WEB IMAGES SUCH AS LOGOS THAT INVOLVE TRANSPARENCY AND FADING.

IMAGES IN THE MIDDLE OF THE EDITING PROCESS.

COMPLEX IMAGES LIKE PHOTOGRAPHS IF FILE SIZE IS NOT AN ISSUE.

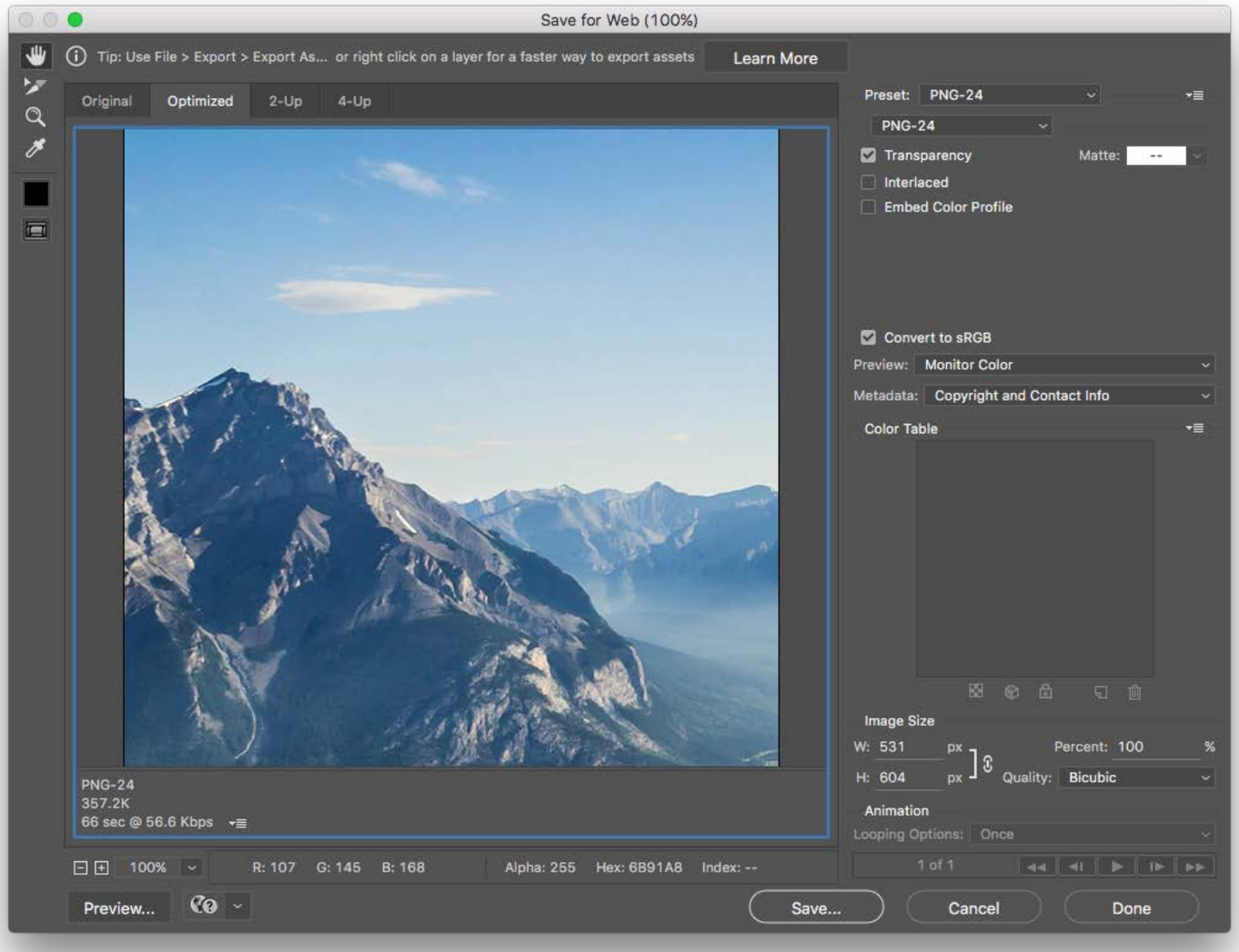
“SAVE FOR WEB” IN ADOBE CS



Adobe products have a “**Save for Web...**” or “**Save for Web and Devices...**” option

“SAVE FOR WEB” IN ADOBE CS

- Click **File > Export > Save for Web... (**or **Export As)**
- Choose a format **(JPEG, PNG 24, or GIF)**
- Adjust image size to max size display
- Save to your images directory



SOME “GOTCHAS”

- Best practice to work in 72 PPI in graphic editor programs. (keeps file sizes down)
- Always work in **RGB** when working with graphics for the web. **CMYK** is for print
- Graphics for **Retina devices** need to be saved out at 2X their “normal” size

BACKGROUND IMAGES

BACKGROUND COLOR REVIEW

```
p {
    background-color: gray;
    color: white;
}
```

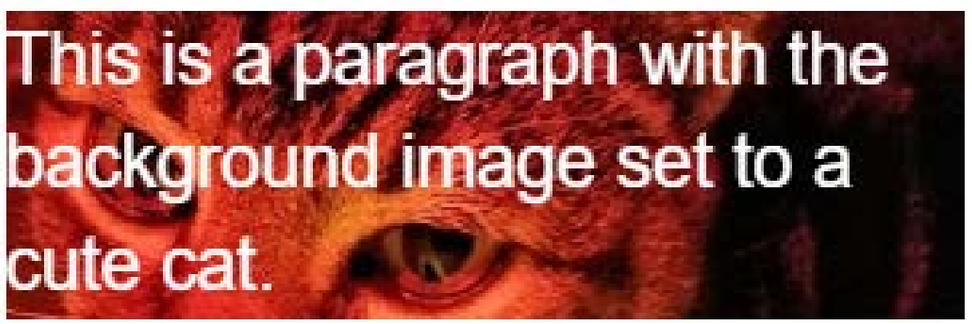
This is a paragraph with the background color set to gray.

BACKGROUND IMAGES

Can set background of an element as an **image** (instead of a color) with the property `background-image`

The `value` is `url("path")`, where `path` is the **relative** or **absolute** path to where the image lives, like this:

```
p {
    background-image: url("images/kitten.jpg");
    color: white;
}
```



BACKGROUND IMAGES

```
p {
    background-image: url("images/kitten.jpg");
    color: white;
}
```

The amount of image that displays in the background is calculated based on image size and container size.

- Make sure to resize images so that the part you want visible is within the “view window”
- Or…

BACKGROUND POSITION EXAMPLES

`background-position:` allows you to move a background image around within its container

- By default, an image is positioned at the top left side of the container

```
section {
        background-image: url("octopus.jpg");
        background-position: top left;
}
```

Image width: 600px by 800px

BACKGROUND POSITION EXAMPLES

Container width: 600px by 200px

```
background-position: top left;
```

```
background-position: center center;
```

```
background-position: bottom right;
```

BACKGROUND REPEAT

`background-repeat:` defines if (and how) the background image will repeat

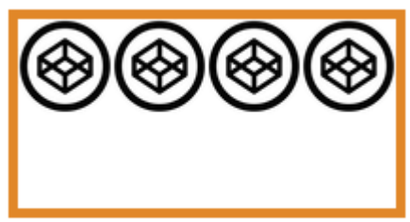
- By default, background images are repeated until they fill the entire container

```
p {
    background-image: url("codepen.gif");
    background-repeat: repeat;
}
```

BACKGROUND REPEAT

`repeat`: tile the image in **both** directions

`repeat-x`: tile the image **horizontally**



`repeat-y`: tile the image **vertically**

`no-repeat`: don't repeat, just show the image **once**

BACKGROUND ATTACHMENT

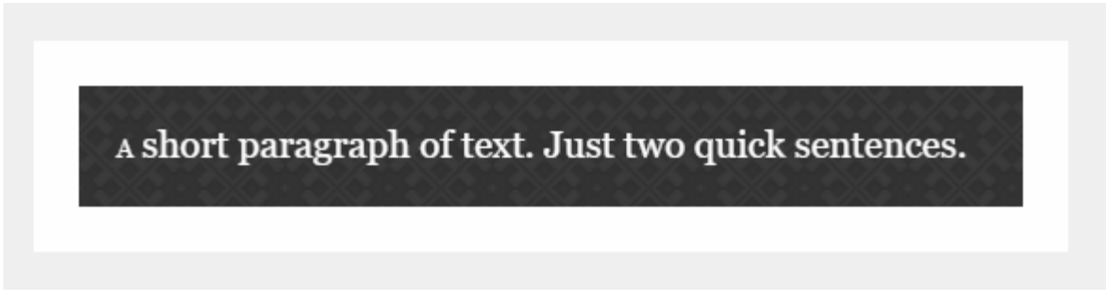
`background-attachment:` images usually scroll with the main view, but setting to `fixed` means the image stays in place when the user scrolls the page

- Difficult to describe, so check out this demo or this demo

```
section {
	background-image: url("pattern.png");
	background-attachment: fixed;
}
```

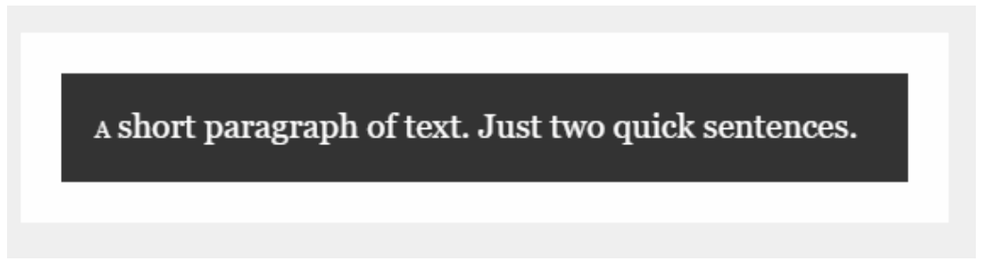
FALLBACK BACKGROUND COLOR

If your background image is dark and your text is light



You may want to specify a `background-color` in **addition** to a `background-image` so that content is visible while the image is loading

So instead of a “blank” area...

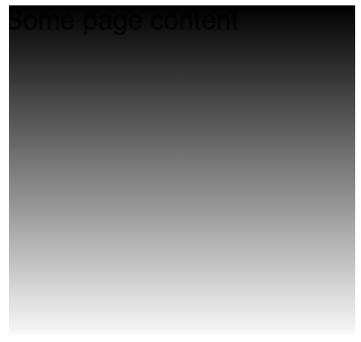


...the user can see content while the image downloads

BACKGROUND GRADIENTS

You can set `background-image` to `linear-gradient`, which is a gradient that the browser draws for you:

```
section { background: linear-gradient(black, white); }
```



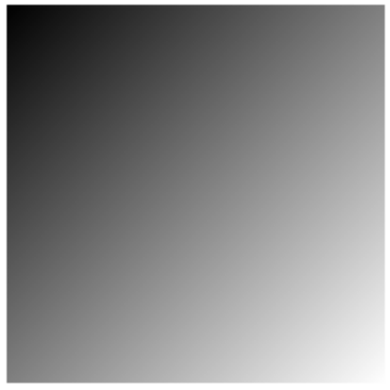
As many colors as you want can be blended, separated by commas:

```
section {
      background: linear-gradient(#ea992e, red, #9e5308);
}
```

BACKGROUND GRADIENTS

By default `linear-gradient` draws from top to bottom, but you can set the gradient to draw at an angle instead by starting with `to`

```
section { background: linear-gradient(to bottom right, black, white); }
```



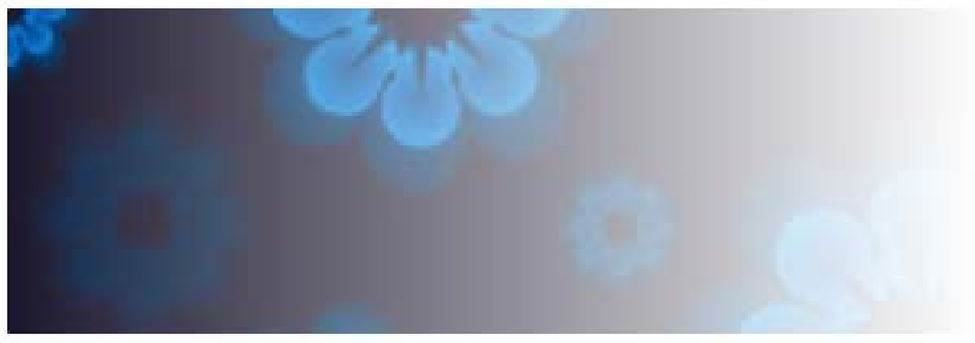
```
section {
      background: linear-gradient(to right, red, #f06d06, yellow, green);
}
```

BACKGROUND GRADIENTS

Background gradients can use rgba colors, meaning you can create a gradient that fades to transparent:

```
body {
	background-image: url("flowers.png");
}
```

```
header {
	background-image: linear-gradient(to
right, rgba(255,255,255,0),
rgba(255,255,255,1));
}
```

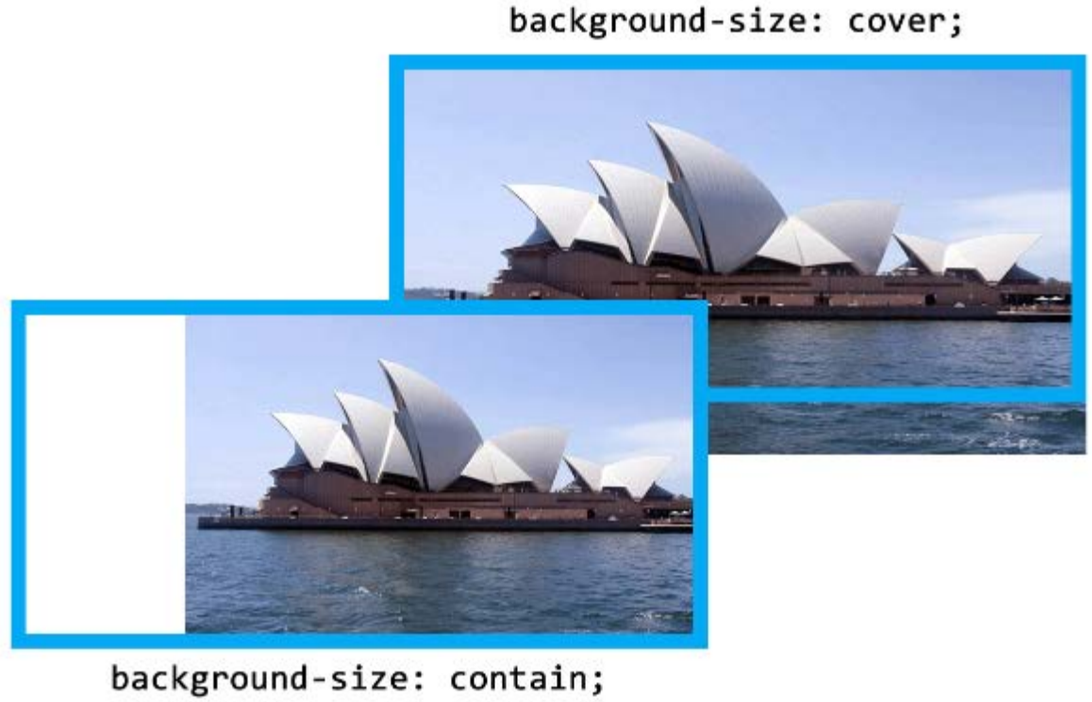


BACKGROUND SIZE

`background-size:` specifies how much of the container that the image covers

`cover`: always cover the entire container (even if that means cropping an edge, or stretching the image)

`contain`: always show the whole image (even if that means there is space on the sides or bottom)



HEIGHT AND WIDTH

To ensure that a background image fully displays, you can set the `height` (and/or `width`) attribute on the element using CSS:

```
header {
    background-image: url("images/hero.png");
    height: 600px;
}
```

HEIGHT AND WIDTH

`height` and `width` can be set on (most) elements to change how much room they take up on the page.

- We'll discuss later why elements like `<a>` and `<em>` don't change when you set their `height` or `width`

The `value` of this property must be a positive number.

- Units are either px or em
- Or you can specify a percentage

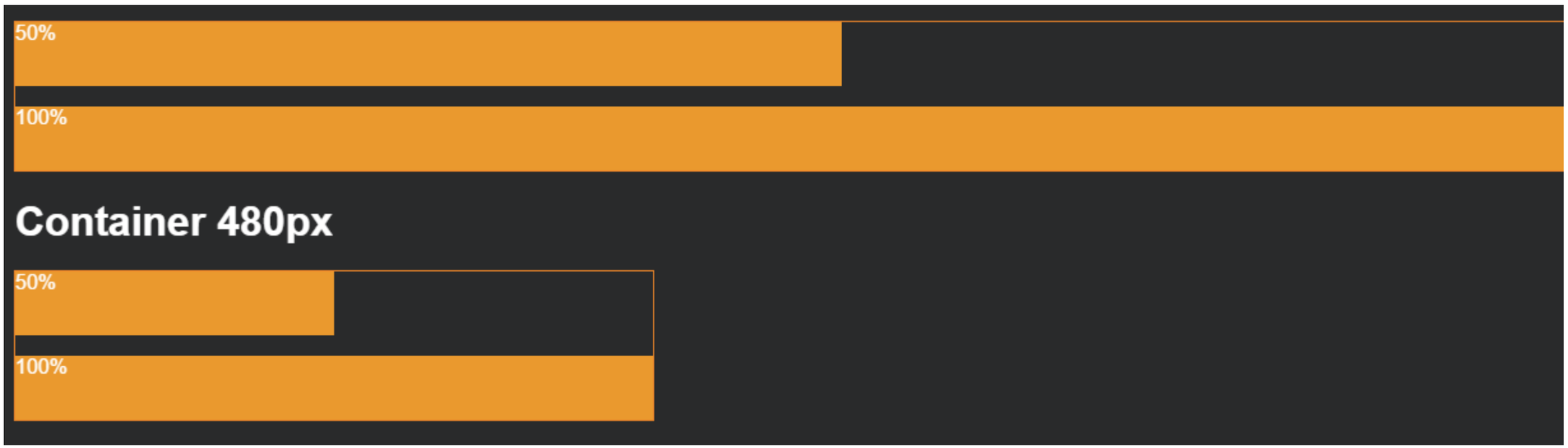
```
header { height: 6em; }
```

HEIGHT AND WIDTH %

Percentage is based on the element's **parent**'s width or height

```
section { width: 50%; }
```

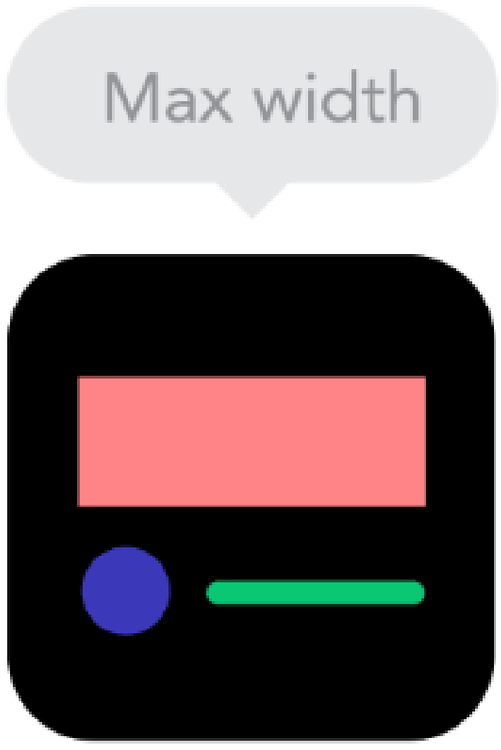
If that `section` were inside a 480 px wide container, it would end up being 240 px wide.

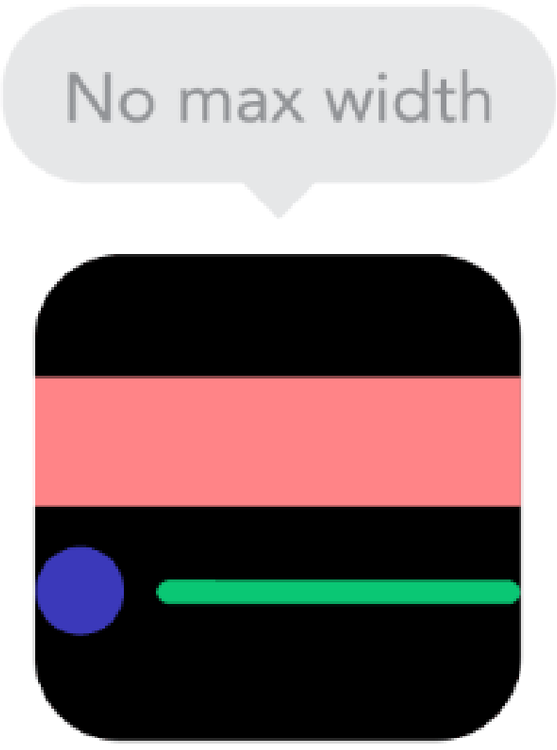


MAX-HEIGHT AND MAX-WIDTH

To ensure an element is **never larger** than a certain value, use `max-height` or `max-width`

- Typically used to make sure content (particularly text) doesn't spread too far out on large monitors



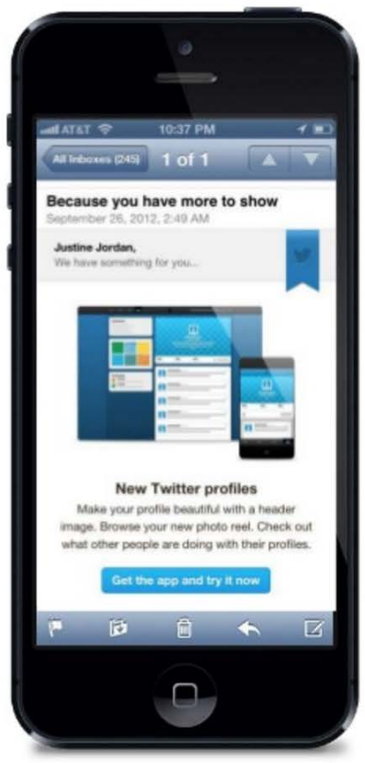


MIN-HEIGHT AND MIN-WIDTH

Specify `min-height` or `min-width` if you want to ensure an element is **never smaller** than a certain value.

- This is especially helpful if your size is “dynamic” (based on percentage) and will vary depending on device

```
img {
      width: 50%;
      min-width: 350px;
}
```



MIN-MAXING

`height` and `width` fix an element to a specific size regardless of display size

- If `width` is wider than the display – scroll bars
- If `width` is smaller than the display – content may wrap even if there is room

`min-height`, `min-width`, `min-height`, and `min-width` allow elements to change when the display size changes, but still allow some control over presentation.

MIN-MAXING

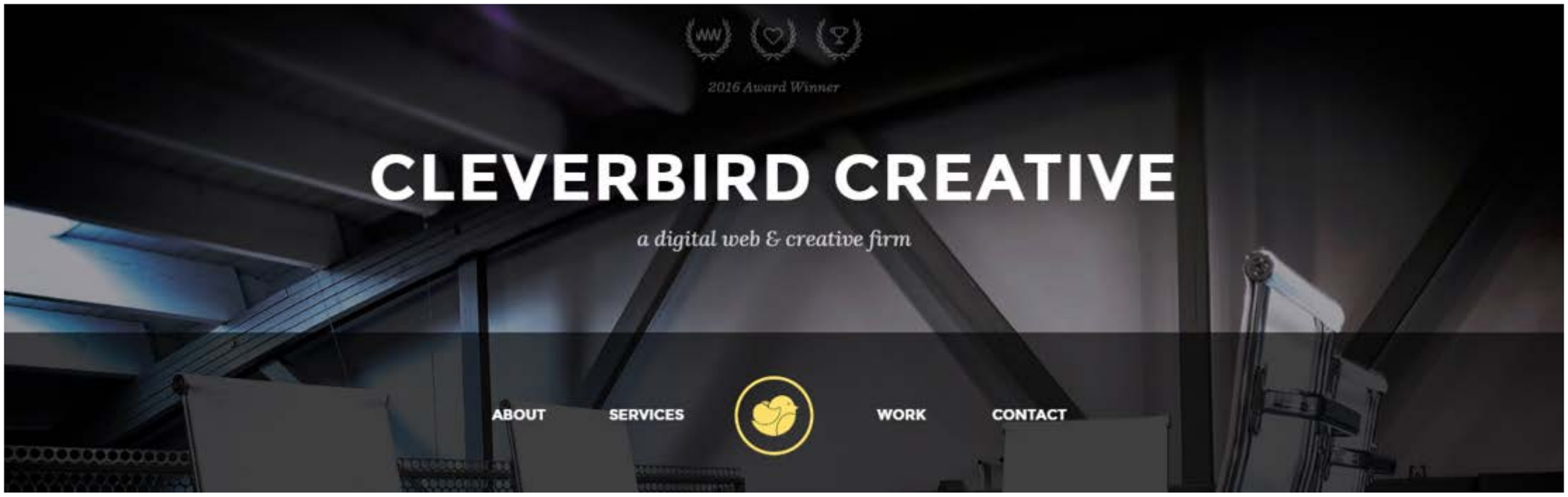
You can choose to set only `width` and/or `height`, only `min-width`/`min-height`, and only `max-width`/`max-height` – or any or all of them, depending on your design.

For example, this `section` will expand up to 500 px wide, and then get no bigger. If you shrink your browser, it will shrink until its 100 px wide, and then get no smaller.

```
section {
      min-width: 100px;
      max-width: 500px;
}
```

NOT ALL HEROES WEAR CAPES

A common use of `background-image` is to create a "hero" image with text overlaying it



PRACTICE TIME!

MAKE A HERO

Re-save the local images for your website in an optimized format using Photoshop.

- How much smaller are the files?

Add a “hero image” to your site.

- Play around with a bunch of the background properties we learned to make your hero look pretty

- Try setting a `width` and `height`. What happens when you resize your browser window? Change to `min-width` – what changes?

{} CSS IN MULTIPLE PLACES

So far, we've been making CSS changes directly on a single webpage, in the `<head>` element.

- These **internal styles** only apply to that page (but affect every element on that page that is styled)

{} CSS IN MULTIPLE PLACES

Can also add **inline styles** to a single element by using the `style` attribute in HTML markup

```
<p style="color: red">This paragraph is
special.</p>
```

- Inside the `style` attribute, use the same syntax as CSS (selector: value)
- Typically discouraged, because it can be hard to maintain

{} CSS IN MULTIPLE PLACES

The most common way to use CSS in “real life” is to use an **external stylesheet**.

- CSS lives in a separate .css file
- The same stylesheet can be included on multiple pages
- A single page can include multiple stylesheets

{} LINKING TO EXTERNAL STYLESHEET

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

- Tells the browser to find and load the styles.css file from the css directory
- The `rel` attribute stands for "relation" - in this case, this link's relationship to the document is "stylesheet"
- This tag goes inside the `<head>` element
- Should be on every page that needs the styles

{} THE "CASCADING" PART

The beauty of CSS is being able to create styles and then override them when you want to customize the look of your pages.

There are **3 rules** for determining how styles get applied:

- Styles are applied from **far** to **near**
- Styles are applied from **top** to **bottom**
- **Children** elements are more specific than **parents**

{} FAR TO NEAR

Styles that are “closer” to the elements they style take precedence.

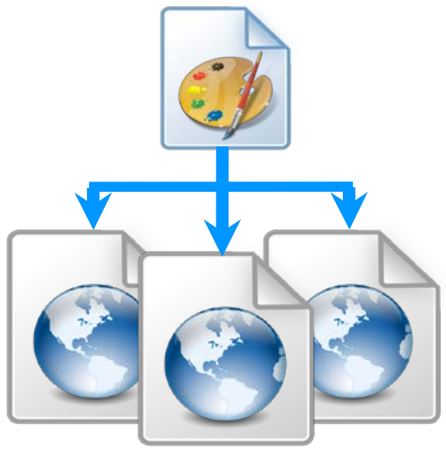
- Browser defaults
- External styles (in a .css file)
- Internal styles (in the <head>)
- Inline styles (directly on an element)

Less Specific

Most Specific

{} FAR TO NEAR

Browser default



External styles
(in a **.css** file)

Internal styles
(in the <head>)

Inline styles
(directly on an element)

Less specific to more specific

{} TOP TO BOTTOM

CSS rules are applied sequentially

If the same property is styled multiple times for the same selector, **the last one wins**

```
p { color: #2f4251; }

p { color: #daa645; } /* this one wins */
```

{} CSS COMMENTS

Just like HTML, CSS can have **comments.**

```
<style>
    /* I am a CSS comment! */
</style>
```

{} CHILDREN ARE SPECIFIC

Children elements usually **inherit** styles from their parents but can **override** parents with their own styles

```
body { color: #2f4251; } /* parent */

p { color: #daa645; } /* child */
```

All text in the body that is NOT a paragraph will be dark gray. Paragraphs will be mustard-colored (even though paragraphs are also in body)

{} SELECTORS CAN BE MORE SPECIFIC

If one style is **more specific** than another, it takes precedence

```
p { color: #daa645; } /* all paragraphs */

a { color: #e7c0c8; } /* links in general */

p a { color: #c4fe46; } /* links in paragraphs */
```

PRACTICE TIME!

{} EXTERNAL STYLESHEETS

Copy and paste the styles from inside `<style></style>` in index.html into a new file.

- Remember best practices for file organization

Save your new files as **styles.css,** and save in a new css folder.

Remove the `<style></style>` tags from index.html.

Create a link to your new stylesheet on all of your webpages.

- Does everything still look the same?

CSS PSEUDO CLASSES

{} PSEUDO SELECTORS

A **CSS pseudo-selector** specifies a special state of the element we want to style

- It's "pseudo" because the element doesn't exist in markup – may change based on user interaction or position relative to other elements

We already saw one used for links:

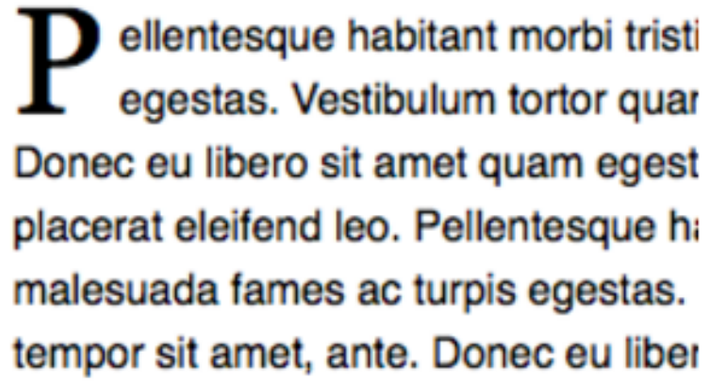
```
a:hover { color: rebeccapurple; }
```

{} OTHER PSEUDO SELECTORS

`: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

```
p:first-letter {
      font-size: 3em;
      float: left;
}
```



{} OTHER PSEUDO SELECTORS

`:focus` styles an element that has the current keyboard focus, from either click or tab

`:checked` styles a selected radio button or checkbox

PRACTICE TIME!

ASSIGNMENT

Using CSS, style links that appear in the header differently than other links.

- Use the rules of **child selectors** to identify those elements.

Add a style **override** to the head of one page that contradicts a style in your stylesheet. What wins?

Add a **internal style** to an element.

Style a link pseudo-class.

Style another type of pseudo-element.

“HOMEWORK”

- Practice!

- Optional: read chapters 10-12 and chapter 16 of HTML and CSS: Design and Build Websites

- Check out the CSS Zen Garden for inspiration on how simply changing CSS can change the entire look and feel of a page