HTML & CSS: FUNDAMENTALS OF DEVELOPMENT

Instructor: Beck Johnson

Week 2

SESSION OVERVIEW

- Week One review and questions

- Overview of CSS – font styling, colors, alignment

- File organization

- Version control and code sharing with Git

- Background images and gradients with CSS
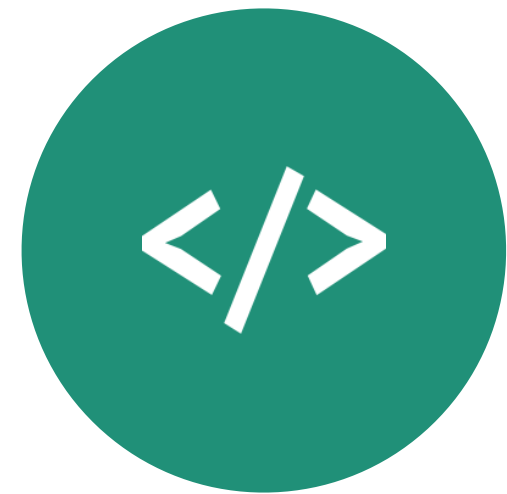
REVIEW!

REVIEW: WEBPAGE COMPONENTS

HTML

Structures and organizes content

CSS

Styles the markup and creates layout

JAVASCRIPT

Brings content and design to life

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

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

```
<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)
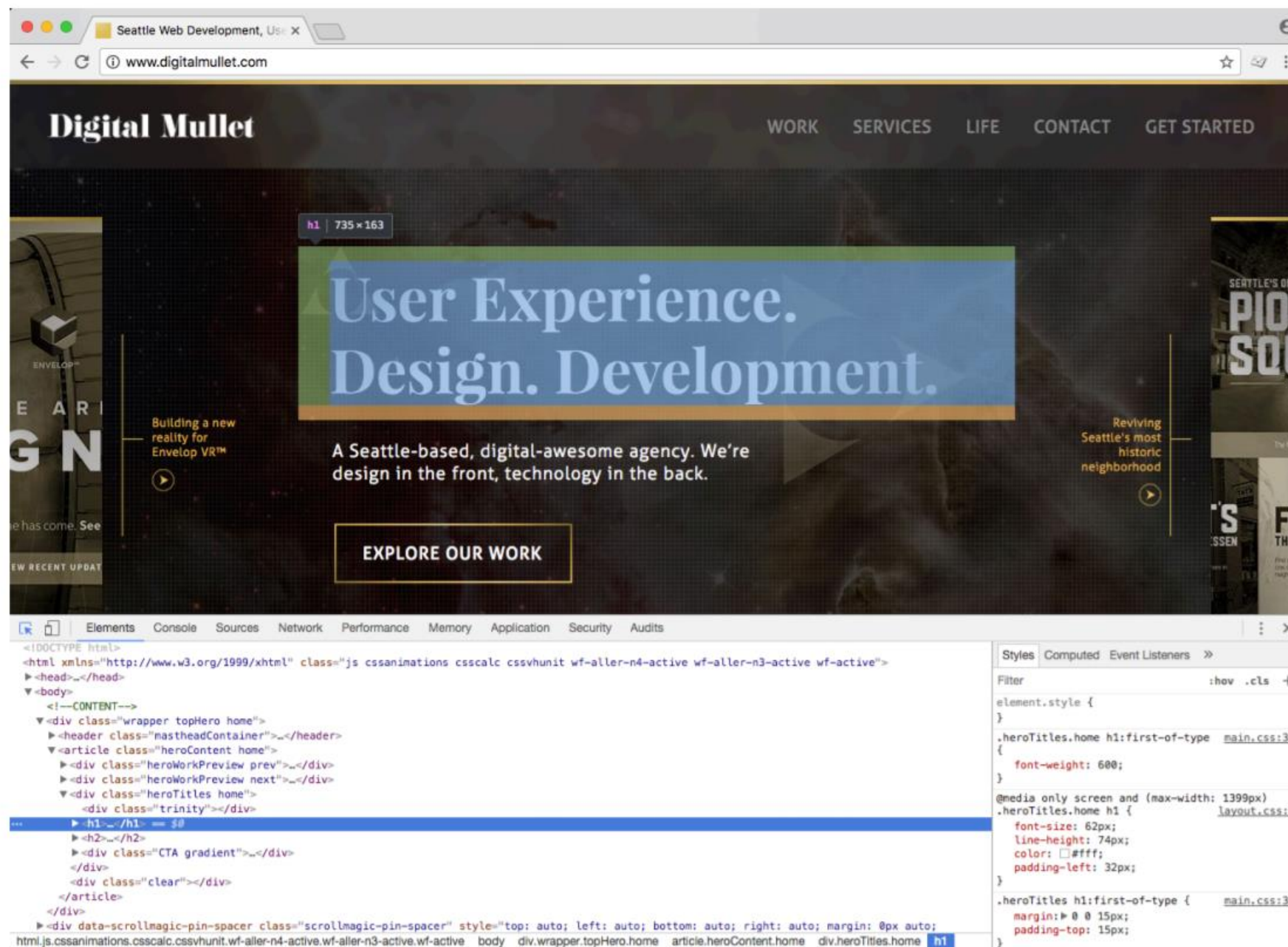 - **alt** is a description of the image (used for screen readers, search engines, etc.)

REVIEW: 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: DEV TOOLS

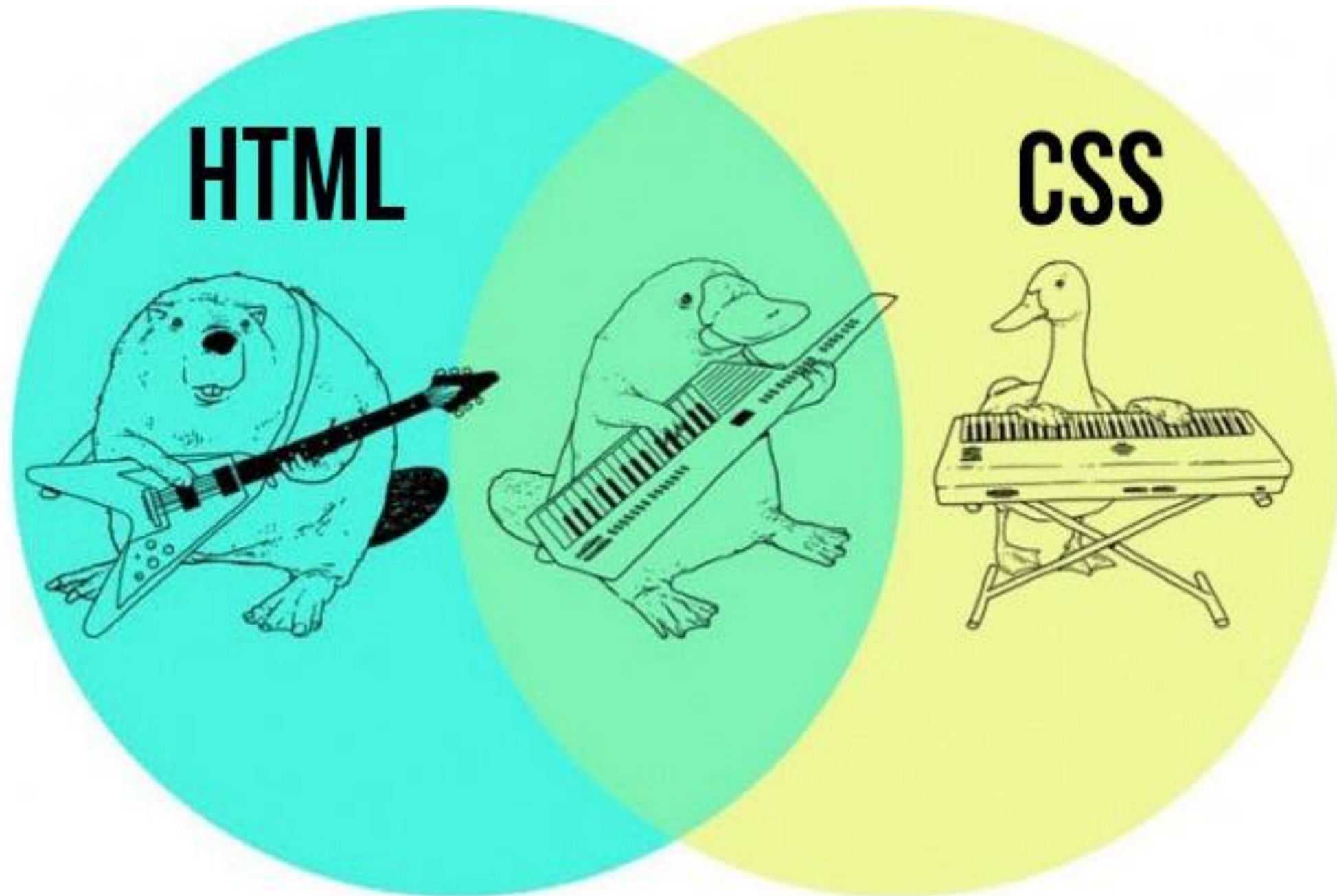Right-click > Inspect, or hit the F12 key

QUESTIONS?

CSS3

HTML + CSS = WEBPAGE

CASCADING STYLE SHEETS

- CSS is a language for specifying how documents are presented to users

- Allows us to override the browser's default presentation styles with a custom version

- Provides consistent and scalable ways to style single elements, single pages, or entire websites

- Separates look and feel from content/markup

CSS: FAIR WARNING

- There is **A LOT** you can do with CSS

- We won't get anywhere close to covering everything!

- We will cover CSS for text styles, colors, positioning, layout, and a couple of extras

WHY USE CSS?

- Helps you avoid duplication by keeping styles in one place (one external stylesheet)

- Makes style maintenance easier - for example, update the font for the whole site in one line of code!

- Separating presentation from content enforces style consistency and allows flexibility

CSS GOES WHERE?

CSS is a different type of language than the HTML we did last week, and has its own syntax

- CSS can go directly in your HTML file, inside a `<style></style>` element

- You can also create a ".css" file that can be linked to your HTML page

`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 (`;`)

So!

```
<style>
    p { color: blue; }
</style>
```

"All paragraphs will have blue text "

```
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)

```
p {
  color: blue;
  font-size: 14px;
}
```

- Multiple `properties` can be defined for a single `selector`, each separated by a semicolon (`;`)
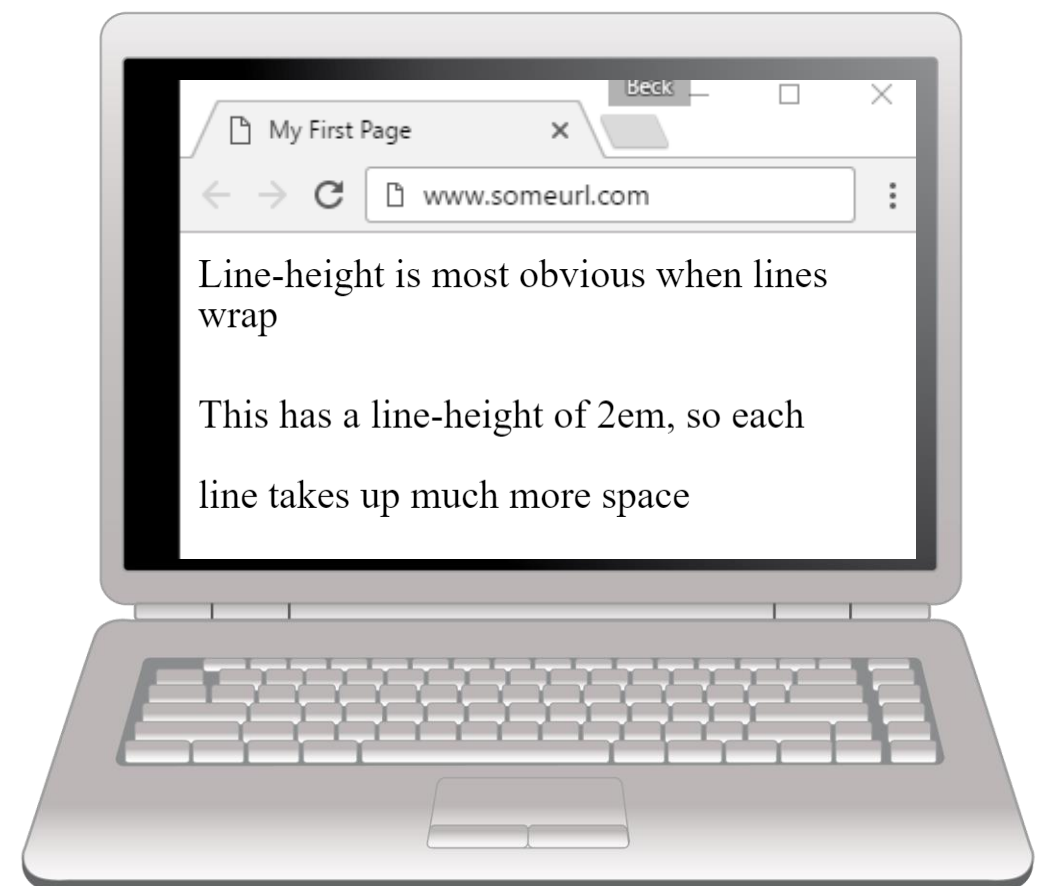
`line-height:` a number followed by a measurement of the height of a line of that element, in ems (em) or pixels (px)

- similar to **leading** in typography

`p { line-height: 1.4em; }`

`font-size:` a number followed by a measurement of the height of that element's text in ems (em) or pixels (px)
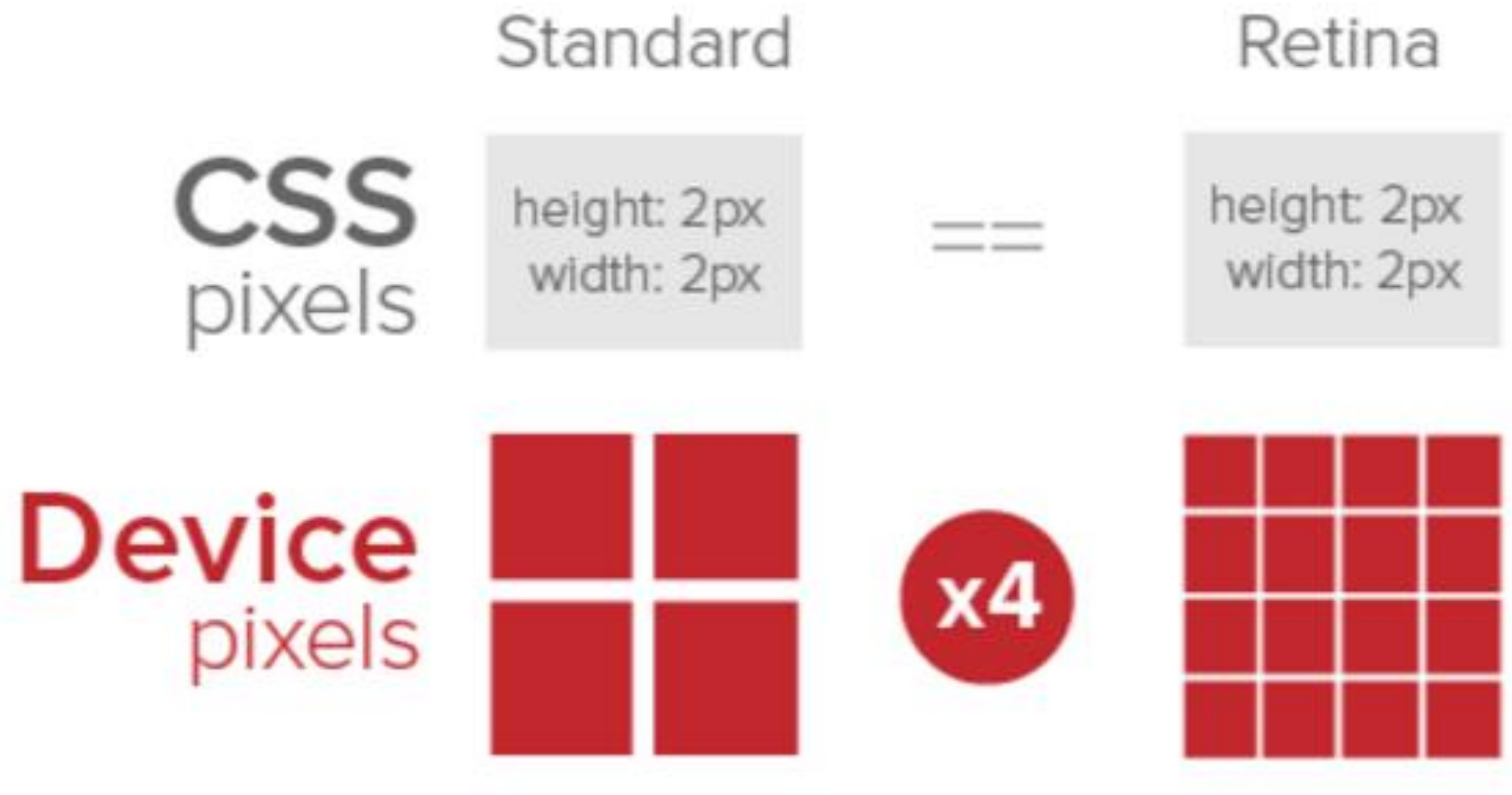
`p { font-size: 14px; }`

The two standard units for sizing in CSS are **px** and **em**

- **px** is an abstract unit that isn't related to font height and isn't a physical unit of measurement

  - Devices with more PPI (pixels per inch) may use several "device" pixels when displaying a 1px line

  - That means that px size varies by device, but should always look "about the same"

Device pixels
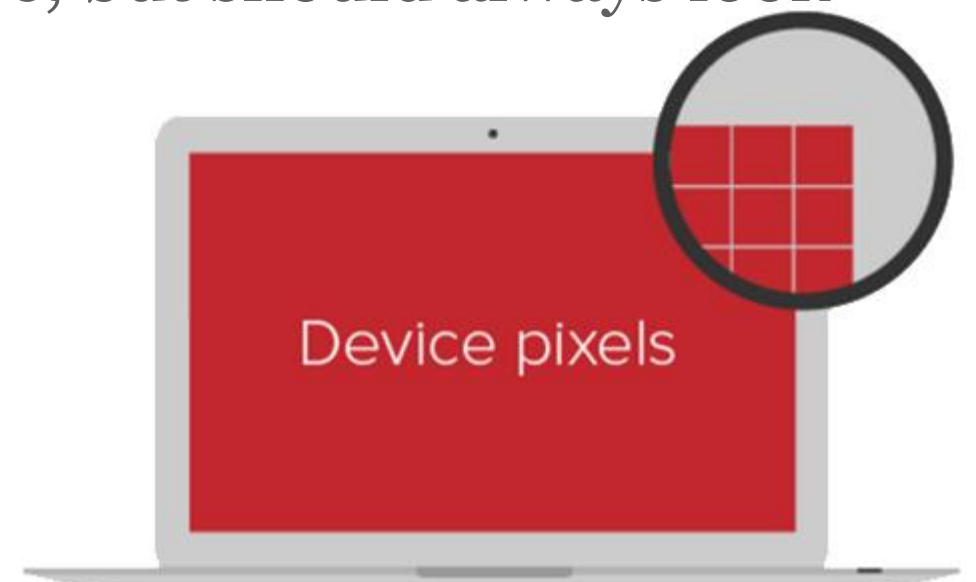
- **em** refers to the height of the letter 'm' of the font being used

  - This unit of measurement is a description of the **relative** size between this element and its parent

  - So `h2 { font-size: 2em; }` means the header is 2 times as big as the letter 'm' of the default font in your html document

Because em is **relative**, that means that if the parent's font size is increased, the children will get bigger too.

	body { font-size: 100%; }	body { font-size: 120%; }
font-size: 1em	The quick brown fox	The quick brown
font-size: 12px	The quick brown fox	The quick brown fox

{ } COMMON FONT PROPERTIES

font-style: normal by default – can also be italic or oblique

font-weight: normal by default – can also be bold, or values of 100, 200, etc. (depending on the typeface)

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

```
p {
    font-family: Arial, Helvetica, sans-serif;
}
```

- The W3 has a list of "web safe" fonts that most people will have installed locally

text-transform: changes font casing. Can be uppercase (all caps), lowercase, or capitalize (first letter of all words capitalized)

letter-spacing: change font kerning by specifying the space between letters in ems (em) or pixels (px)

```
p {
    font-family: Arial;
    text-transform: uppercase;
    letter-spacing: 2.4px;
}
```

LOOKS LIKE THIS

{ } 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**

  - Name: `white`

  - Hex: `#ffffff`

  - RGB: `rgb(255, 255, 255)`

  - RGBA: `rgba(255, 255, 255, 0.8)`

```
p {
    color: black;
}


p {
    background-color: #000000;
}


p {
    background-color: rgb(0, 0, 0);
}
```
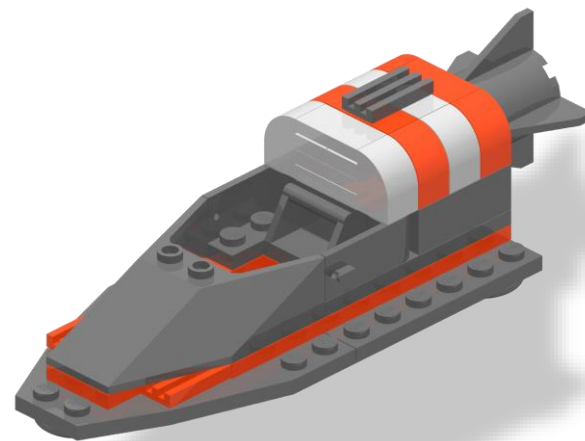
```css
a { color: blue; }

a:visited { color: gray; }

a:hover { color: purple; }

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

Let's inspect a [live demo](#) of how this looks

You can change the alignment of text using the
text-align property.

Values:

- center
- left
- right
- justify

```
h1 { text-align: center; }
```

- You can add multiple `selectors` to a CSS rule

- You can add multiple `properties` to a CSS rule

```
<style>
    ul {

        color: #ffffff;

    }


    ol {

        font-size: 16px;

        font-weight: bold;

        color: #444444;

    }
</style>
```

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

- Start with /*

- End with */

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

PRACTICE TIME!

Add a `<style></style>` section in the `<head>` on your page

Make some style changes using CSS

- Consider changing font color, font family, font size, link color, text alignment, and background colors

FILE ORGANIZATION

FILE ORGANIZATION

- If you structure your site correctly, you are one step closer to faster updates

    - Structure should be not just for you, but for anyone who might use, need or want any of your files (images, scripts, stylesheets, etc)

- The next person to work on or look at your code will be able to understand what you've done and where to find things

FILE ORGANIZATION

Typical files in a website include:

HTML files (.html)
CSS files (.css)
Javascript files (.js)
Images (.png, .jpg, .gif)

- HTML should usually go in the **main** (root) directory

- Make **subdirectories** for media, CSS, and Javascript files

FILE NAMING RULES

- Use a consistent naming convention when naming files and folders

  - For example, always all lowercase, or words always separated by dashes, etc

- Capitalization matters
  - INDEX.html is not the same as index.html

- Use only letters, numbers, hyphens (-) or underscores (_)

- No spaces in file names

- Comment your files – especially if you have unfinished development code, or if you think you may forget *why* you made the decision you did

```
.viewmore {
    max-height: 2.85714286em; /* line-height of the paragraph x 2 */
}
```

- Indent your code (trust me)

CODE INDENTATION

The Javascript code on the right doesn't have consistent formatting, and is hard to read

```javascript
 5        swapImages(true);
 6
 7    var t = false;
 8    $(window).on('resize', function () {
 9
10            if (t !== false) {
11    clearTimeout(t);
12        }
13
14     t = setTimeout(swapImages, 200);
15    });
16    });
17
18    function swapImages(flag) {
19      $('img[data-lg-src]').each(function () {
20          var $img = $(this);
21        var a = $(window).width();
22        if (flag) {
23            $img.attr("data-sm-src", $img.attr('src'));
24            }
25
26
27       if (a >= 769) {
28              $img.attr('src', $img.attr("data-lg-src"));
29        } else if (a >= 481) {
30                $img.attr('src', $img.attr("data-md-src"));
31          } else {
32            $img.attr('src', $img.attr("data-sm-src"));
33            }
34      });
35        }
```

CODE INDENTATION

This code is indented, so it's easier to see the "if/else" logic

Comments are added to explain decisions

```
 5          swapImages(true);
 6
 7          // On resize, swap in the correct image (after waiting for event thrashing to halt)
 8          var timer = false;
 9          $(window).on('resize', function () {
10
11              if (timer !== false) {
12                  clearTimeout(timer);
13              }
14
15              timer = setTimeout(swapImages, 200);
16          });
17      });
18
19      function swapImages(setMobileImages) {
20
21          $('img[data-lg-src]').each(function () {
22
23              var $img = $(this);
24              var windowSize = $(window).width();
25
26              if (setMobileImages) {
27                  $img.attr("data-sm-src", $img.attr('src'));
28              }
29
30              if (windowSize >= 769) {
31                  $img.attr('src', $img.attr("data-lg-src"));
32              } else if (windowSize >= 481) {
33                  $img.attr('src', $img.attr("data-md-src"));
34              } else {
35                  $img.attr('src', $img.attr("data-sm-src"));
36              }
37          });
38      }
```

This is one popular way to indent CSS

- Starting bracket is on the same line as the selector

- Each property is on its own tabbed line

- Ending bracket is on its own line

```css
 8  html, body {
 9      background-color: #fff;
10      color: #4a3c31;
11      font-size: 16px;
12      height: 100%;
13  }
14
15  body, p {
16      font-family: "Lato", Arial, Helvetica, Lucida Grande, Sa
17      line-height: 1.5;
18  }
19
20  a {
21      color: #4a3c31;
22      text-decoration: none;
23  }
24
25  img {
26      max-width: 100%;
27  }
28
29  strong {
30      font-family: "Lato-Bold", Arial, Helvetica, Lucida Gran
31  }
```

PRACTICE TIME!

Create a folder for your images, and move all images there

- Fix the paths in all your `<img src>` tags so that images show like before

- Remember the different between **relative** and **absolute** paths

Prettify your CSS and HTML so that it's easy to read

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)

You 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

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 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**

Styles that are "closer" to the elements they style take precedence.
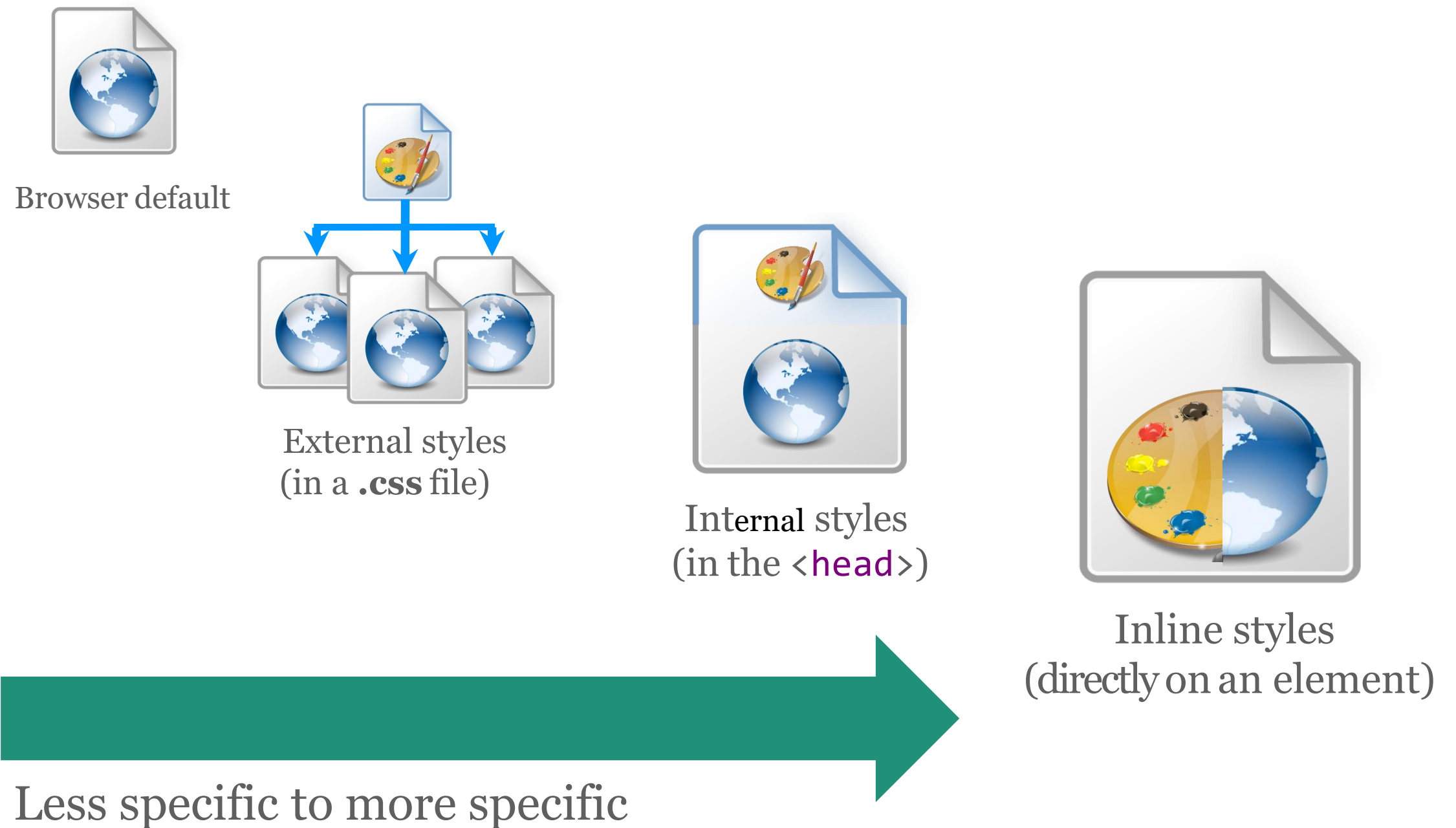
- Browser defaults

- External styles (in a .css file)

- Internal styles (in the `<head>`)

- Inline styles (directly on an element)

Less Specific

↓

More Specific

Browser default

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

Internal styles
(in the `<head>`)

Inline styles
(directly on an element)
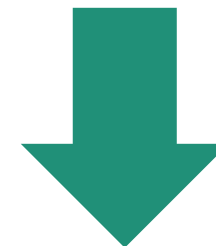
Less specific to more specific

CSS rules are applied sequentially

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

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

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

{} CHILDREN ARE SPECIFIC

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

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

b { color: #e7c0c8; } /* bold text in general */

p b { color: #c4fe46; } /* bold text in paragraphs */
```

PRACTICE TIME!

{ } EXTERNAL STYLESHEETS

Create a folder for css, then create a new file in that folder called **styles.css**

- Copy and paste the styles from inside `<style></style>` into that file

- Remove the styles from the `<style></style>` tag

Create a link to your new stylesheet on all of your webpages:
`<link href="css/styles.css" rel="stylesheet">`

- Does everything still look the same?

VERSION CONTROL

In modern development, most websites are a team effort.

Version control systems (VCS) allow multiple people to work on the same file with less risk of overriding changes.

git   SUBVERSION   ⋈ TFS

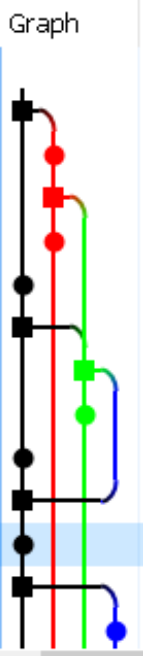**git** is a free, open-source version-control system

Git is software that tracks any changes that you (or anyone else) make to a project

- A project contains every file and subdirectory inside a folder

- The "master copy" of the project is stored on a remote server

- You can download a copy of the project from the server, create new versions of the project, and also have access to every version (of every file!) saved along the way

⚒ GIT

When you upload changes to the remove server, you can add a friendly message indicating what you changed:

Graph	Actions	Message	Author
		Working tree changes	
		`dev` `origin/dev` **Merge branch 'dev' of http://tfs.svccorp.com/tfs/SCICollection/Dev... Beck Johnson**	Beck Johnson
		Moved to configuration helper config value reading	Dos Reis, Maximiliano
		Merge branch 'dev' of http://tfs.svccorp.com/tfs/scicollection/DevOps/_git/DMCom	Dos Reis, Maximiliano
		Updated SEO Redirects for Obituaries Search	Dos Reis, Maximiliano
		fix formatting for result count on obit search page	Beck Johnson
		Merge branch 'dev' of http://tfs.svccorp.com/tfs/SCICollection/DevOps/_git/DMCom into dev	Beck Johnson
		Merge branch 'dev' of http://tfs.svccorp.com:8080/tfs/scicollection/DevOps/_git/DMCom	de Hormaechea, Sebastián
		Obituaries Changes - Refactor & enhacements & fix	de Hormaechea, Sebastián
		make associate h1, NOT location name (whoops)	Beck Johnson
		#12241: restore fix for horizontal scroll bar, fix header aligment, re-widen search box	Beck Johnson
		#12863 - make day/month shorter on Recent Obits cards	Beck Johnson
		Merge branch 'dev' of http://tfs.svccorp.com/tfs/scicollection/DevOps/_git/DMCom	Koleff, Martín
		Fixed issue	Berra, Ignacio

SHA-1: 44e682c7da5e5f2fd3b82a18290ae1c689fcb7ec

* #12863 - make day/month shorter on Recent Obits cards

Path	Extension	Status	Lines added	Lines removed
DM3/src/Feature/DignityMemorial/Location/code/Views/Location/Detail/LocationRecentObituaries.cshtml	.cshtml	Modified	1	1

⚒ GITHUB

GitHub.com is a free online hosting provider for code that is tracked using Git

- Code is stored publicly (unless you want to pay extra)

- Very popular storage place for open-source projects

- The website for this class is hosted by GitHub!

GitHub

BACKGROUND IMAGES

BACKGROUND COLOR REVIEW

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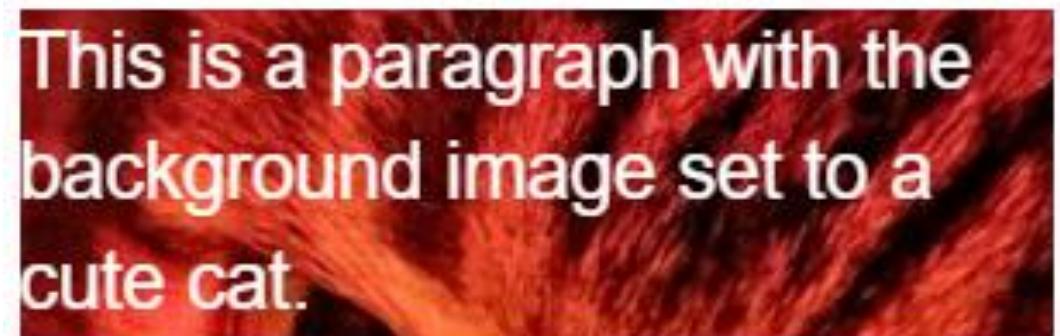
```
p {
    background-image: url("images/kitten.jpg");
    color: white;
}
```


This is a paragraph with the background image set to a cute cat.

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

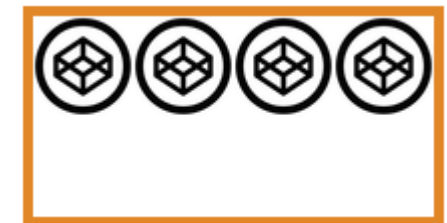- By default, background images are repeated until they fill the entire container

```css
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:` 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](#)
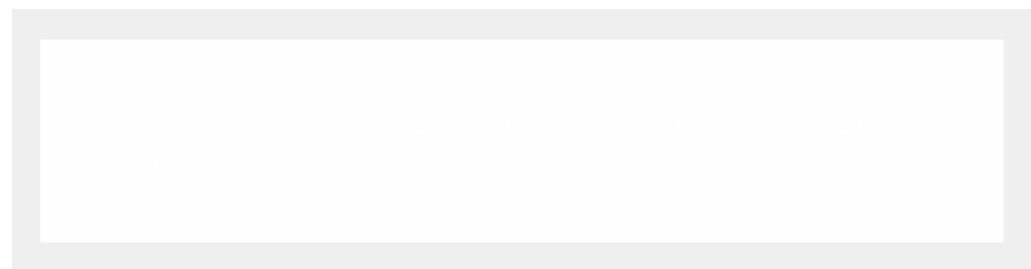
```
section {
    background-image: url("pattern.png");
    background-attachment: fixed;
}
```
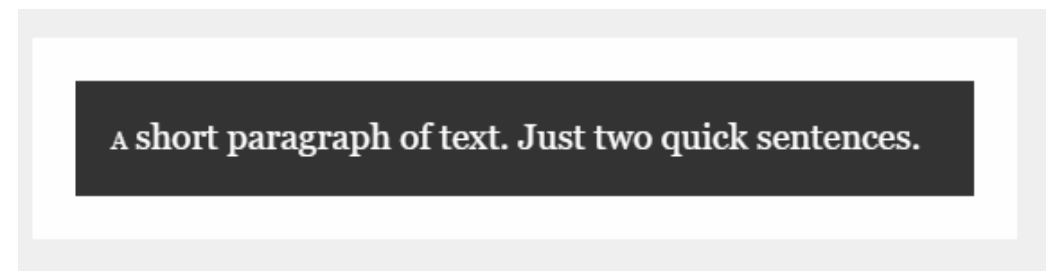
FALLBACK BACKGROUND COLOR

If your background image is dark and your text is light

A short paragraph of text. Just two quick sentences.
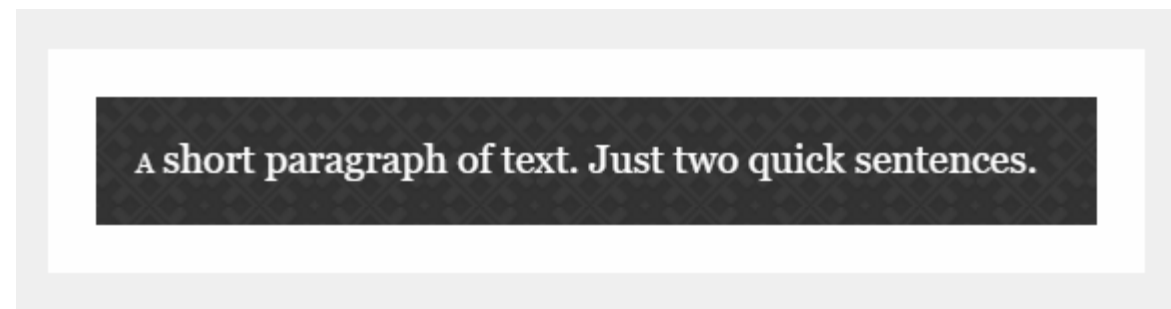
You may want to specify a `background-color` in **addition** to a `background-image` so that content is visible while the image is loading

So instead of a "blank" area...

A short paragraph of text. Just two quick sentences.

...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); }
```
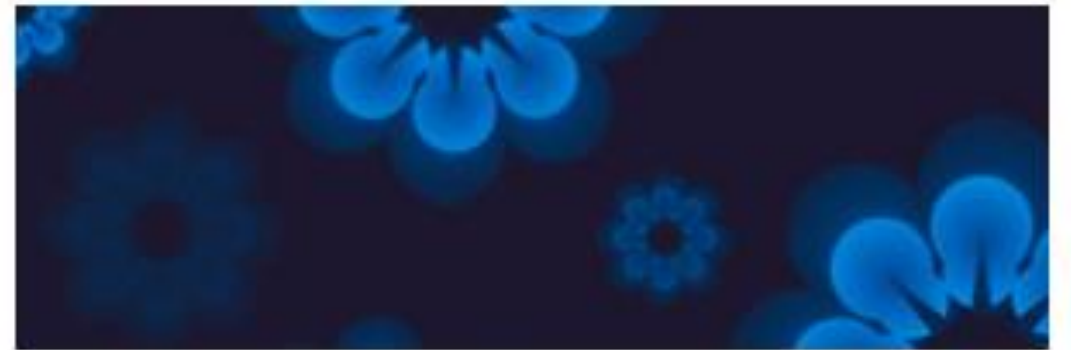


```
section {
    background: linear-gradient(to right, red, #f06d06, yellow, green);
}
```

BACKGROUND GRADIENTS

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

```css
body {

    background-image: url("flowers.png");

}
```



```css
header {

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

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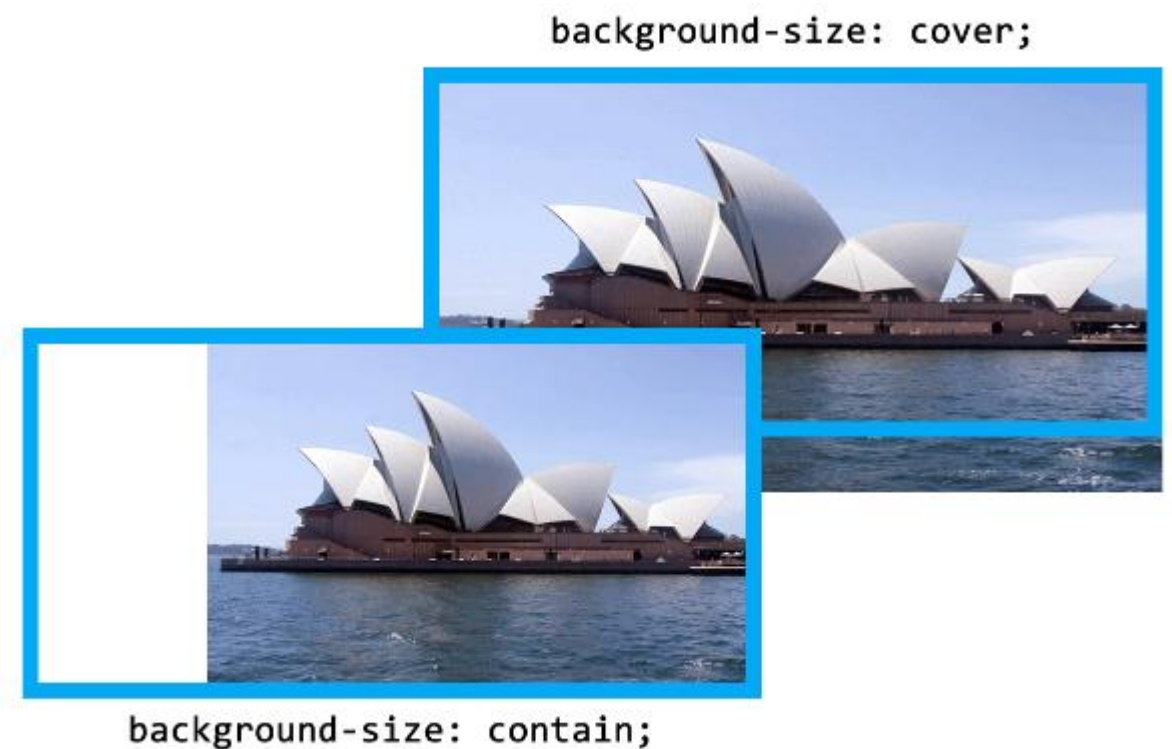
`background-size:` specifies how much of the container that the image covers

`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)

background-size: cover;

background-size: contain;

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

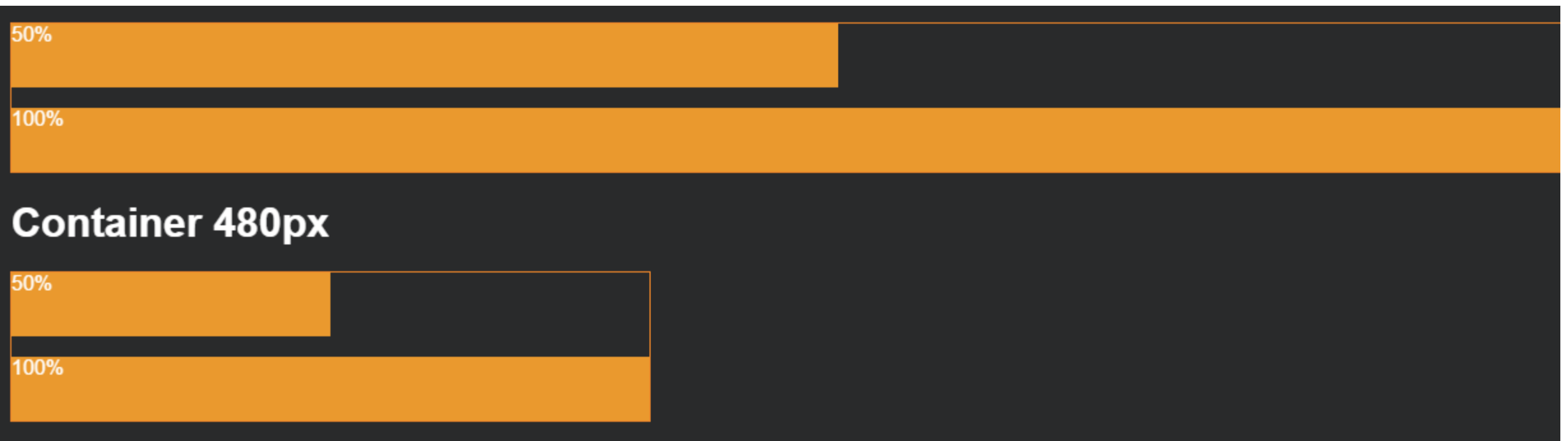```
header { height: 6em; }
```

# HEIGHT AND WIDTH %

Percentage is based on the element's **parent**'s width or height
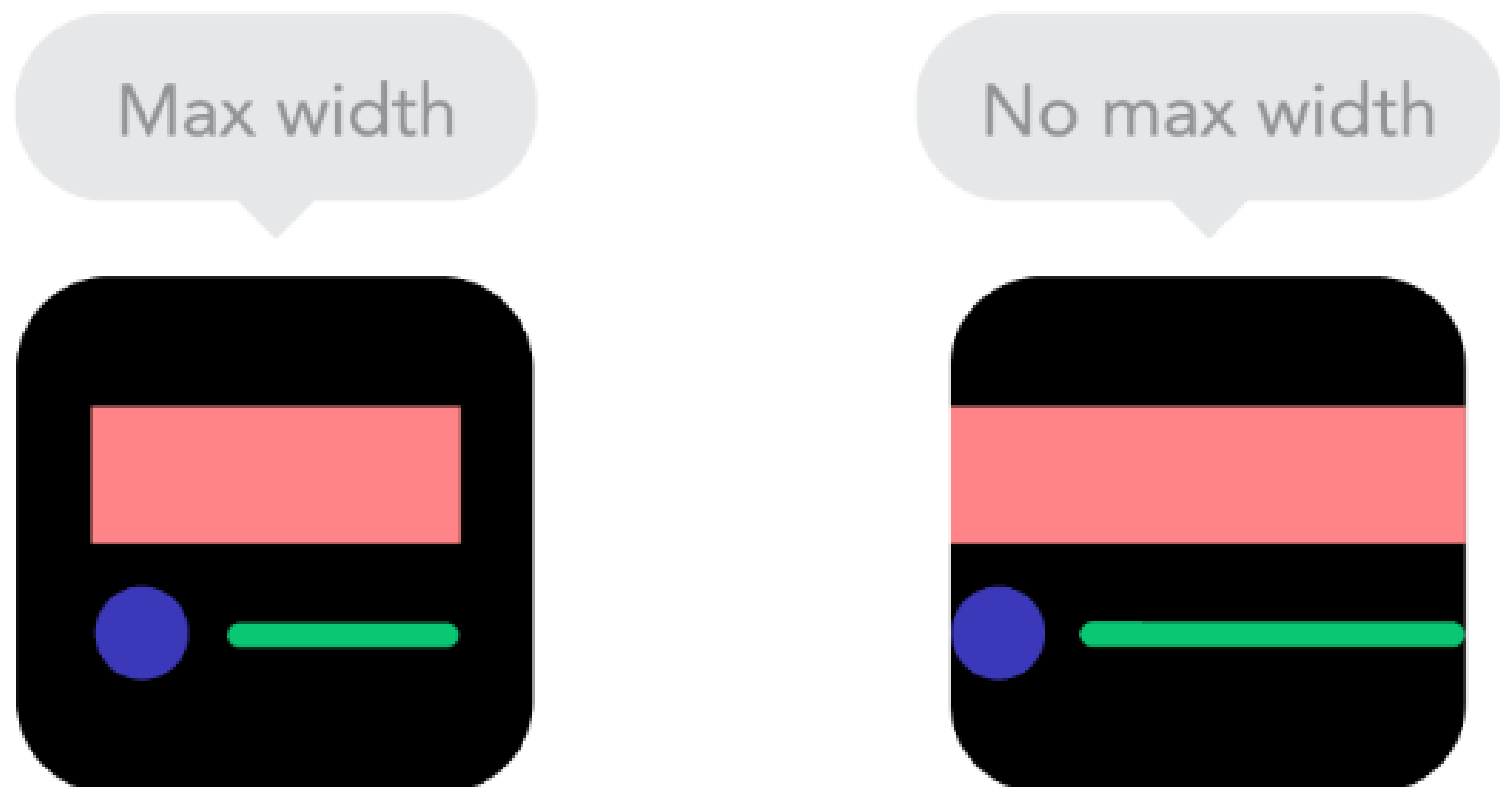
```
section { width: 50%; }
```

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

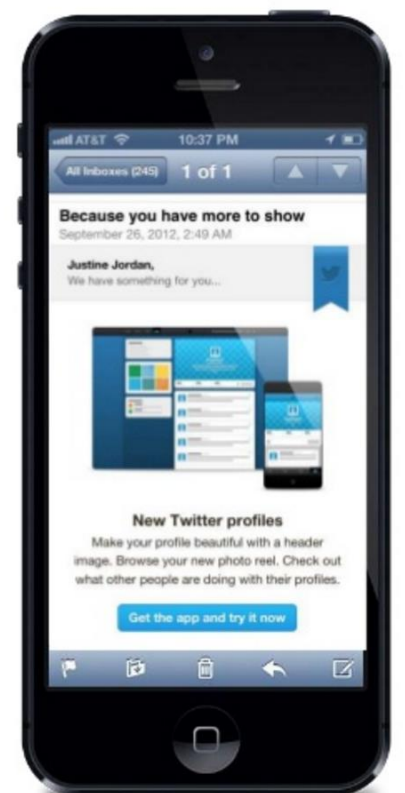# MIN-HEIGHT AND MIN-WIDTH

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
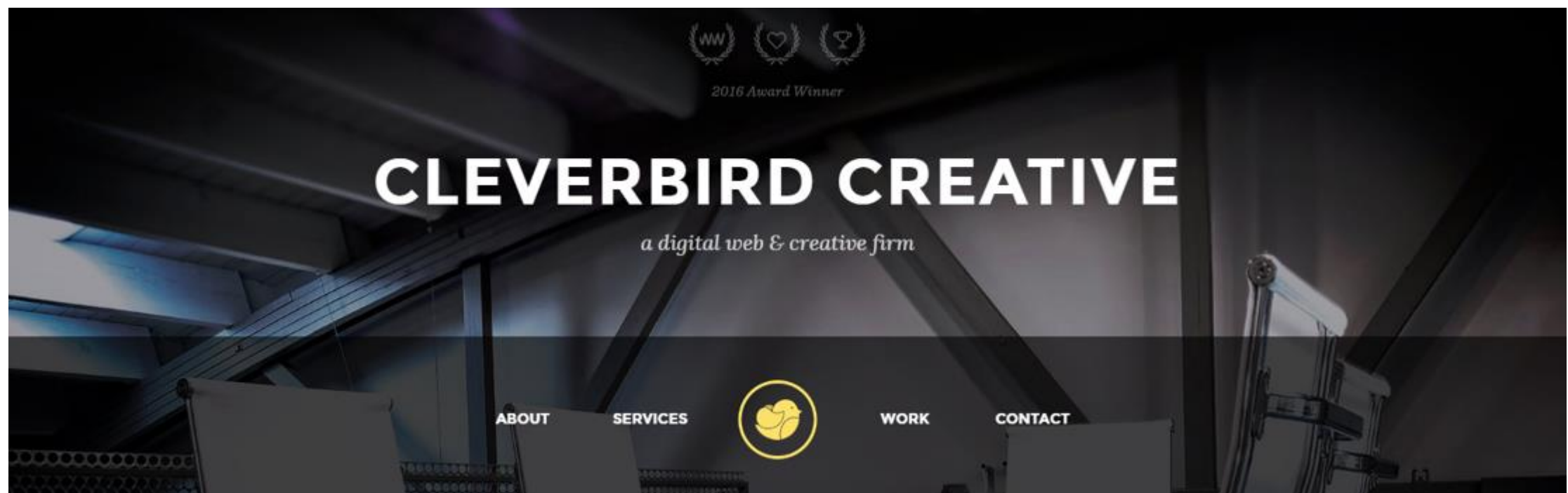
PRACTICE TIME!

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?

- 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