INTRODUCTION TO HTML & CSS

Instructor: Beck Johnson

Week 2

TODAY

- Week One review and questions
- File organization
- CSS Box Model: margin and padding
- Background images and gradients with CSS
- Make a hero banner!

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

REVIEW: 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.)

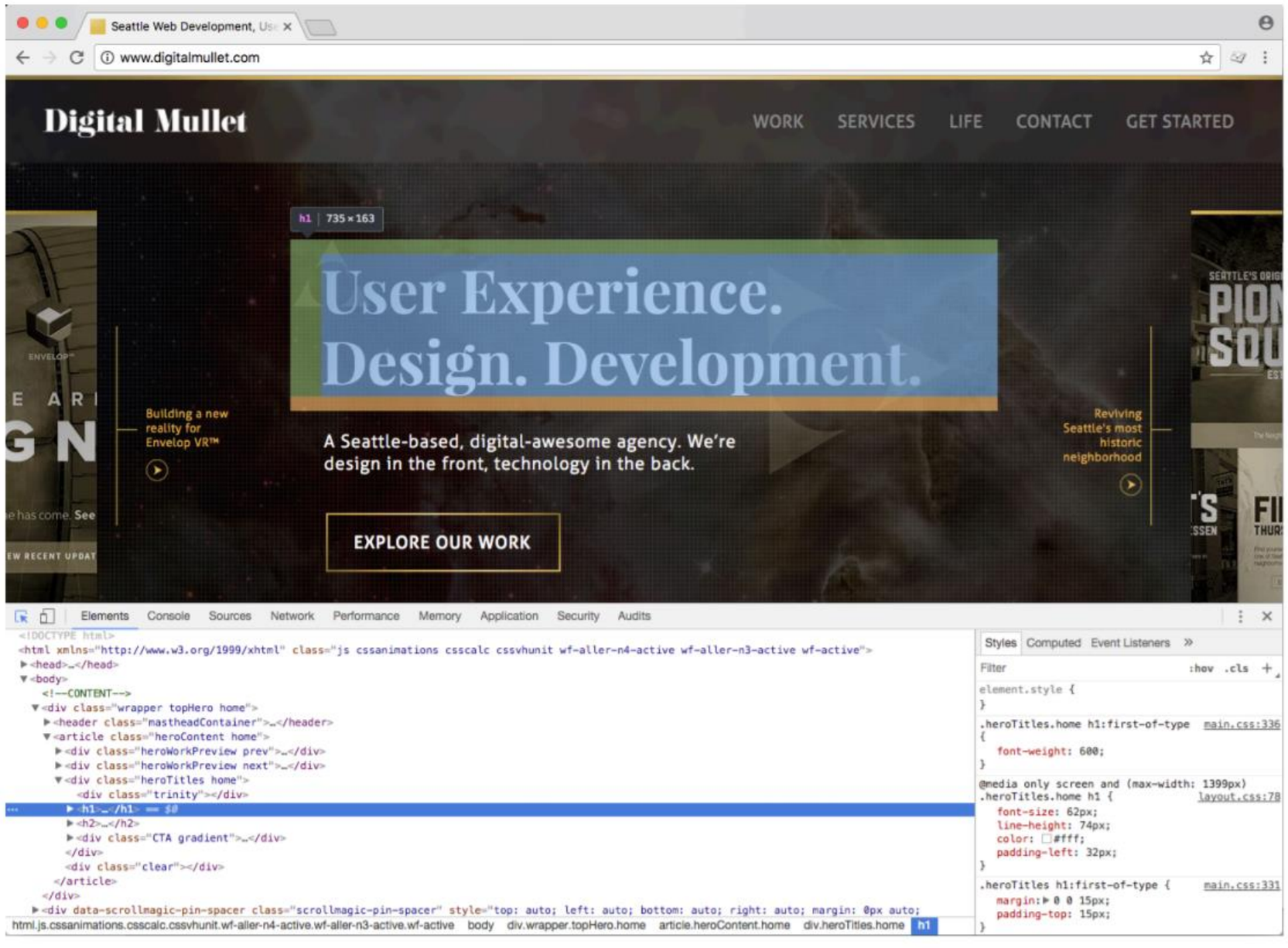
REVIEW: LINKS

```
<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



REVIEW: CSS

CSS (Cascading Style Sheet) is a different type of language than HTML, and has its own syntax

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

- CSS can be added in a separate .css file that can be linked to your HTML page

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`)

QUESTIONS?

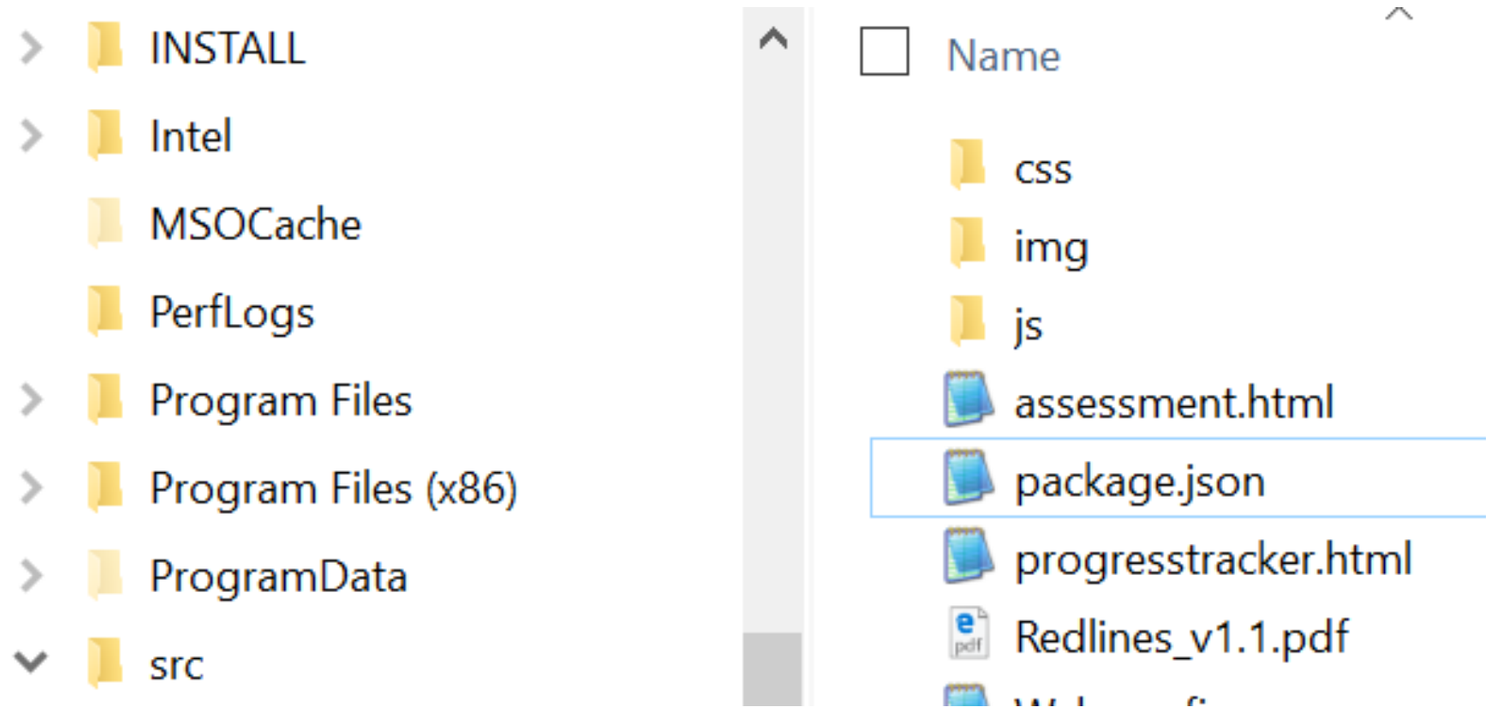
FILE ORGANIZATION

FILE ORGANIZATION

Typical files in a website include:

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

- Usually devs make **subdirectories** for media, CSS, and Javascript files



CODE ORGANIZATION

- 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 and space out your code so that it's easier for you to read

FILE NAMING RULES

- Spaces in folders or file names can cause issues
  - Most devs prefer to separate words in files using capitalization, dashes, or underscores

```
likeThis

or-this

this_is_also_used
```

URL-SCUSE ME?

A URL is a path to a file, either on your computer or a remote computer (a “server”)

When you type an URL into your browser, it’s navigating to a file stored on a server

When you use an URL in a `src` for an `<img>` tag, it finds that image by navigating to that location, starting from the folder that the HTML file is inside

```
<img src="image.gif" />
```

ABSOLUTE FILE PATHS

Absolute paths are URLs that start with **http** or **https**

```
<a href="http://google.com">Ubiquitous
search engine</a>
```

- Because these files are not hosted by you, if someone renames or deletes the file, your link will be broken

RELATIVE FILE PATHS

Relative paths are URLs that are located **relative** to your current file

- Relative paths start with / or ../ or are a filename
- Any files that are saved on your local drive should be linked using relative paths

```
<a href="other-page.html">Link to another page on my website that is in the same folder</a>
```

RELATIVE FILE PATHS

Use `../` in a path to navigate "up" a directory

```
<img src="image.gif" />                    (image.gif is in same folder)

<img src="../images/image.gif" />  (image.gif is in parent folder)
```

CSS INDENTATION

This is one popular way to indent CSS

- Starting bracket is on the same line as the `selector`
- Each `property` is on its own line, tabbed once
- Ending bracket is on its own line

```
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 Grand
31 }
```

HTML INDENTATION

This is a standard way to indent HTML

- Children elements are tabbed once
- Most elements are on a new line

```
<header>
    <div class="container">
        <div class="row">
            <div class="col-sm-8">
                <h1>HTML & CSS: Introduction to Development</h1>
                <p>
                    An introductory course about HTML & CSS principles
                    for the <a href="http://www.svcseattle.com/" target
                </p>
            </div>
            <div class="col-md-2 col-sm-3 col-md-offset-2 col-sm-offset
                <a href="https://www.amazon.com/HTML-CSS-Design-Build-W
                Recommended reading: <i><a href="https://www.amazon.com
            </div>
        </div>
    </div>
</header>


<section id="week1">
    <div class="container">
        <div class="row">
            <div class="col-lg-12 text-center">
                <h3>Week 1: Getting started</h3>
                <hr class="star">
            </div>
        </div>
        <div class="row">
            <div class="col-md-3 col-md-offset-1 col-sm-6">
```

{} CSS ORGANIZATION

Last week, we made CSS changes directly in the `<head>` element of our HTML documents

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

{} CSS ORGANIZATION

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

{} CSS ORGANIZATION

The most common way to use CSS in the real world 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 a 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, so that they apply in this order:

1. Browser defaults
2. External (from a .css file)
3. Internal (from styles in the `<head>`)
4. Inline (directly on the element)

{} 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: red;     }

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

{} HTML CHILDREN

In an HTML document, an element that is nested inside another element is referred to as a “child” of that element

- `<h2>` is a child of `<header>`
- `<a>` are children of `<nav>`

```
<html lang="en">
▶ <head>…</head>
▼ <body>
  ▼ <section>
    ▼ <header>
        <h2>Kangaroo Valley Safari</h2>
      </header>
    ▼ <p>
        "Located two hours south of Sydney in
        the Southern Highland of New South
        Wales..."
      </p>
    ▼ <nav>
        <a href="#">SHARE</a>
        <a href="#">LEARN MORE</a>
      </nav>
    </section>
  </body>
</html>
```

Both `<a>` and `<h2>` are also children of `<section>`

{} HTML CHILDREN

In CSS, to style only elements that are inside another element, use this syntax:

```
parent child { property: value; }
```

```
nav a { color: #c4fe46; }
```

“Change the color of links that are contained within a nav”

```
<html lang="en">
▶ <head>…</head>
▼ <body>
  ▼ <section>
    ▼ <header>
        <h2>Kangaroo Valley Safari</h2>
      </header>
    ▼ <p>
        "Located two hours south of Sydney in
        the Southern Highland of New South
        Wales..."
      </p>
    ▼ <nav>
        <a href="#">SHARE</a>
        <a href="#">LEARN MORE</a>
      </nav>
    </section>
  </body>
</html>
```

{} CHILDREN ARE SPECIFIC

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

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

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

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

PRACTICE TIME!

PRACTICE FILE ORGANIZATION

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

Create a new file called **styles.css**

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

Add a link to your new stylesheet on all of your webpages:

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

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

- Use indentation and whitespace

THE CSS BOX MODEL

CSS BOX MODEL

CONTENT: stuff in the box

PADDING: bubble wrap and packing peanuts

BORDER: sides of the box

MARGIN: space between multiple boxes

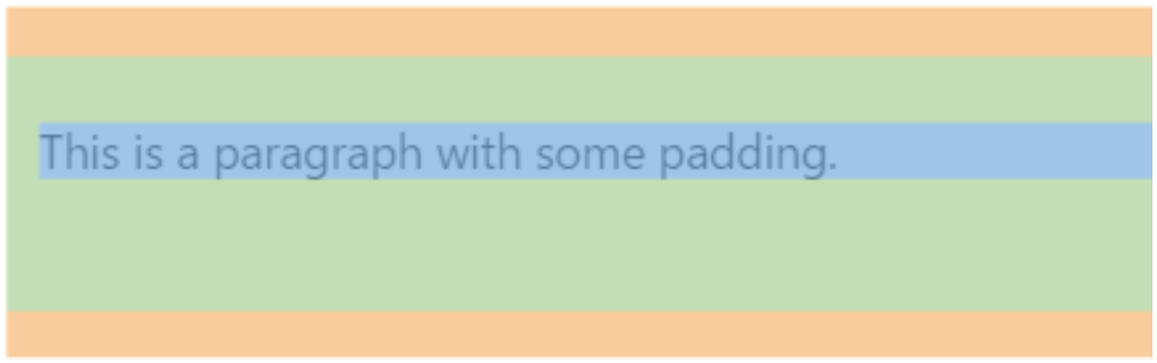
CSS BOX MODEL

PADDING

Padding creates space **inside** an element.

Padding affects how far content is from the border.

```
p {
    padding-top: 20px;
    padding-right: 5px;
    padding-bottom: 40px;
    padding-left: 10px;
}
```



Shorter way:

```
p {
    padding: 20px 5px 40px 10px;
}
```

PADDING

Padding is useful for moving content away from the edges of its container.

This is a paragraph with zero padding (default)

This is a paragraph with 1em padding

PADDING

If **top/bottom** and **left/right** padding match...

```
p {
    padding-top: 20px;
    padding-right: 10px;
    padding-bottom: 20px;
    padding-left: 10px;
}
```

Combine them!

```
p { padding: 20px 10px; }
```

PADDING

If **all** padding matches...

```
p {
    padding-top: 20px;
    padding-right: 20px;
    padding-bottom: 20px;
    padding-left: 20px;
}
```

Combine EVEN MORE!

```
p { padding: 20px; }
```

PADDING

Padding can be applied only to the top, only to the bottom, and so on – or any combination of those:

```
p {
    padding-left: 40px;
}

p {
    padding-top: 20px;
    padding-right: 10px;
}
```

MARGIN

Margin moves elements away from one another.

```
img {
    margin: 6px;
}
```

MARGIN

Margin creates space **outside** an element.

- Same abbreviation style and rules as padding

```
p {
    margin-top: 20px;
    margin-right: 5px;
    margin-bottom: 40px;
    margin-left: 10px;
}
```

Is the same as

```
p { margin: 20px 5px 40px 10px; }
```

MARGIN

You can give `margin` a **negative** value to shift elements in the opposite direction.

```
p {
    margin-top: -20px;
}
```

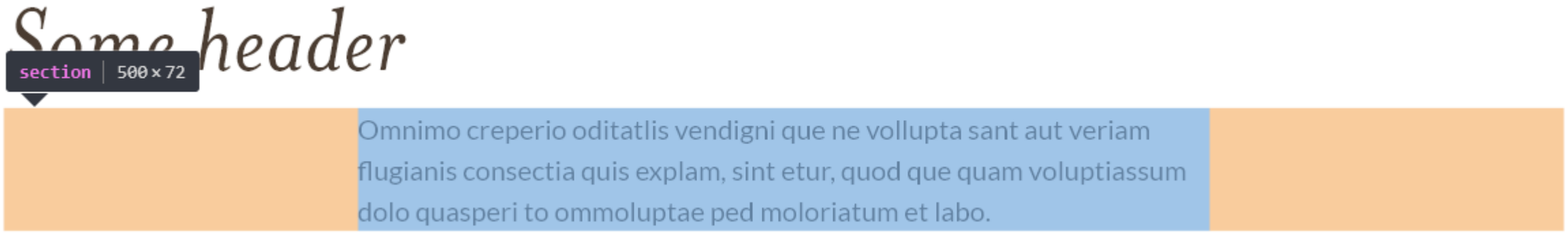
This may result in overlapping text!

MARGIN AUTO

To automatically center elements, you can use the property `auto`, which evenly applies a margin on both sides

- When using `auto`, a `width` must be applied to the element, so that the browser knows how much margin to automatically apply

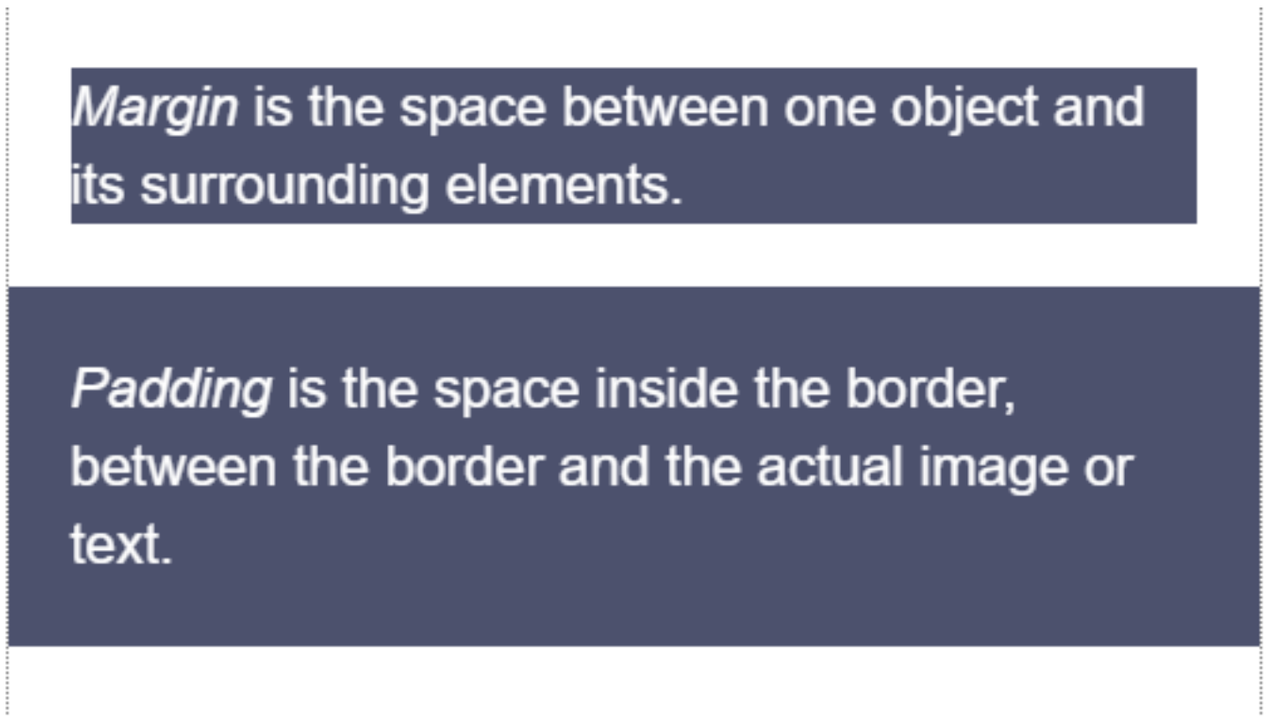
```
section {
    margin: 0 auto;
    width: 500px;
}
```



MARGIN VS. PADDING

Use `margin` to separate the element from the things that are around it.

Use `padding` to move the element away from the edges of the block.



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:

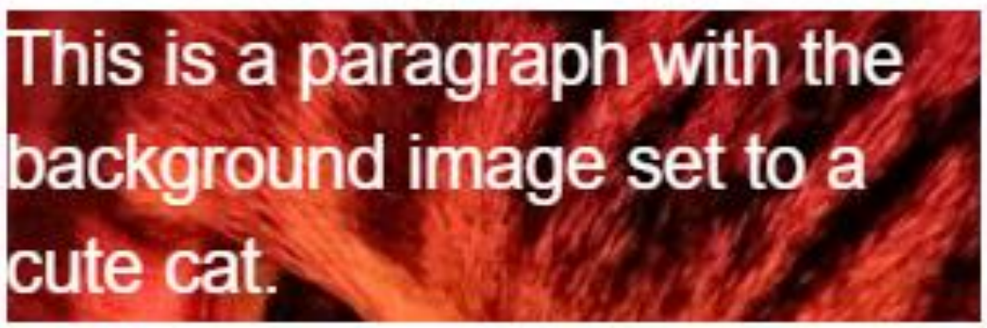
```
p {
    background-image: url("images/kitten.jpg");
    color: white;
}
```

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

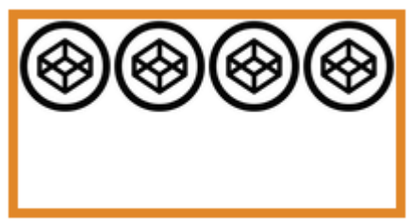
- By default, background images are repeated until they fill the entire container

```
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

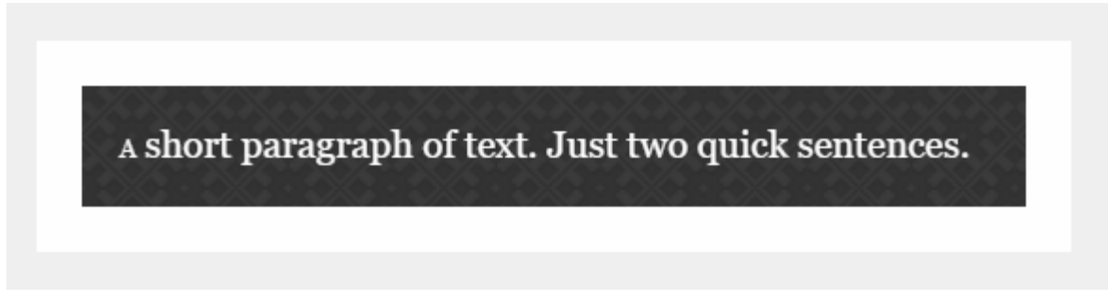
`background-attachment:` images usually scroll with the main view, but setting to `fixed` means the image stays in place when the user scrolls the page

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

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

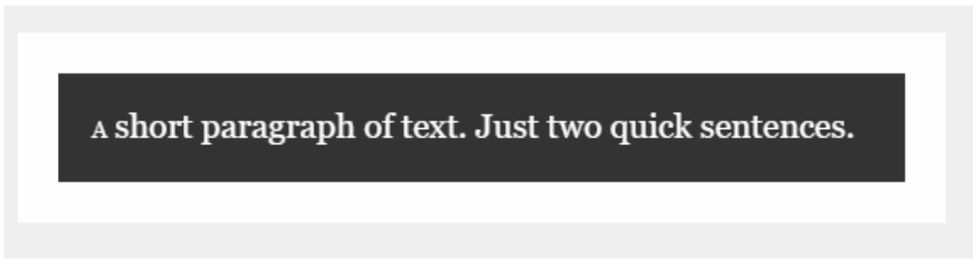
FALLBACK BACKGROUND COLOR

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…

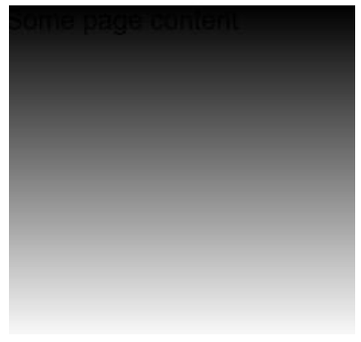


…the user can see content while the image downloads

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); }
```



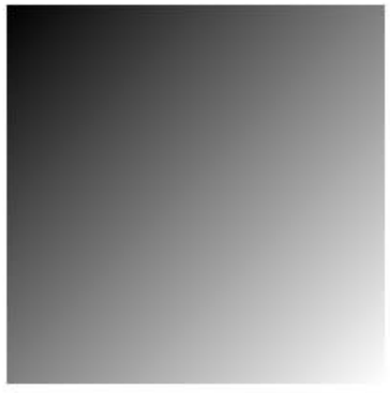
As many colors as you want can be blended, separated by commas:

```
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); }
```

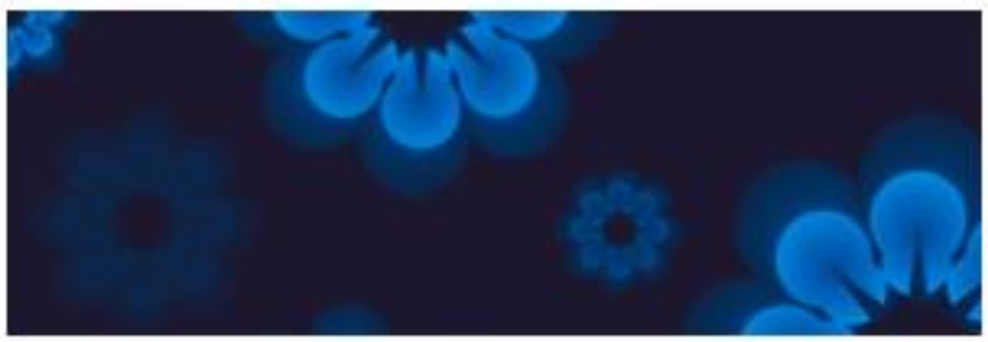


```
section {
        background: linear-gradient(to right, red, #f06d06, yellow, green);
}
```

BACKGROUND GRADIENTS

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

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



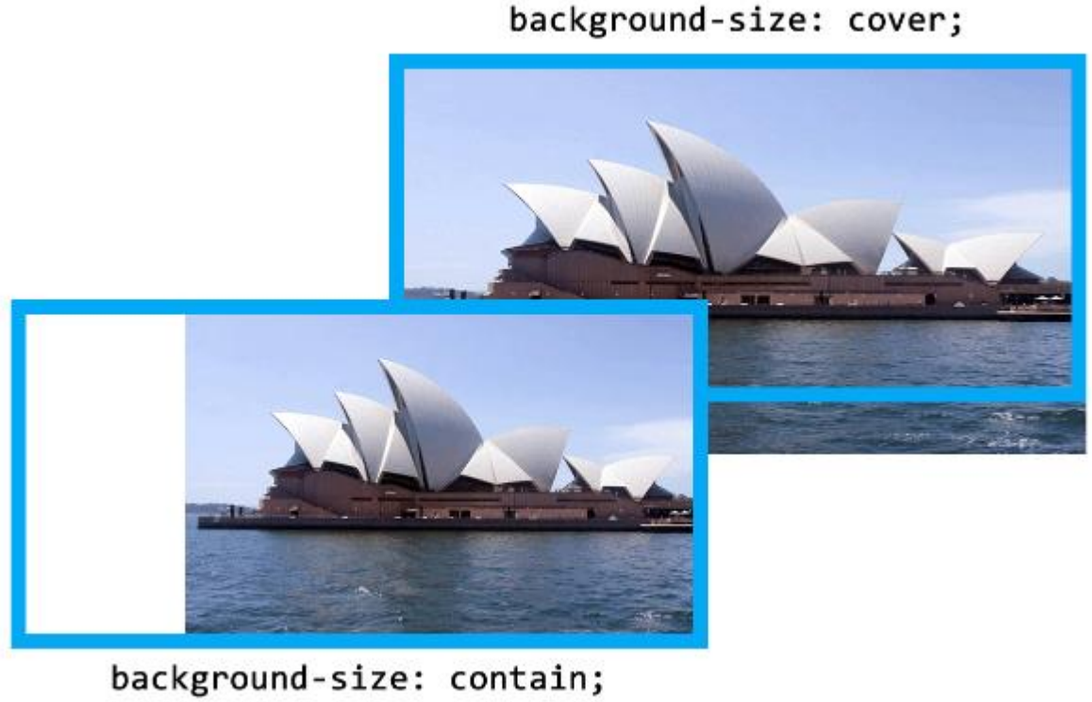
```
header {
        background-image: linear-gradient(to
right, rgba(255,255,255,0),
rgba(255,255,255,1));
}
```

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 making the image bigger)

`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

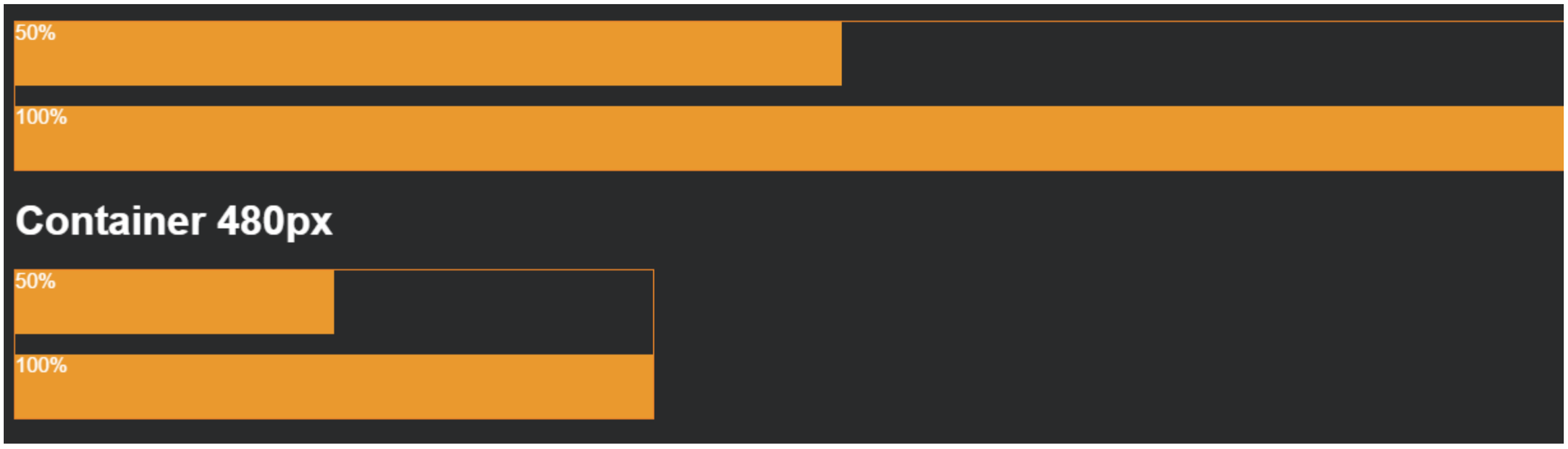
```
header { height: 6em; }
```

HEIGHT AND WIDTH %

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

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

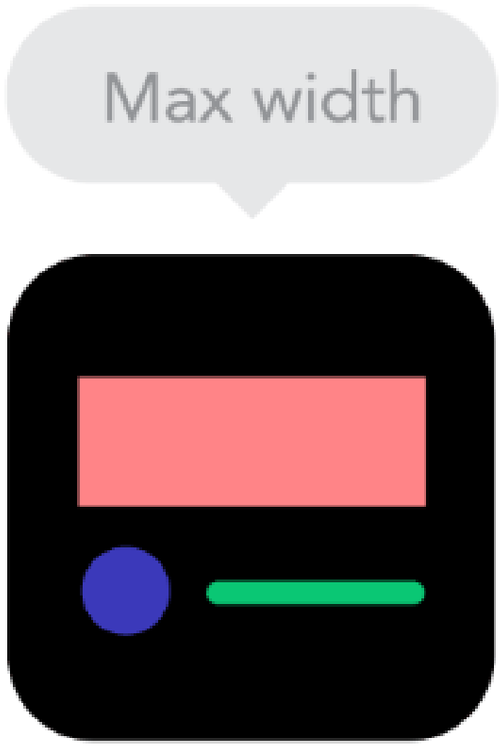
If that `section` were inside a 480 px wide container, it would end up being 240 px wide.

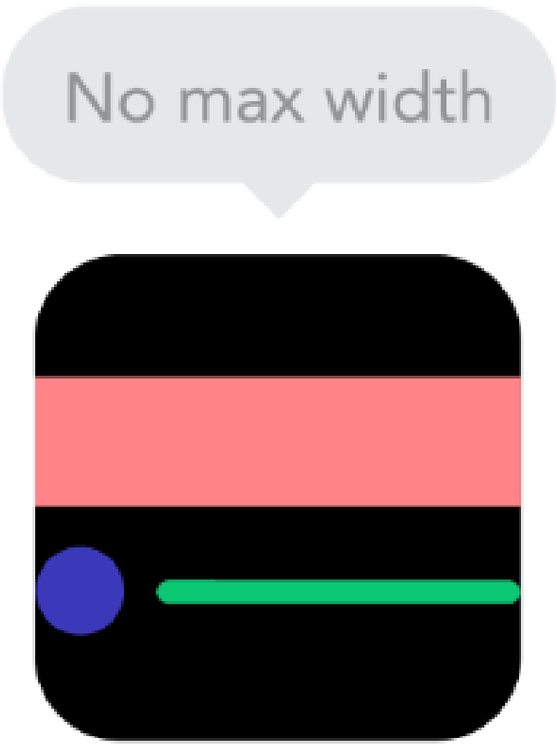


MAX-HEIGHT AND MAX-WIDTH

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

A better commercial real estate experience

Colliers International Group Inc. (NASDAQ: CIGI) (TSX: CIGI) is a top tier global real estate services and investment management company with more than 14,000 skilled professionals operating in 68 countries. What sets us apart is not what we do, but how we do it. Our culture encourages people to think differently, share great ideas and create solutions that help our clients accelerate their success.

PRACTICE TIME!

MAKE A HERO

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?

HOMEWORK

Before next week, edit **about.html**

- Link to an external CSS file (consider using the styles.css file you created in class today)
- Make a style “override” that is specific to this page
  - Either inline as an or in the page head

Email your HTML and CSS files to
beckjohnson@gmail.com

“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