INTRODUCTION TO HTML & CSS

Instructor: Beck Johnson
Week 3

- Review background image and external CSS files

- The CSS box model - borders

- Block vs inline vs flex elements

- Classes and IDs

- Coding from a design "comp"

REVIEW!

REVIEW: 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
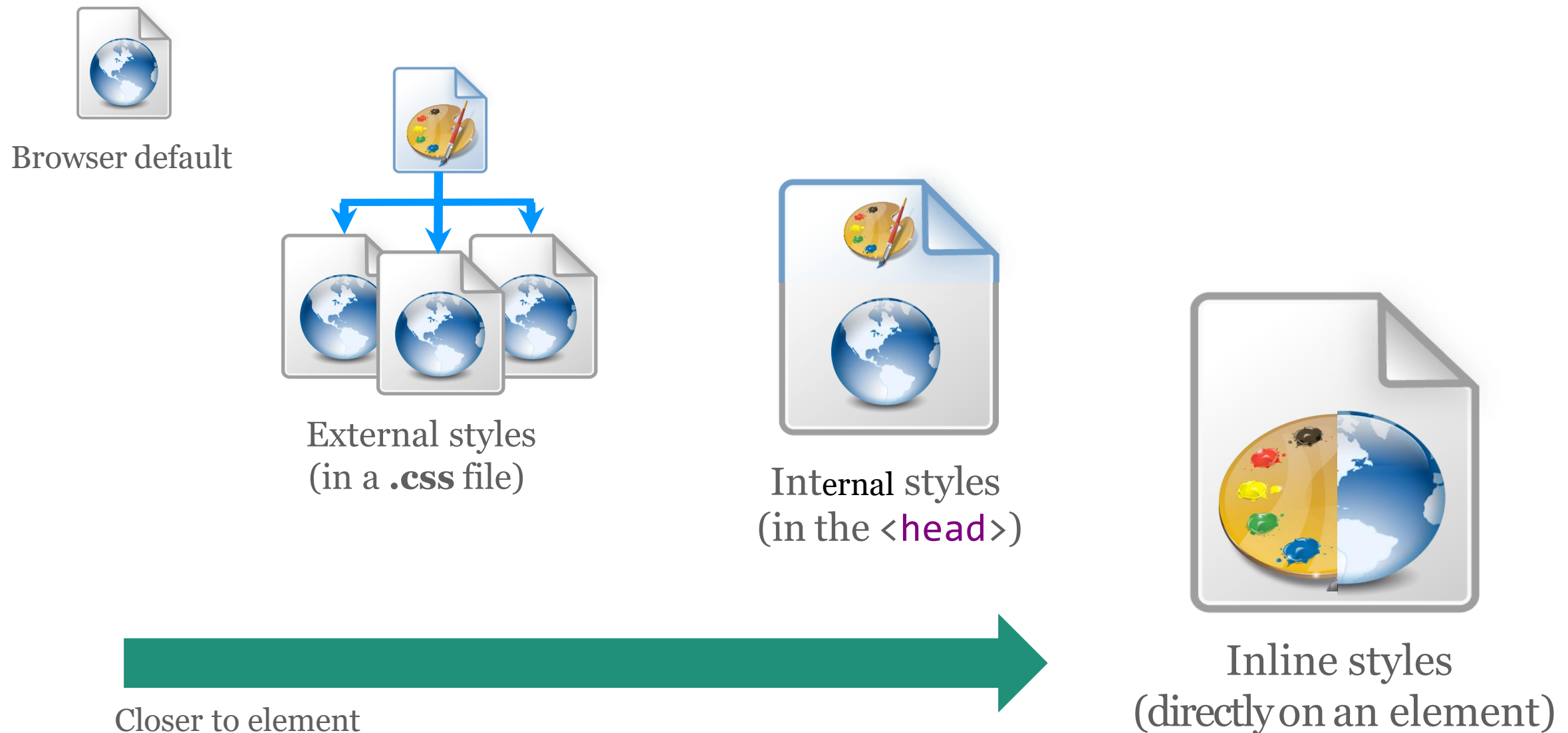
- Should be on every page that needs the styles

The 3 rules for determining how styles get applied:

- Styles are applied from **far** to **near**

- Styles are applied from **top** to **bottom**

- Styles are applied from **parent** to **child**

REVIEW: NEAR TO FAR

Styles that are "closer" to the elements they style take precedence

Browser default

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

Internal styles
(in the `<head>`)

Inline styles
(directly on an element)

Closer to element

REVIEW: TOP TO BOTTOM

If the same `property` is styled multiple times for the same `selector`, the last one sticks.

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

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

REVIEW: CHILD TO PARENT

If the child is styled differently from the parent, the child's style is used instead

```
li { color: #daa645; } /* all list items */

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

li a { color: #c4fe46; } /* links in lists */
```

The background of an element can be an **image** (instead of a color) using the property `background-image`

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

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


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

{} REVIEW: BACKGROUND IMAGES

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

`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

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

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

`background-image` can also be a `linear-gradient`

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

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 right, red, #f06d06, yellow, green);

}
```

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

- The `value` must be a positive number

- Units are either px or em or %

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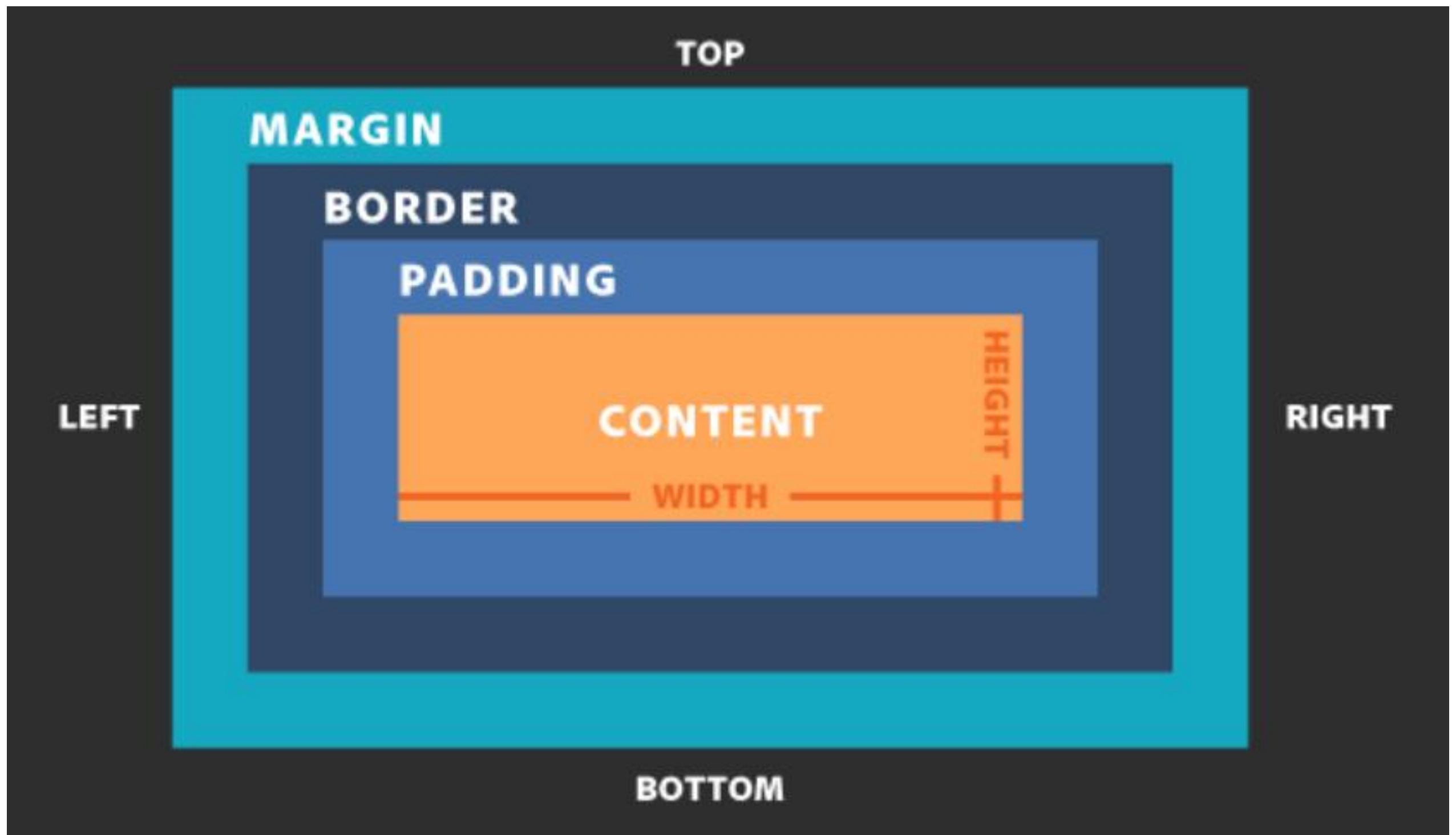```

`min-height` and `min-width` specify minimum dimensions

`max-height` and `max-width` specify maximum dimensions

# QUESTIONS?

# THE CSS BOX MODEL

# CSS BOX MODEL

# 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.

Margin is the space between one object and its surrounding elements.

Padding is the space inside the border, between the border and the actual image or text.

# BORDER STYLES

Between margin and padding, you can set a border

Values are separated with spaces, in this order:

- Width (usually in pixels, but can be em)

- Border style (solid, dotted, dashed, etc)

- Color

```
p {
    border: 2px dotted #ff0000;
}
```

Border styles:

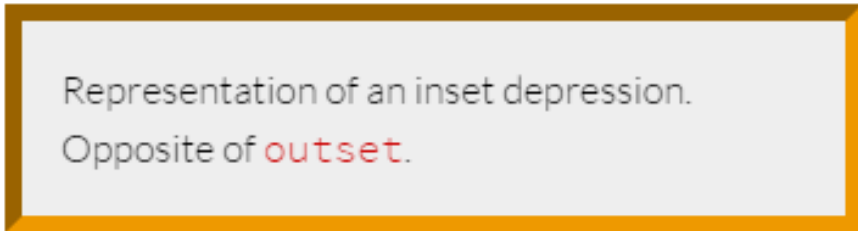|	
solid	Solid line.
dotted	Series of dots.
dashed	Series of dashes.
double	Two solid lines.
groove	Representation of a carved groove. Opposite of ridge.
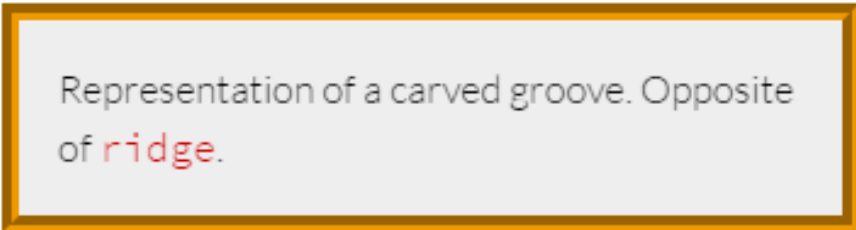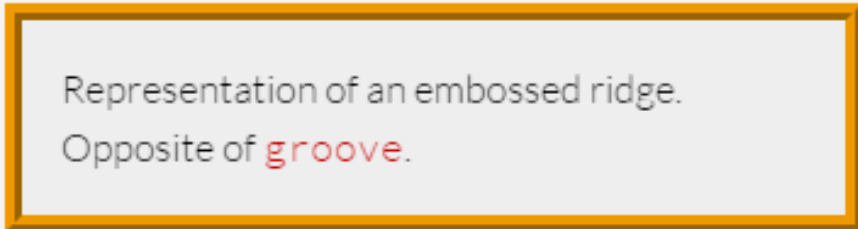
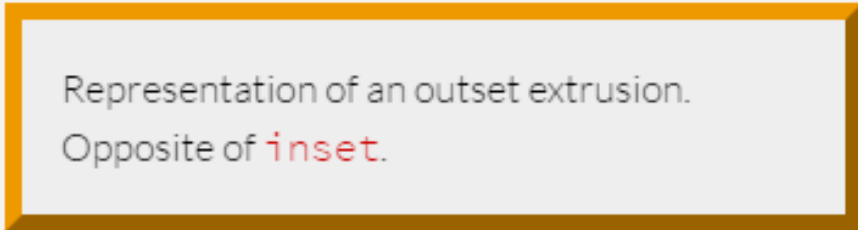|	
ridge	Representation of an embossed ridge. Opposite of groove.
inset	Representation of an inset depression. Opposite of outset.
outset	Representation of an outset extrusion. Opposite of inset.

BORDER STYLES

You can set a border on only one side of an element:

```css
h1 { border-bottom: 3px solid black; }
```

HEADER WITH BORDER BOTTOM

BORDER STYLES

A common use of border is to visually separate list items in a navigation menu.


```css
ul {
    list-style : none;
}

li {
    padding: 1em;
    background-color: #283c51;
    border-top: 1px solid #395673;
    color: #adadad;
}
```
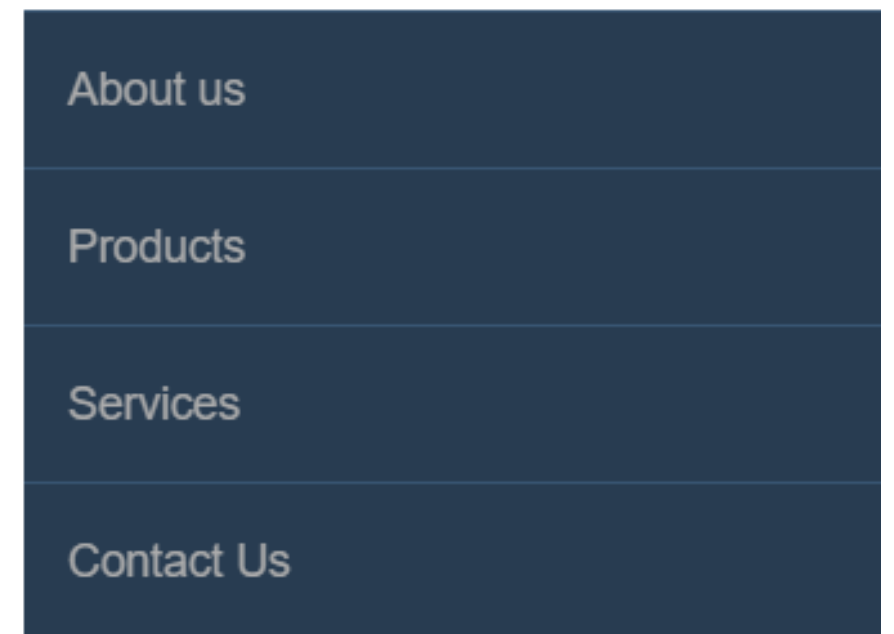
```html
<ul>
    <li>About us</li>
    <li>Products</li>
    <li>Services</li>
    <li>Contact Us</li>
</ul>
```

LIST STYLE

Note that we set

```css
ul {
    list-style: none;
}
```

to remove the bullets that appear
by default on an unordered list

BORDER RADIUS

To make an element appear curved, use the property
`border-radius`

- The value is a number (in px or em) or percentage

- You can use `border-radius` even if you don't explicitly set a `border`

```
li {
    /* same styles... */
    border-radius: 2em;
}
```

About us

Products

Services

Contact Us

BORDER RADIUS

border-radius can be used to create a circle.

- Set border-radius to 50%

- Set height and width to the same value

```
li {
    border-radius: 50%;
    background-color: black;
    color: white;
    text-align: center;
    height: 3em;
    width: 3em;
    line-height: 3em;
    margin: 5px;
}
```

BORDER RADIUS

This technique can be used on images to crop them into a circle

- Note: if the image itself doesn't have a square ratio, it will look distorted

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

img {
    border-radius: 50%;
    height: 200px;
    width: 200px;
}
```

STATES IN CSS

CSS allows you to apply styles based on the **state** of an element, for example:

- Being **hovered** over with a mouse
- Gaining **focus** via tabbing or clicking

This is known as a CSS pseudo-class
- It's "pseudo" because the element doesn't exist in markup – what it selects may change based on user interaction or position relative to other elements

Whenever you see a : in a selector, that style will only apply to elements that are in that state or that position

We saw this already for hovering:

```css
p:hover {
    background-color: #999;
}
```

This paragraph gets fancy when you hover over it

This paragraph gets fancy when you hover over it

`:first-letter` is a way to style the first letter of an element without needing to add markup

```
p:first-letter {
    font-size: 20px;
    float: left;
}
```

P ellentesque habitant morbi tristi egestas. Vestibulum tortor quar Donec eu libero sit amet quam egest placerat eleifend leo. Pellentesque hi malesuada fames ac turpis egestas. tempor sit amet, ante. Donec eu liber

`:first-child` selects only the first child

`:last-child` selects only the last child

From the previous example, you would probably only want the first letter of the *first* paragraph to be big

```css
p:first-child:first-letter {
    font-size: 20px;
    float: left;
}
```

`:first-child` is also commonly used in lists to style the first element differently from the others

- For example, separate list items with borders, but don't show a top border on the first item:

```css
li {
    border-top: 1px solid #333;
}

li:first-child {
    border-top: none;
}
```

PRACTICE TIME!

- Add a **list** of links in your navigation menu if you don't already have one

- Make the navigation menu pretty by using padding, margin, border, border-radius, background color, and other tricks we've learned.

  - ONLY style lists that are in the nav menu – not any lists that may appear on the rest of the page

- Try using a pseudo-selector somewhere on your page to identify an element either by state (`:focus`, `:hover`) or by position (`:first-child`, `:first-letter`)

BLOCK VS. INLINE ELEMENTS

BLOCK ELEMENTS

- Expand naturally to fill their parent container
  - Takes up a "full line"

- Can have margin and/or padding

- Can have height and/or width

- By default, will be placed **below** previous elements in the markup

BLOCK ELEMENTS EXPAND NATURALLY ———————→

AND NATURALLY DROP BELOW OTHER ELEMENTS

Examples of block elements:

- Headings `<h1>...<h6>`
- Paragraphs `<p>`
- Lists `<ul>`, `<ol>`

INLINE ELEMENTS

- Flow along with text content

- Only take up as much space as necessary

- Ignore width and height properties

- Margin and padding only pushes other elements away horizontally, not vertically

- Top and bottom margin/padding is ignored

INLINE ELEMENTS FLOW WITH TEXT

PELLENTESQUE HABITANT MORBI TRISTIQUE SENECTUS ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS. VESTIBULUM INLINE ELEMENT VITAE, ULTRICIES EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

Examples of inline elements:

- Links `<a>`

- Font emphasis `<em>`

- Font bold `<strong>`

Pellentesque *inline element* morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

INLINE-BLOCK ELEMENT

- Is a hybrid of inline and block

- Takes up width and height like block-level elements

- Flows with content

- Can have margin and padding

- Examples of inline-block elements:
  - Image `<img />`

Pellentesque inline block inline block inline block morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

You can change whether or not any element is block, inline, or inline-block by using the CSS display property.

- This means we can do some neat things!

```
li {
    display: inline-block;
}
```
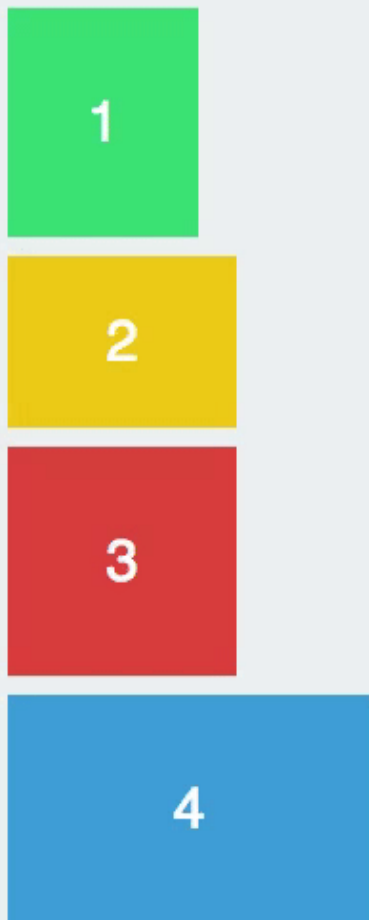
FLEX

- Invented in 2017, this display type allows for better layout control
  - There are no innate HTML elements that have `display: flex`

- Unlike `block`, `inline`, or `inline-block`, it doesn't affect itself, it affects the layout of its children

By default, `display: flex` will try to place **all its children** on the same row
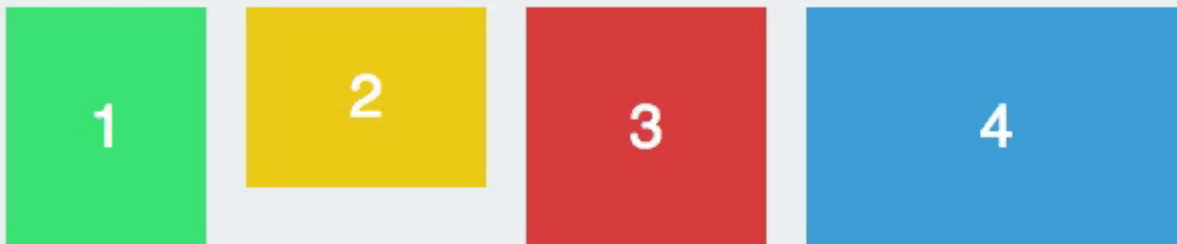
- That may mean that they shrink to fit!

display: block;

We'll talk more about `display: flex` next week when we go over layout

align-items: flex-start;

```css
a {
    color: #0099CC;
    display: inline-block;
    border: 2px solid #0099CC;
    border-radius: 6px;
    padding: 16px 32px;
    text-decoration: none;
    text-transform: uppercase;
    transition: all .4s;
}

a:hover, a:focus {
    color: #fff;
    background-color: #008CBA;
}
```

```html
<a href="index.html">
        Blue
</a>
```

BLUE

PRACTICE TIME!

ASSIGNMENT

Turn your navigation into a horizontal menu using CSS

- Give the `li` elements a `display` property of either `inline` or `inline-block` (or you can try giving `ul` a `display` property of `flex`)

- Update your styles so that they look nice in the new orientation

Note: if you like your menu how it is, experiment with changing `display` of another element on the page instead

Create an `<a>` link that looks like a button.

- Style the button differently on hover and focus and/or click

<html>

(MORE) HTML ELEMENTS

`<span></span>`

A `<span>` is a **generic inline element**

- No default style

- Used to style inline content

`<div></div>`

A `<div>` is a **generic block element**

- No default style

- Heavily used as a wrapper for other elements, to create complex layouts

WHY USE DIV OR SPAN?

Both `div` and `span` really need something extra to be useful, since they have no presentation style by default.

- Used mostly to create **layout**

- Have no semantic meaning

- You don't need to "reset" them before making them fit your design (like `ul` or `p`)

- What do they need?

ID & CLASS SELECTORS

CSS lets us target **all** paragraphs like this:

```css
p {
    font-size: 20px;
}
```

But what if we want to style only **some** paragraphs?

CLASSES AND IDS

You can add class and id attributes to any HTML element to identify it for styling.

- You decide the class and id names – be descriptive!

```
<p class="important">Big text</p>

<p class="anyLettersOrNumb3rsOr_Or-">Still totally valid</p>
```

Adding a `class` or `id` does nothing to an element by default.

- Classes and ids don't have any styling information by themselves
- They require you to add CSS if you want styling to be applied

Multiple elements can have the same `class`

- A class is like a barcode – all of the same products have the same barcode

Only **one** element per page can use the same `id`

- An id is like a serial number – it uniquely identifies one specific instance of a product

- In CSS, target a `class` with a **period**

- Will style **all** types of elements that have that `class`:

```
.ghost { color: white; opacity: .1; }

<p class="ghost">Spooooky!</p>

<div class="ghost">This will be spooky toooo</div>
```

ID ATTRIBUTES

- An **id** can only be used **once** per page

- Elements **cannot** have multiple **id** attributes

```
<div id="mainContent">
    <!-- This better be the only one! -->
</div>
```

ID SELECTORS IN CSS

```html
<div id="lego"></div>
```

In CSS, target an id with a **hash**:

```css
#lego {
    display: block;
}
```

If you put a hash followed by the element's id in the URL, the browser will **jump** to that location on the same page:

```
<a href="#kittens">Proceed directly
to kittens</a>

...

<div id="kittens">Meow<div>
```

ID ATTRIBUTES

Q: What horrible thing will happen if you use an `id` twice on the same page?

A: Well...actually nothing.

- But your page won't validate

- Jump links will go to whatever `id` appears first

- And any Javascript that needs to locate that specific element will fail

HOW TO CHOOSE - CLASS OR ID?

If you think it's likely or possible that you'll want to apply the same style to multiple things, definitely use `class`

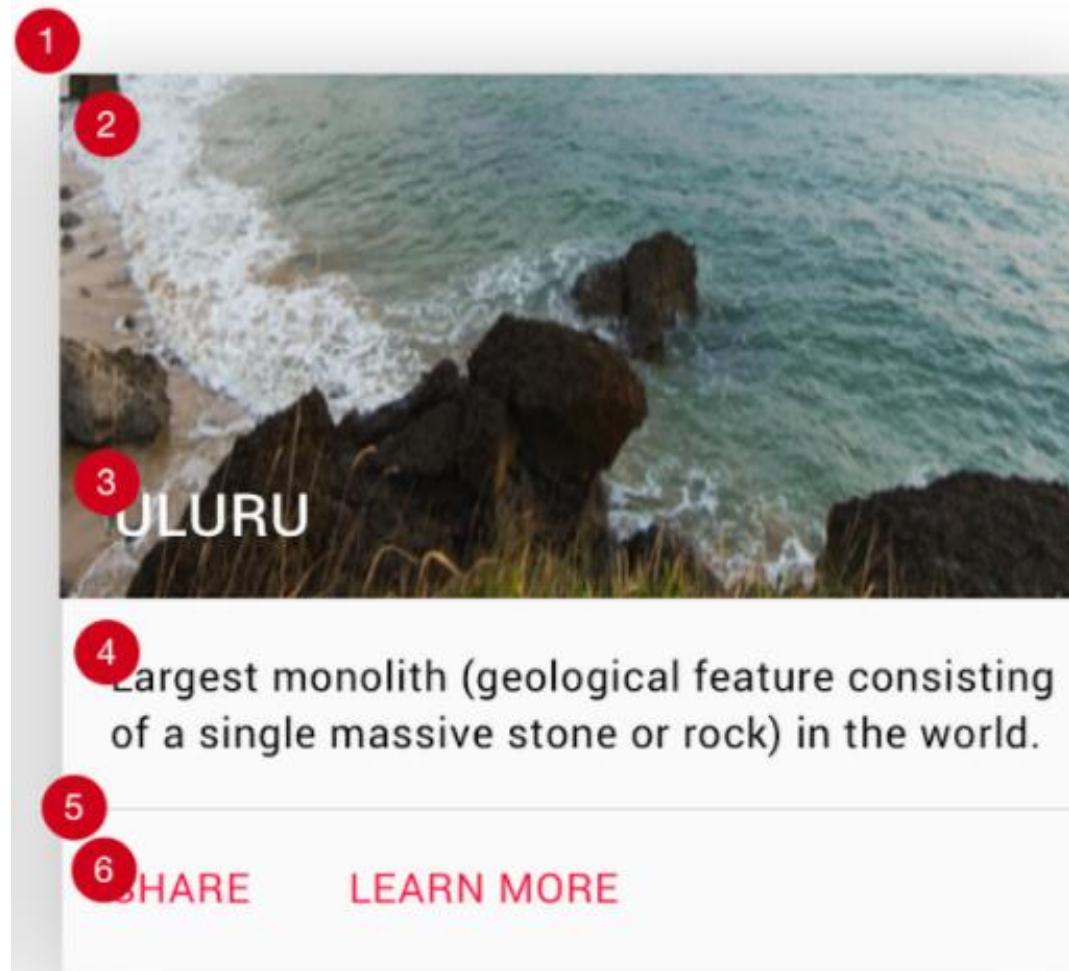If your element is guaranteed to be the only one on the page, you can use `id` – or you can still use `class`

If your element needs to be linked to directly, use `id`

PRACTICE TIME!

{ } MATCH THE COMP

Using all the techniques you've learned, try to match this comp:

1 height: 304 px
width: 345 px
background: #FAFAFA
shadow: 0 px 7 px 35 px 0 px
shadow color: rgba (0, 0, 0, 0.3)
radius: 2px
margin: 14 px

2 height: 176 px
width: 345 px

3 font: roboto
weight: medium
size: 20 px
color: #FFFFFF
text-transform: uppercase
padding: 140 px 16 px 0 px 16 p

4 font: roboto
weight: regular
size: 14 px
color: #000000
padding: 16 px

5 border: 1 px solid #E0E0E0

6 font: roboto
weight: regular
size: 14 px
color: #FF1744
text-transform: uppercase
padding: 0 px 16 px 0 px 16 px

1 2 3 ULURU

4 Largest monolith (geological feature consisting of a single massive stone or rock) in the world.

5 6 SHARE   LEARN MORE

{ } MATCH THE COMP: TIPS

Create a new page. You can put CSS in the `<head>` or create a new stylesheet, your choice

For the tiles: `<div class="tile"></div>`

- Use this URL to generate a random nature photo for your background image:
  https://placeimg.com/344/204/nature

- Use a `<nav>` for the row that contains links
  - Either add a `class` to this `<nav>` or identify it using `.tile nav` in your CSS

- To get the drop-shadow effect, apply this CSS to `.tile`
  `box-shadow: 0 7px 35px 0 rgba(0, 0, 0, 0.3);`

HOMEWORK

On index.html or aboutme.html, give an element a descriptive `class` and apply a special style to it using a CSS class selector

- Style a **child** element of this element

Assign an `id` to an element on your page and apply a unique style using that `id` selector

- Optional: create a link that jumps to that element

Email me your files at beckjohnson@gmail.com

"HOMEWORK"

- Practice!

- Optional: read chapter 8 of
*HTML and CSS: Design and Build
Websites*

- Try playing with this [interactive
demo](#) of the CSS box model