CSS POSITIONING

WEEKLY OVERVIEW

WEEK 4	Responsive Design / CSS Positioning
WEEK 5	Forms / Final Project Lab
WEEK 6	JS Intro

AGENDA

- Display
- CSS Positioning
- CSS Transitions
- CSS Transformations
- Image Overlay Lab

LAB

LEARNING OBJECTIVES

- Differentiate between inline, block, and inline-block elements
- Use the `position` property to position elements on the page
- Utilize transitions to add basic animations on hover

FEWD

GRID SYSTEMS REVIEW

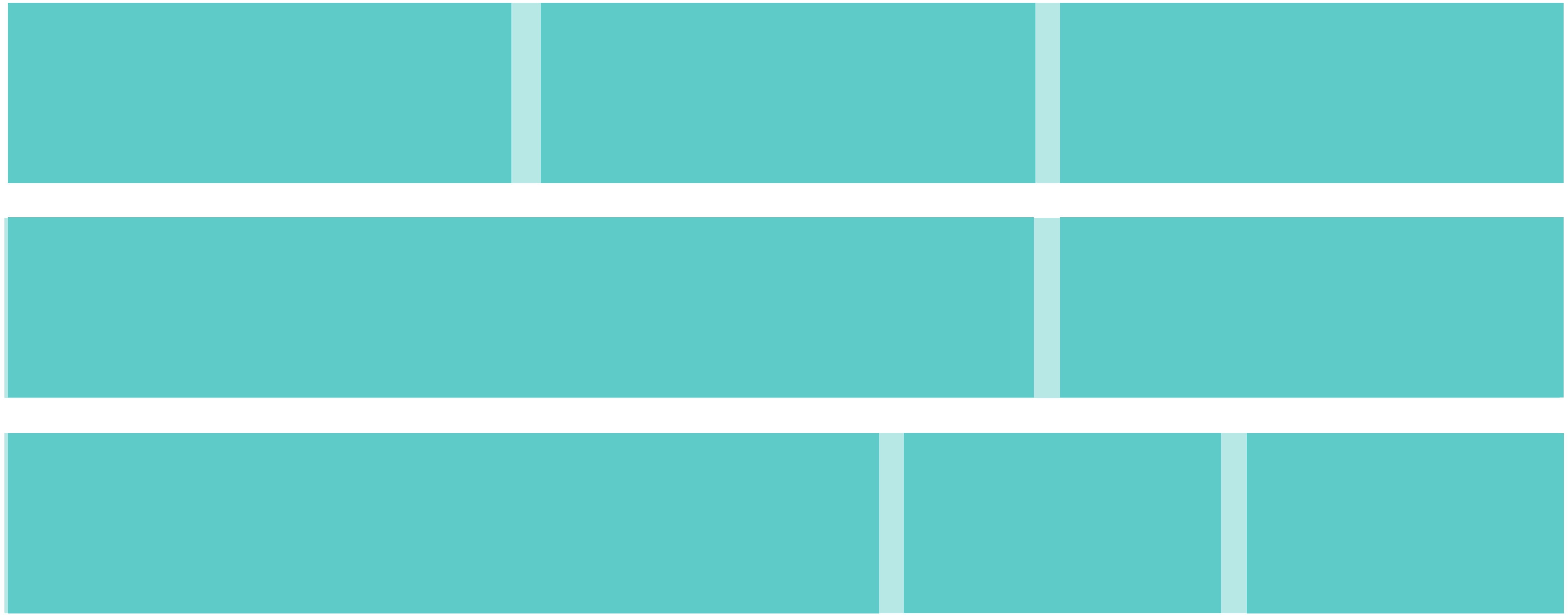
TURN AND CHAT

- Part 1
  - What does the “pink” class do?
  - Why do we have classes like “pink”, “green”, “blue”, etc?
- Part 2
  - What does the “col-1-2” class do?
- Part 3
  - What does the “col-md-1-2” class do? How is it different from the “col-1-2” class?
  - When would we want to use the “col-md-1-2” class?
  - Why does the media query for max-width: 970px come before the media query for max-width: 750px?

https://codepen.io/larissam/pen/aYjEGK?editors=1100#0

ROWS

COLUMNS



ROWS

Since all our rows will be set to **`display: flex;`** we can store this property in a class and apply it to any element to turn it into a row.

```
.row {
  display: flex;
  flex-wrap: wrap;
  margin-bottom: 20px;
}
```

Make sure to add “flex-wrap: wrap” to your row class!

COLUMNS

We can create classes for the different options for column widths. We can then apply these classes to any flex item to set its width.

```
.col-1-4 {
  width: 25%;
}

.col-1-3 {
  width: 33.333333%;
}

.col-1-2 {
  width: 50%;
}
```

COLUMNS — RESPONSIVE

We can then create classes for different screen sizes in a media query:

```
@media screen and (max-width: 970px) {
  .col-md-1-4 {
    width: 25%;
  }
  .col-md-1-2 {
    width: 50%;
  }
  .col-md-full {
    width: 100%;
  }
}
```

COLUMNS — RESPONSIVE

We can then create classes for different screen sizes in a media query:

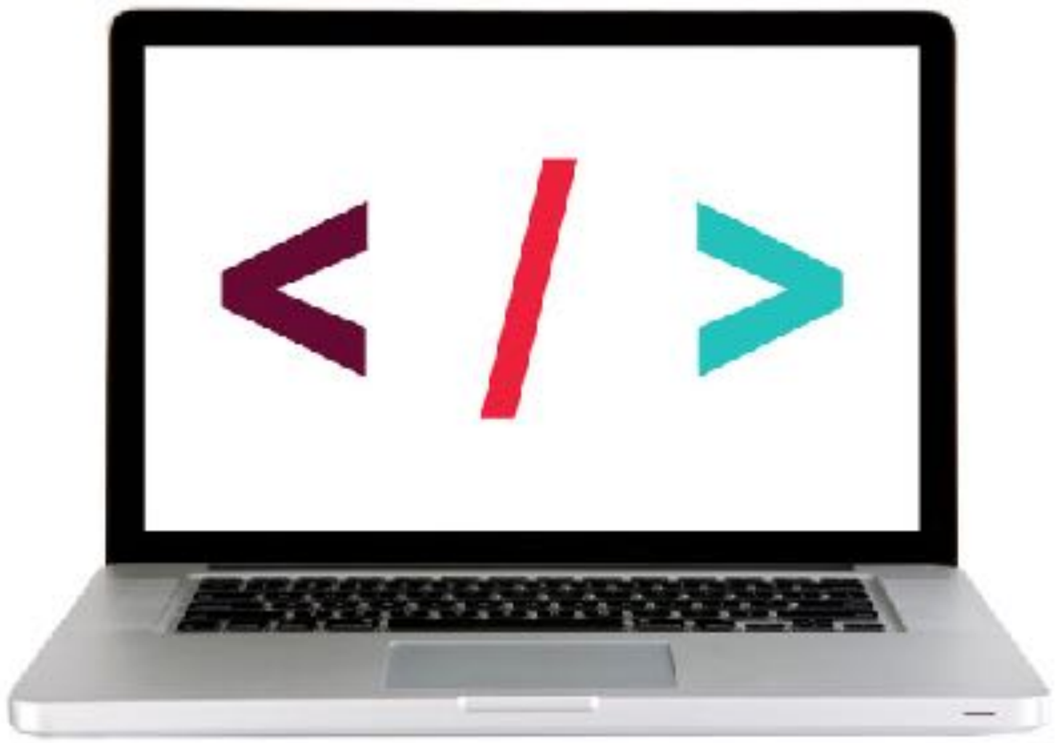
```css
@media screen and (max-width: 750px) {
  .col-sm-full {
    width: 100%;
  }
}
```

COLUMNS — RESPONSIVE

And finally, we can add any classes that apply to our HTML. We can add multiple classes to one element with a space-separated list:

```
<section class="col-1-4 col-md-1-2 col-sm-full">Content</section>
```

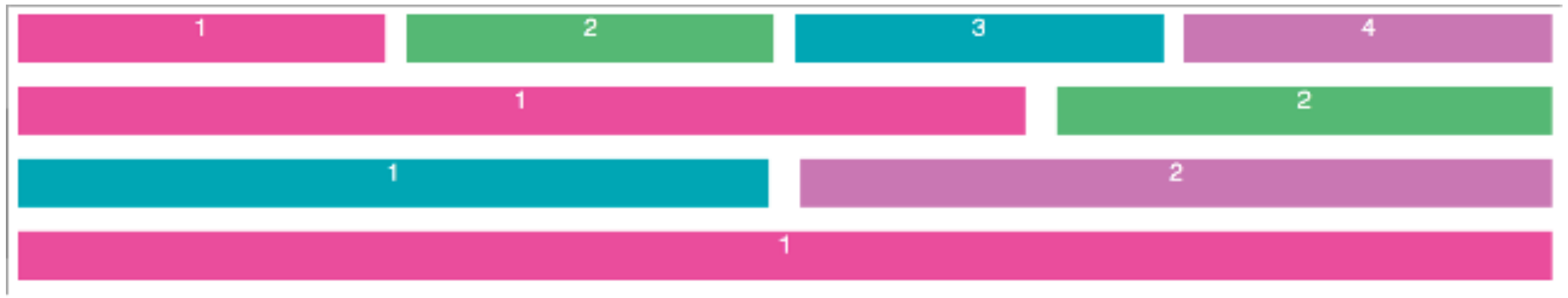
CODEALONG: GRID SYSTEMS



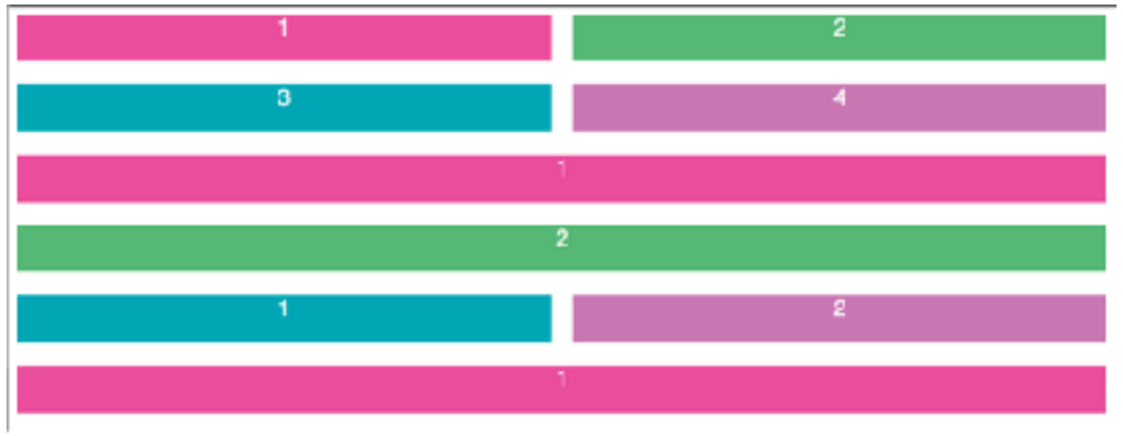
https://codepen.io/larissam/details/oqMppQ/

ACTIVITY – APPLY THE GRID CLASSES!

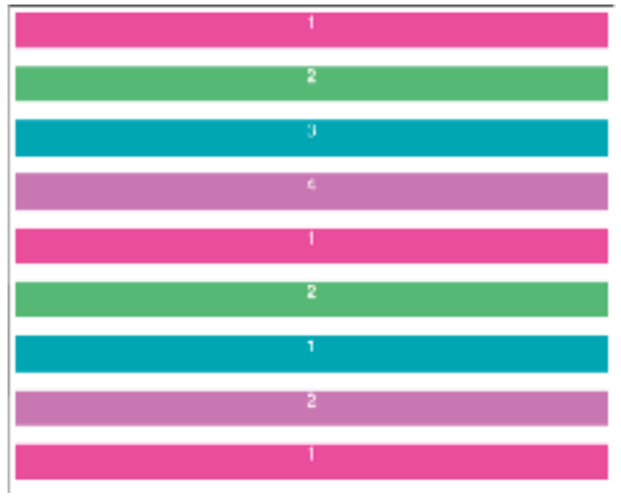
Large (default) viewport:



Medium viewport:



Small viewport:



https://codepen.io/larissam/pen/gxRQrj?editors=1100#0

ACTIVITY: PACIFIC BURGER - RESPONSIVE

EXERCISE

KEY OBJECTIVE

- Practice applying media queries to achieve a responsive layout.

TYPE OF EXERCISE

- Individual/Partner

TIMING

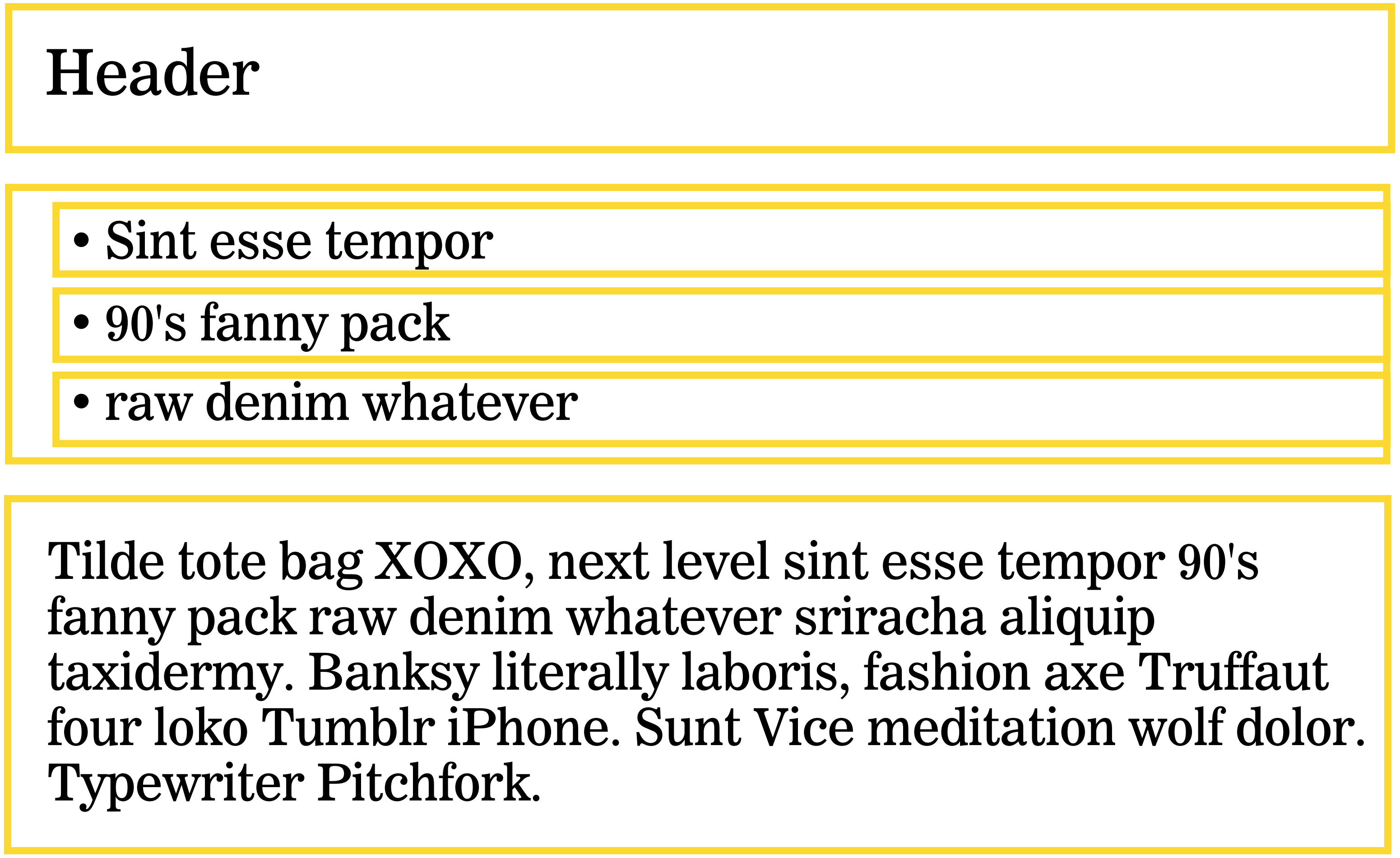
1. Open up the code in starter code > pacific_responsive
2. Add media queries and grid system classes “row”, “col-2-3”, “col-1-3”, “col-sm-full”, etc to create the responsive layouts in the mocks

FEWD

DISPLAY REVIEW

BLOCK ELEMENTS

An **block element** will always start on a new line.


Header

- Sint esse tempor
- 90's fanny pack
- raw denim whatever

Tilde tote bag XOXO, next level sint esse tempor 90's fanny pack raw denim whatever sriracha aliquip taxidermy. Banksy literally laboris, fashion axe Truffaut four loko Tumblr iPhone. Sunt Vice meditation wolf dolor. Typewriter Pitchfork.

INLINE ELEMENTS

An **inline element** does not start on a new line and only takes up as much space as needed.

Tilde tote bag XOXO, next level sint esse tempor 90's fanny pack raw denim whatever sriracha aliquip taxidermy. Banksy **literally** laboris, fashion axe Truffaut four loko Tumblr iPhone. Sunt Vice meditation *wolf* dolor. Typewriter www.Pitchfork.com.

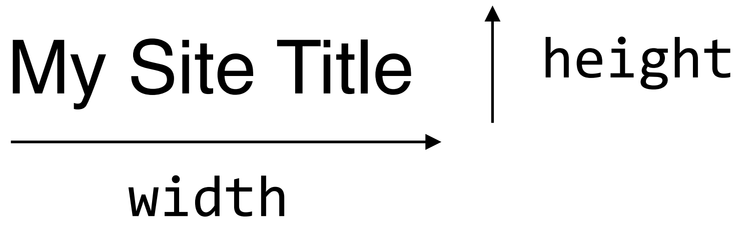
Banksy literally laboris, fashion axe Truffaut four loko Tumblr iPhone. Sunt Vice meditation wolf dolor. Typewriter www.Pitchfork.com.

DISPLAY

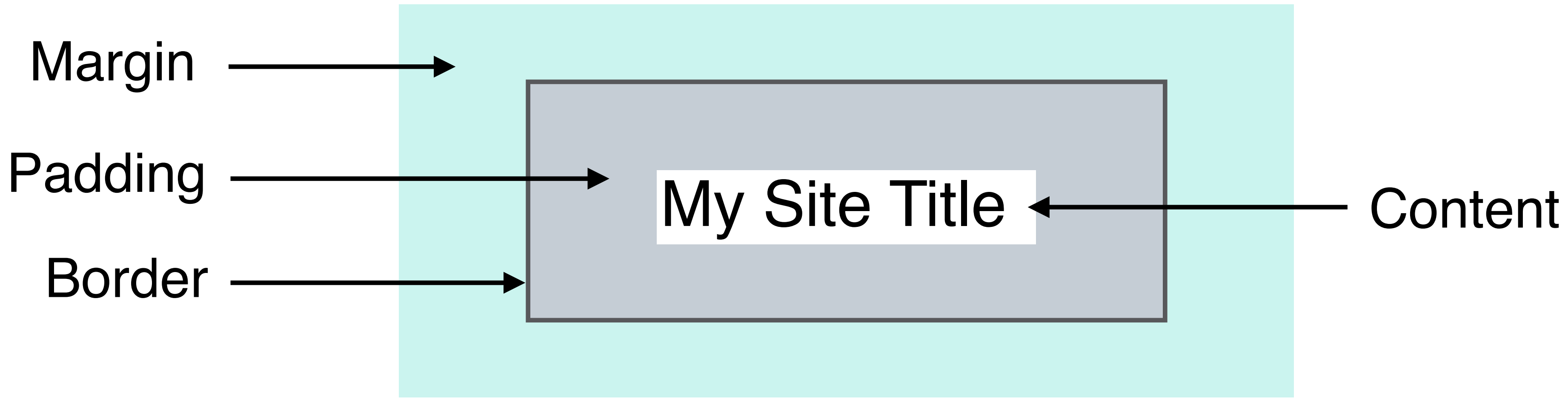
You can change whether elements are displayed as inline or block elements by using the **`display`** property.

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

CONTENT



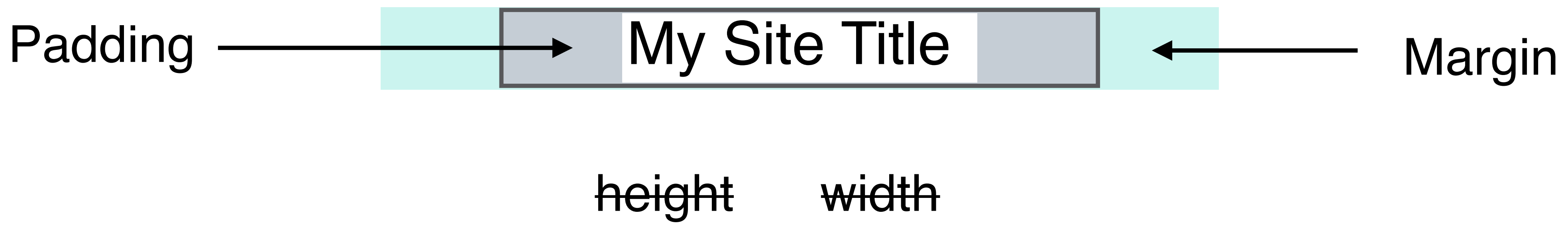
BOX MODEL — BLOCK ELEMENTS



DIMENSION – A KEY DIFFERENCE BETWEEN INLINE AND BLOCK

If you try to add dimension to an inline element:

- `Padding` and `margin` will only apply to the *left and right*
- `Width` and `height` will have *no effect*



DISPLAY — INLINE (RARE)

To make a block-level element act like an inline element:

BEFORE:

Monday

Tuesday

Wednesday

```css
li {
  display: inline;
}
```

Monday   Tuesday   Wednesday

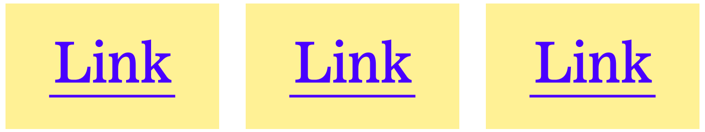
- Elements will sit next to each other
- **Still can't set a width, height, or margin and padding on top and bottom**

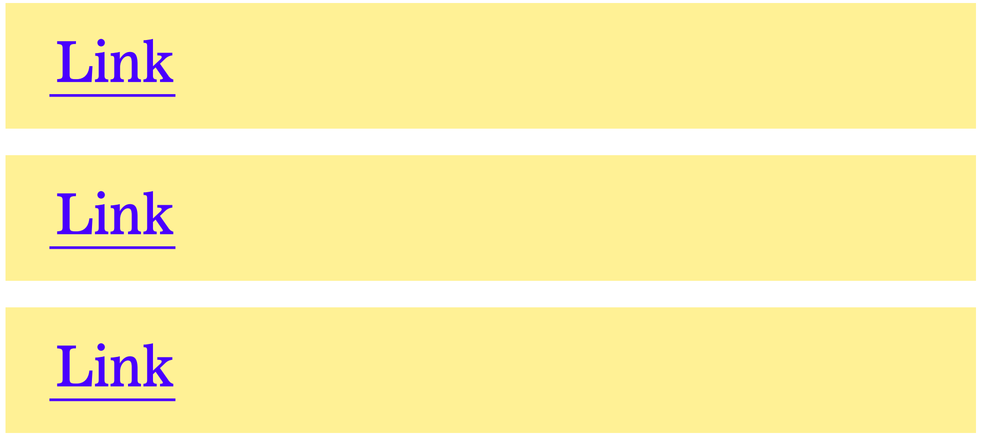
DISPLAY — BLOCK

Make an inline element act like a block-level element:

```
a {
  display: block;
}
```

BEFORE:


Link Link Link


Link

Link

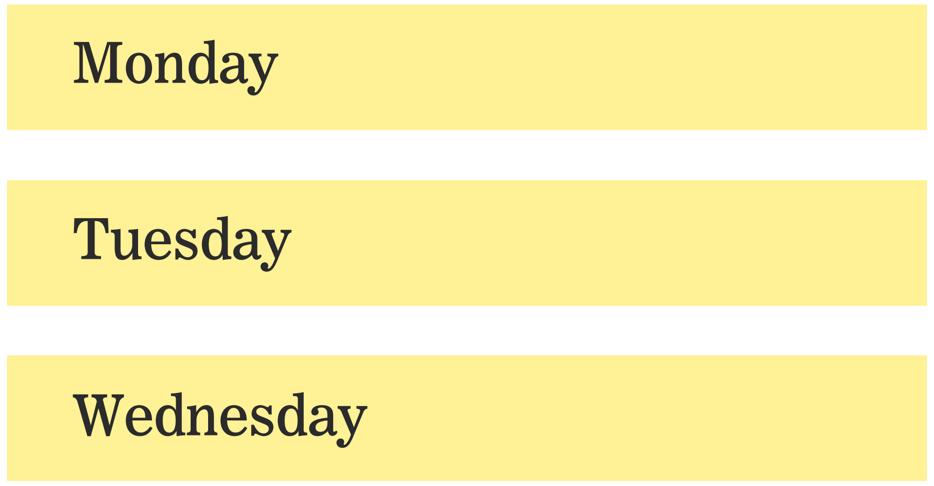
Link

- Elements will stack on top of each other
- We can add all dimensions (width, height, padding, margin)

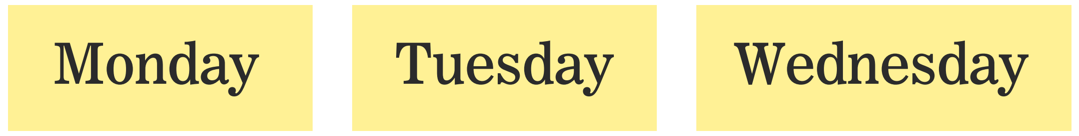
DISPLAY — INLINE-BLOCK

Make a **`block`** *or* **`inline`** element flow like an **`inline`** element, while allowing us to set a **`width`**, **`height`**, **`padding`**, and **`margin`**:

BEFORE:



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

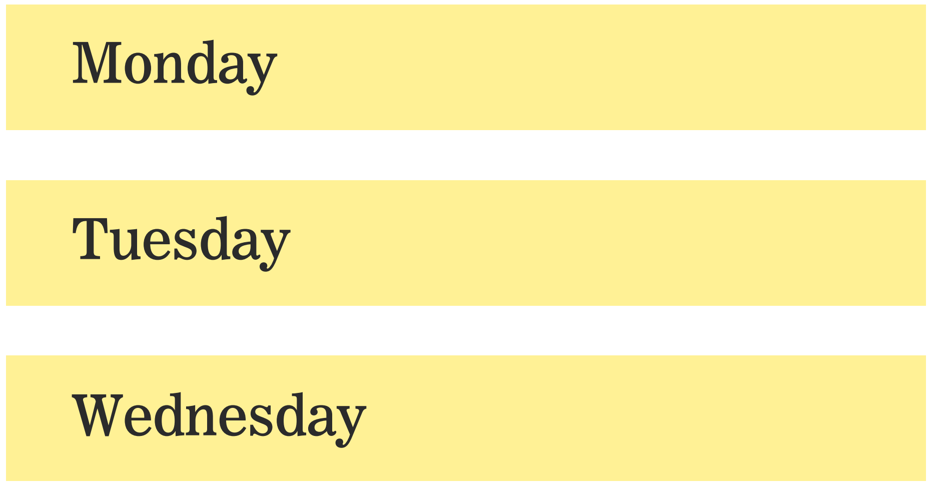


- Elements will sit on a line next to each other
- **We can now set a width, height, and margin & padding on top and bottom!!**

DISPLAY — NONE

Hide an element from the page:

BEFORE:


Monday

Tuesday

Wednesday

```css
li {
  display: none;
}
```

- Elements will be hidden from the page

TEXT-ALIGN

	TEXT-ALIGN
BLOCK	yes
INLINE / INLINE-BLOCK	no

ACTIVITY — DISCUSS DISPLAY

EXERCISE

KEY OBJECTIVE

- Review display properties in pairs

TIMING

1. What is the difference between display: inline, inline-block, and block?
2. What is an example of an inline element, an inline-block element, and a block element?

CSS POSITIONING

ACTIVITY — POSITIONING

EXERCISE

KEY OBJECTIVE

- Differentiate between various positioning techniques.

TYPE OF EXERCISE

- Groups

TIMING

1. Complete steps 1 - 4B in `positioning_intro`
2. Bonus: If you finish early, look up "z-index CSS". What does this property do?

TURN, CHAT, AND POST IN SLACK!

- What features can be built using positioning?
- What does z-index do?
- Find a website that uses positioning and post it in Slack!

STATIC POSITIONING

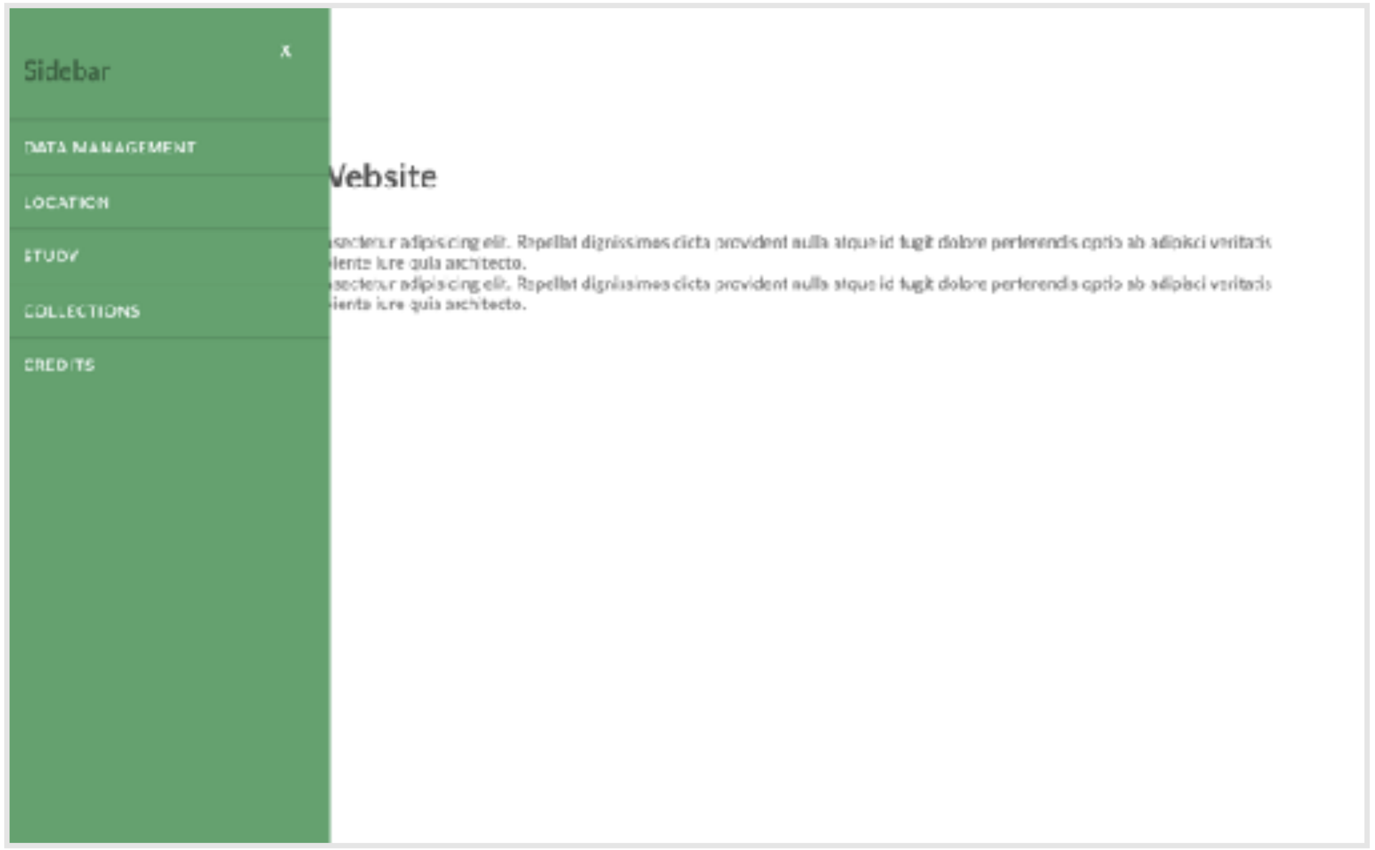
- By default, elements on a page are similar to these wooden blocks.
- They will stack one on top of the other in the same order that they are placed in an HTML file. This is referred to as the "normal flow" of a document.

STATIC POSITIONING

- We can use the `position` property in our CSS to take elements out of the normal flow of the document and specify where they should appear.

```css
.my-class {
   position: fixed;
}
```

CSS POSITIONING — SIDEBAR

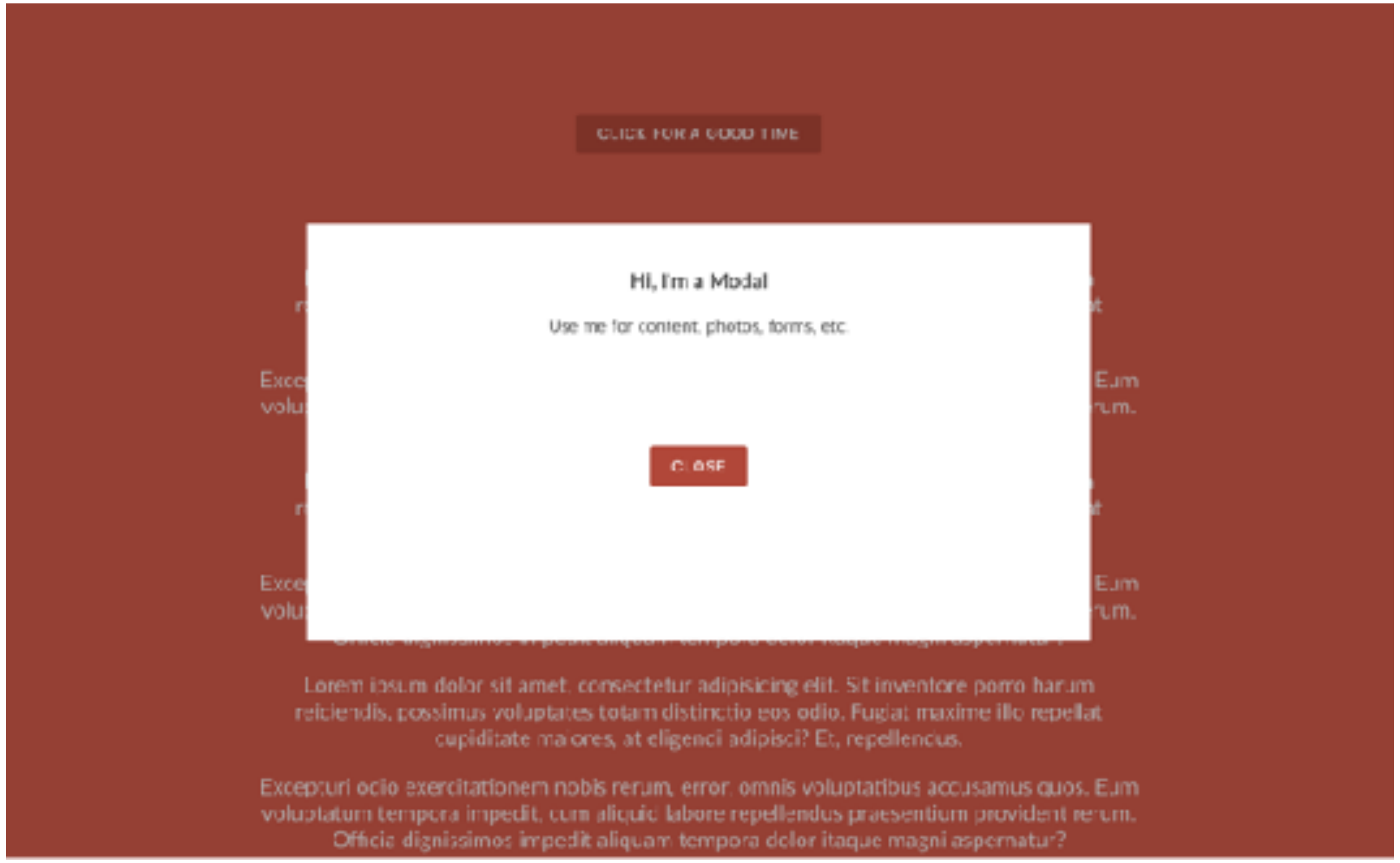


Vebsite

sectetur adipisicing elit. Repellat dignissimos dicta provident nulla atque id fugit dolore perferendis optio ab adipisci veritatis
iente iure quia architecto.
sectetur adipisicing elit. Repellat dignissimos dicta provident nulla atque id fugit dolore perferendis optio ab adipisci veritatis
iente iure quia architecto.

https://larissam.github.io/fewd_sidebar_menu_example/

CSS POSITIONING — MODAL WINDOW



Hi, I'm a Modal

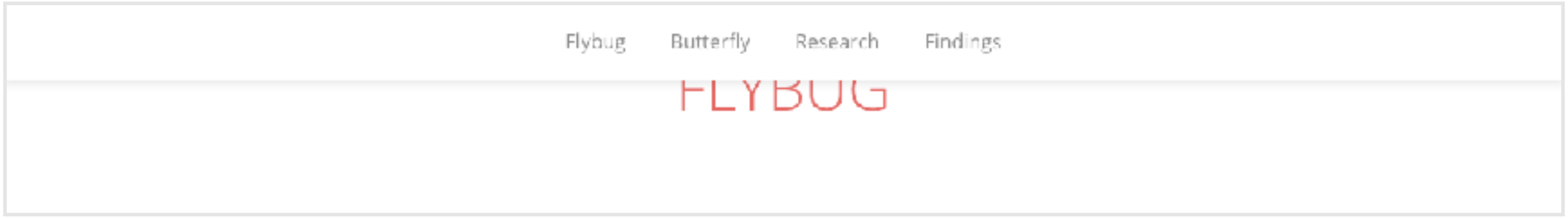
Use me for content, photos, forms, etc.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sit inventore porro harum reiciendis, possimus voluptates totam distinctio eos odio. Fugiat maxime illo repellat cupiditate maiores, at eligendi adipisci? Et, repellendus.

Excepturi odio exercitationem nobis rerum, error omnis voluptatibus accusamus quos. Eum voluptatum tempora impedit, cum aliquid labore repellendus praesentium provident rerum. Officia dignissimos impedit aliquam tempora dolor itaque magni aspernatur?

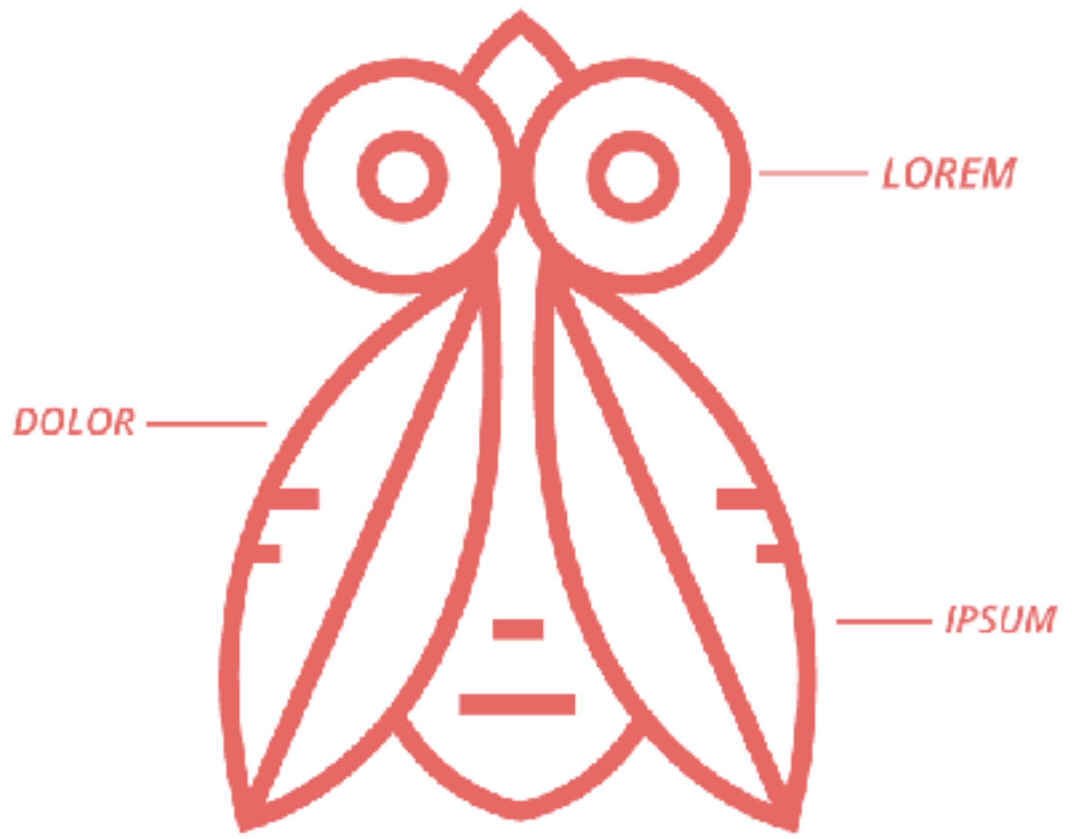
https://larissam.github.io/fewd_modal_example/

CSS POSITIONING — STICKY NAV



https://larissam.github.io/fewd_sticky_nav_example/

CSS POSITIONING — LABELS FOR IMAGE



https://larissam.github.io/fewd_sticky_nav_example/

STATIC POSITIONING

CSS POSITIONING

STATIC

RELATIVE

FIXED

ABSOLUTE

STATIC POSITIONING

- The default position on each element is `static`.
- Elements with a position of `static` will appear in order and stack on top of each other, like we would expect.

```css
yourSelectorHere {
   position: static;
}
```

FIXED POSITIONING

CSS POSITIONING

STATIC

RELATIVE

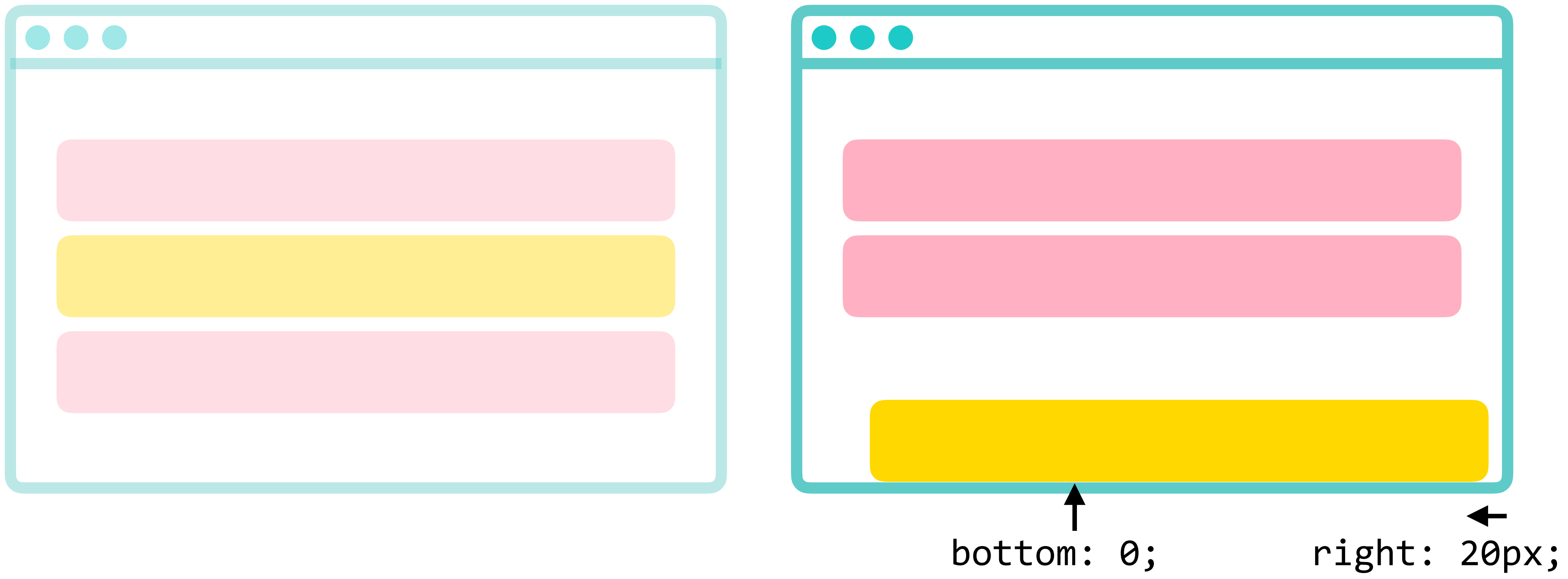
FIXED

ABSOLUTE

FIXED POSITIONING

- Positioned in relation to *the browser window*
- When the user scrolls, it stays in the same place.
- Use **`right`**, **`top`**, **`left`** and **`bottom`** properties to specify where the element should go in relation to the browser window.

```
yourSelectorHere {
   position: fixed;
   bottom: 0;
   right: 20px;
}
```



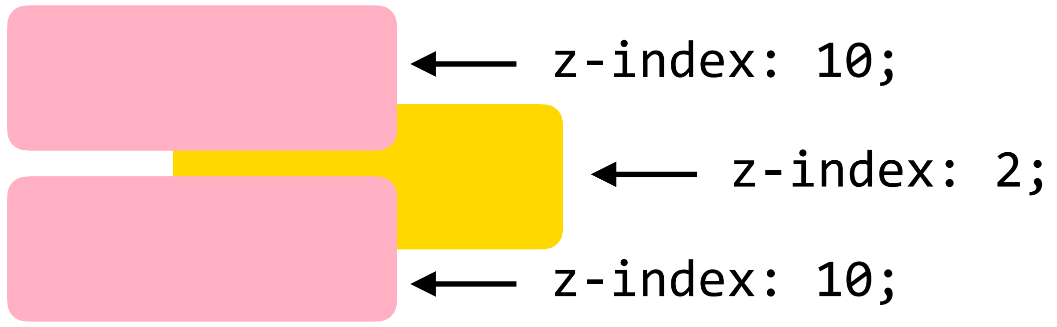
Z-INDEX

OVERLAPPING ELEMENTS — Z-INDEX

- With **`relative`**, **`absolute`**, and **`fixed`** positioning, elements can overlap.
- We can use **`z-index`** to control which elements are layered on top of each other.
- This property takes a number — the higher the number the closer that element is to the front.

```
.yellow {
  z-index: 2;
}

.pink {
  z-index: 10;
}
```



Think of this like 'bring to front' and 'send to back' in programs like Adobe Illustrator.

ACTIVITY — FIXED NAV

EXERCISE

KEY OBJECTIVE

- Practice using CSS positioning

LOCATION

- Starter Code > creepy_crawlers

TIMING

1. Follow step 1 in style.css

RELATIVE POSITIONING

CSS POSITIONING

STATIC

RELATIVE

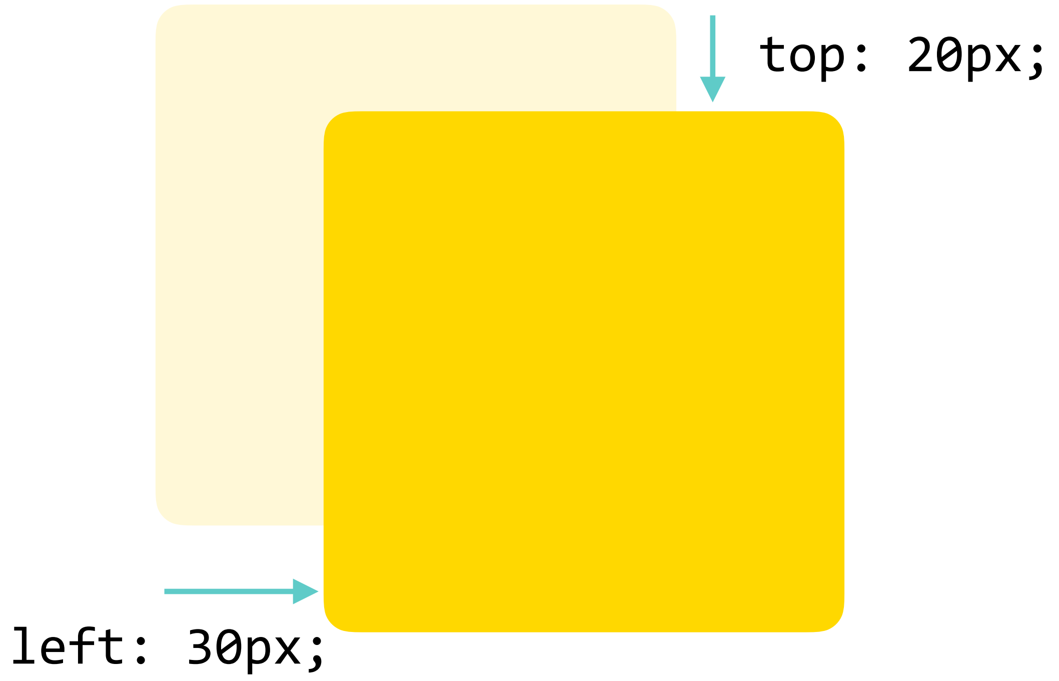
FIXED

ABSOLUTE

RELATIVE POSITIONING

- Moves an element *relative to where it would have been in normal flow*.
- For example: `left: 20px` adds 20px to an element's **left** position

```
yourSelectorHere {
    position: relative;
    top: 20px;
    left: 30px;
}
```



ANIMATION

ABSOLUTE

CSS POSITIONING

STATIC

RELATIVE

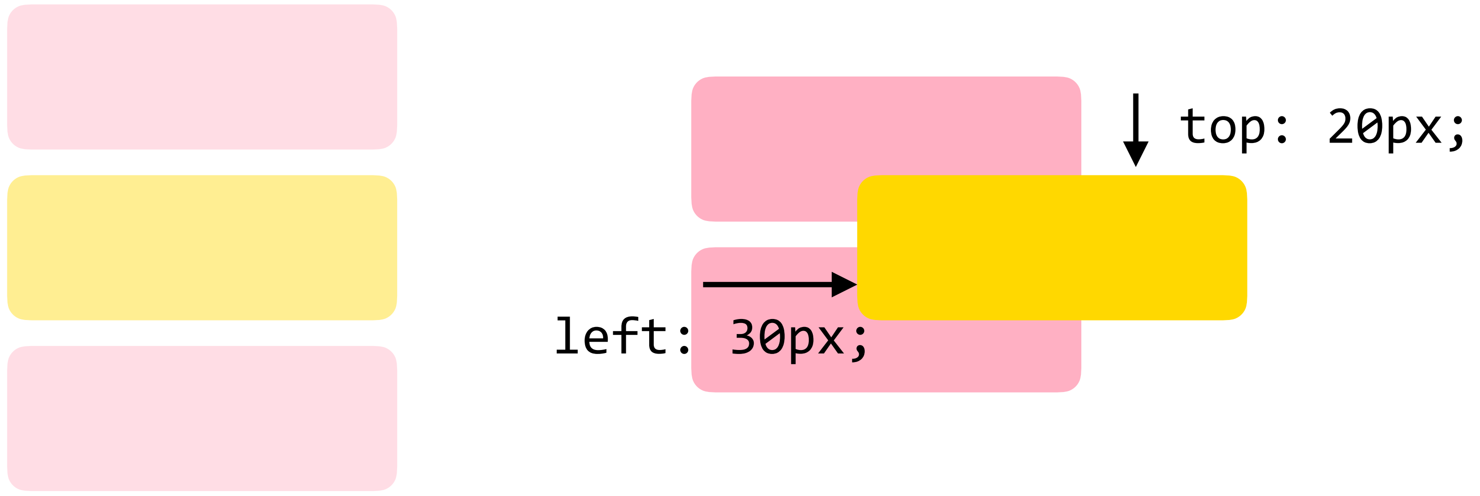
FIXED

ABSOLUTE

ABSOLUTE POSITIONING

- Element is taken out of the normal flow of the document.
- No longer affects the position of other elements on the page (they act like it's not there).
- You can add the *right*, *top*, *left* and *bottom* properties to specify where the element should appear

```
yourSelectorHere {
  position: absolute;
  top: 20px;
  left: 30px;
}
```

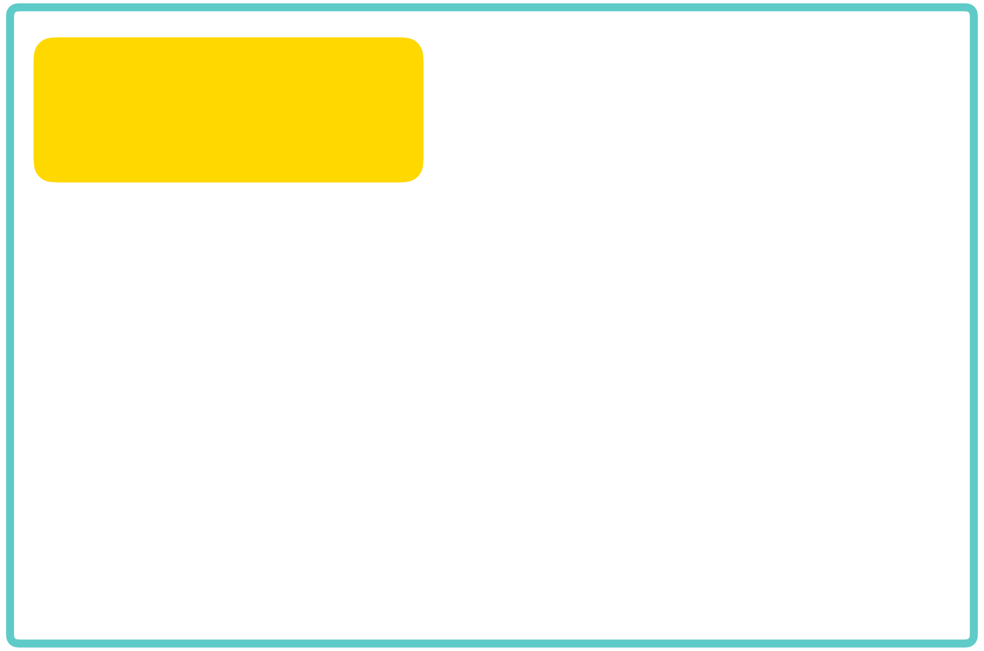


POSITIONING THINGS ABSOLUTELY

Parent we want child to be positioned relative to

```
<section>
   <div class="info"></div>
</section>
```

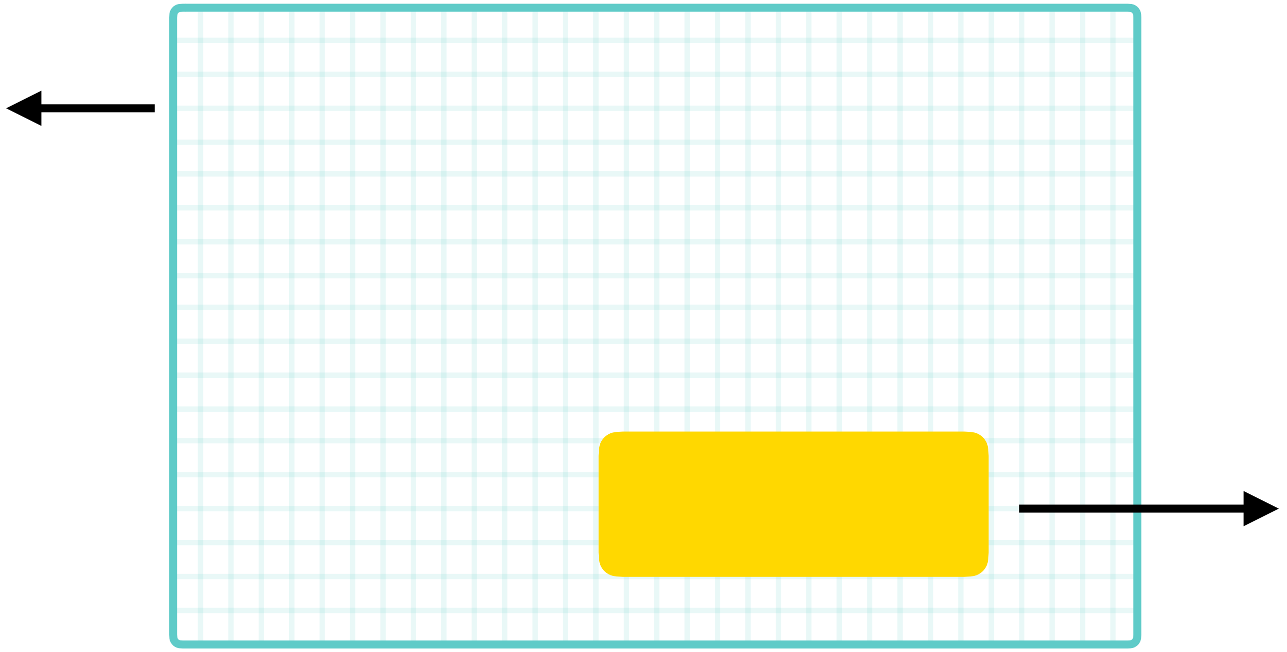
Child we want to position



POSITIONING THINGS ABSOLUTELY

Parent:

`position: relative;`



Child:

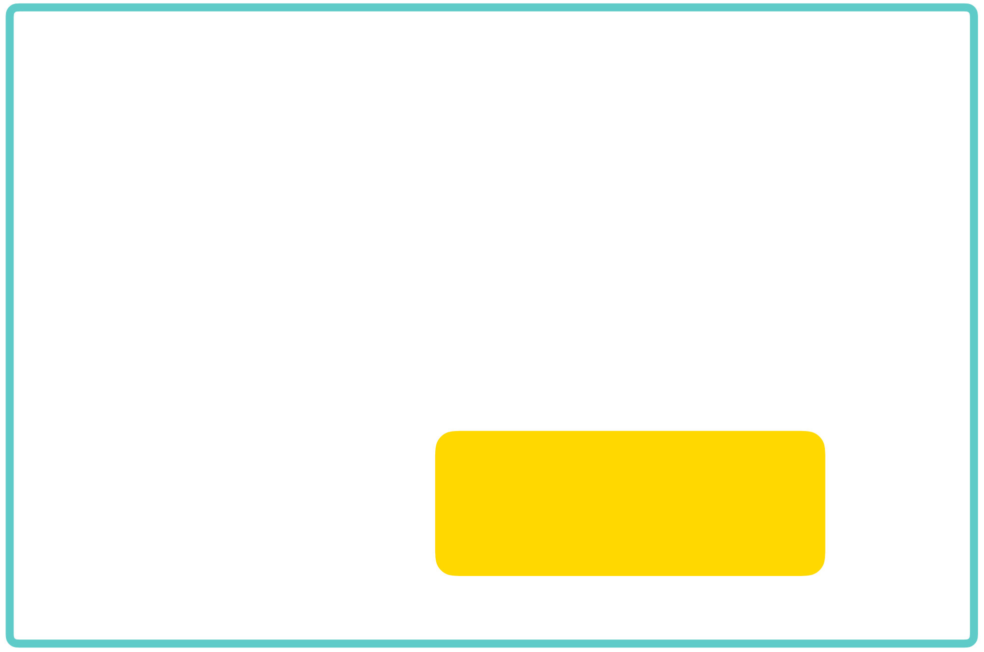
- `position: absolute;`
- `top, bottom, left, or right`

POSITIONING THINGS ABSOLUTELY

```
<section>
   <div class="info"></div>
</section>
```

```
section {
  position: relative;
}

.info {
  position: absolute;
  bottom: 20px;
  right: 50px;
}
```



ACTIVITY — ABSOLUTE POSITIONING

EXERCISE

KEY OBJECTIVE

- Practice using CSS positioning

LOCATION

- Starter Code > creepy_crawlers

TIMING

8 min	1. Follow step 2 in main.css

WANT TO LEARN MORE?

Resources for more info/examples:

- A List Apart: CSS Positioning 101

ACTIVITY — DISCUSS POSITIONING

EXERCISE

KEY OBJECTIVE

- Differentiate between various positioning techniques.

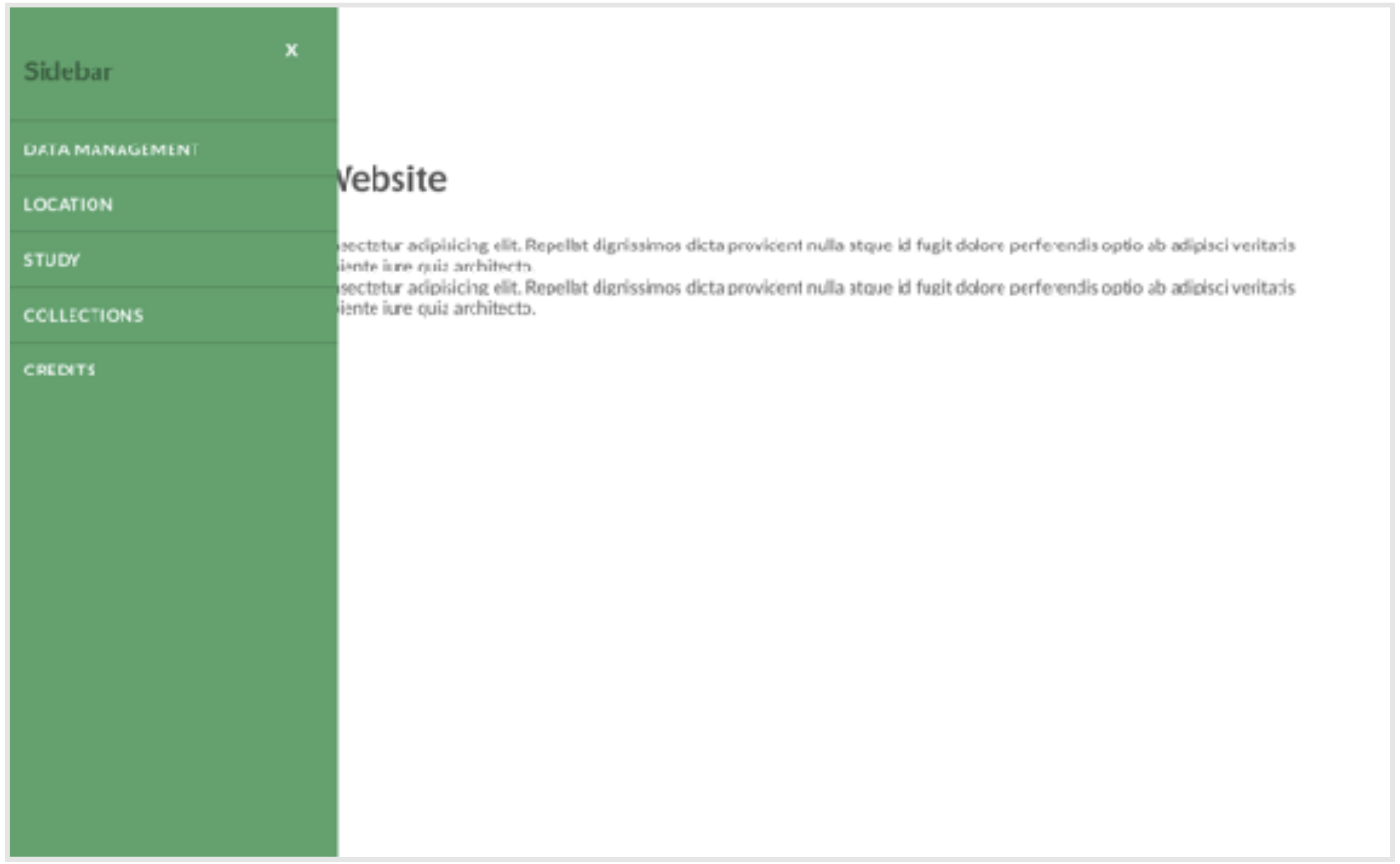
TYPE OF EXERCISE

- Work in groups

TIMING

1. We want to position an element inside another element. One is a parent and the other is the child.
2. Which would we set a position of relative and which would we give a position of absolute? Do we need to add anything else?

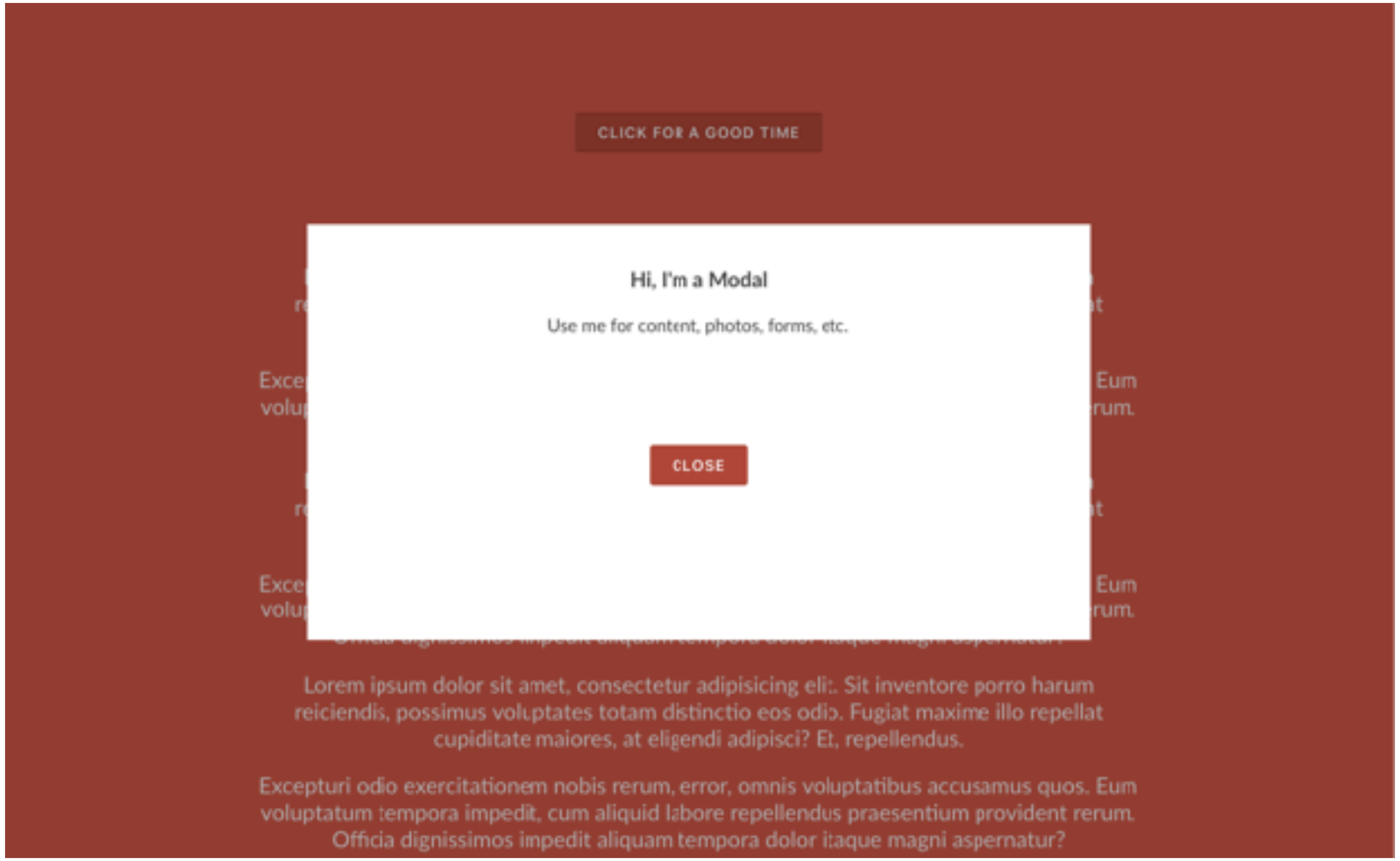
CSS POSITIONING — SIDEBAR



Vebsite

ectetur adipisicing elit. Repellat dignissimos dicta provident nulla atque id fugit dolore perferendis optio ab adipisci veritatis
iente iure quia architecto.
sectetur adipisicing elit. Repellat dignissimos dicta provident nulla atque id fugit dolore perferendis optio ab adipisci veritatis
iente iure quia architecto.

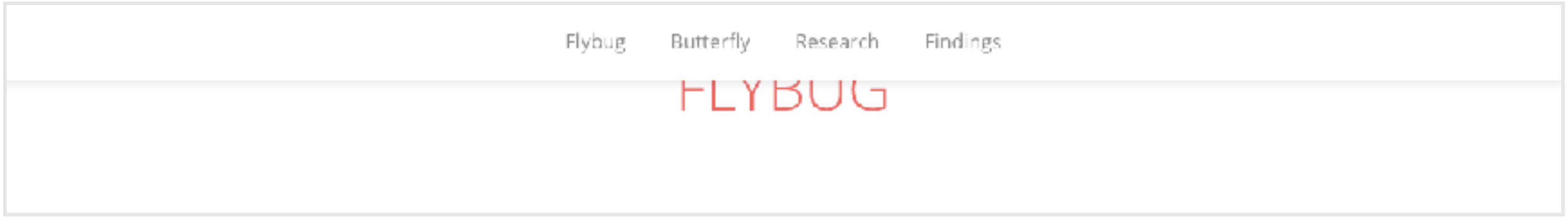
CSS POSITIONING — MODAL WINDOW



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sit inventore porro harum reiciendis, possimus voluptates totam distinctio eos odio. Fugiat maxime illo repellat cupiditate maiores, at eligendi adipisci? Et, repellendus.

Excepturi odio exercitationem nobis rerum, error, omnis voluptatibus accusamus quos. Eum voluptatum tempora impedit, cum aliquid labore repellendus praesentium provident rerum. Officia dignissimos impedit aliquam tempora dolor itaque magni aspernatur?

CSS POSITIONING — STICKY NAV



CSS POSITIONING — LABELS FOR IMAGE

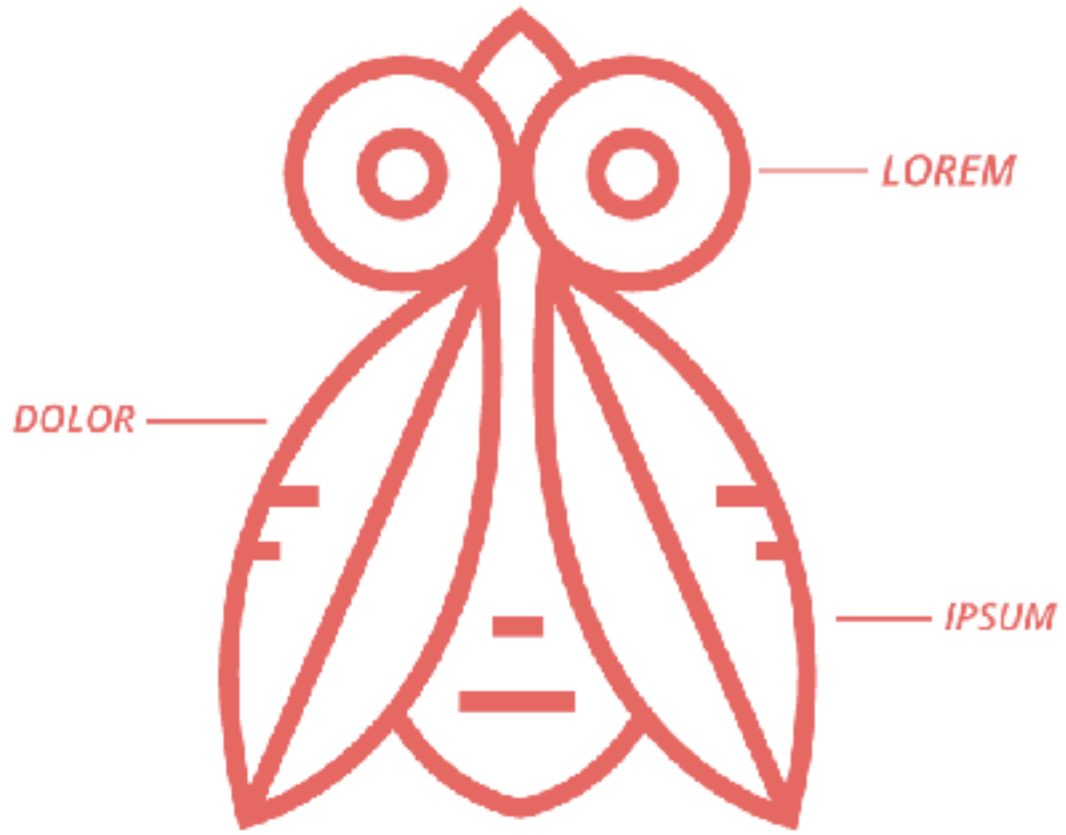


IMAGE OVERLAY CODE ALONG

ACTIVITY - DISCUSSION

EXERCISE

LOCATION

- starter code > image_overlay

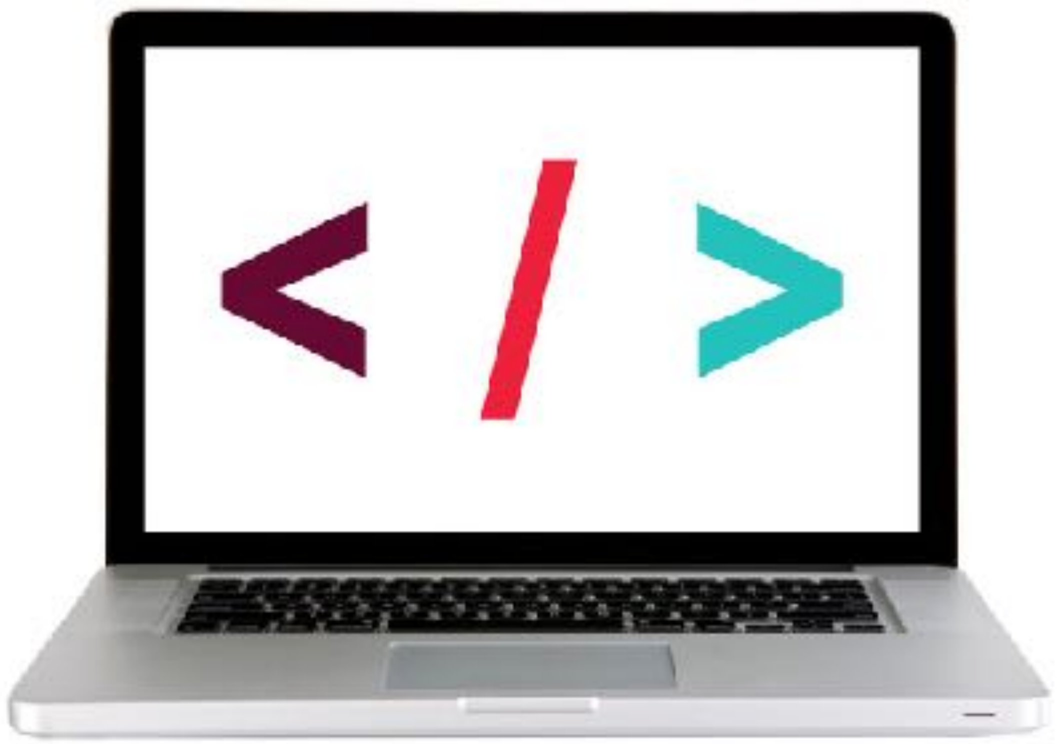
KEY OBJECTIVE

- Discuss positioning with a partner

TIMING

1. Determine what kind of positioning we will need.
2. What selectors will we use for these elements?

ACTIVITY – CODE ALONG

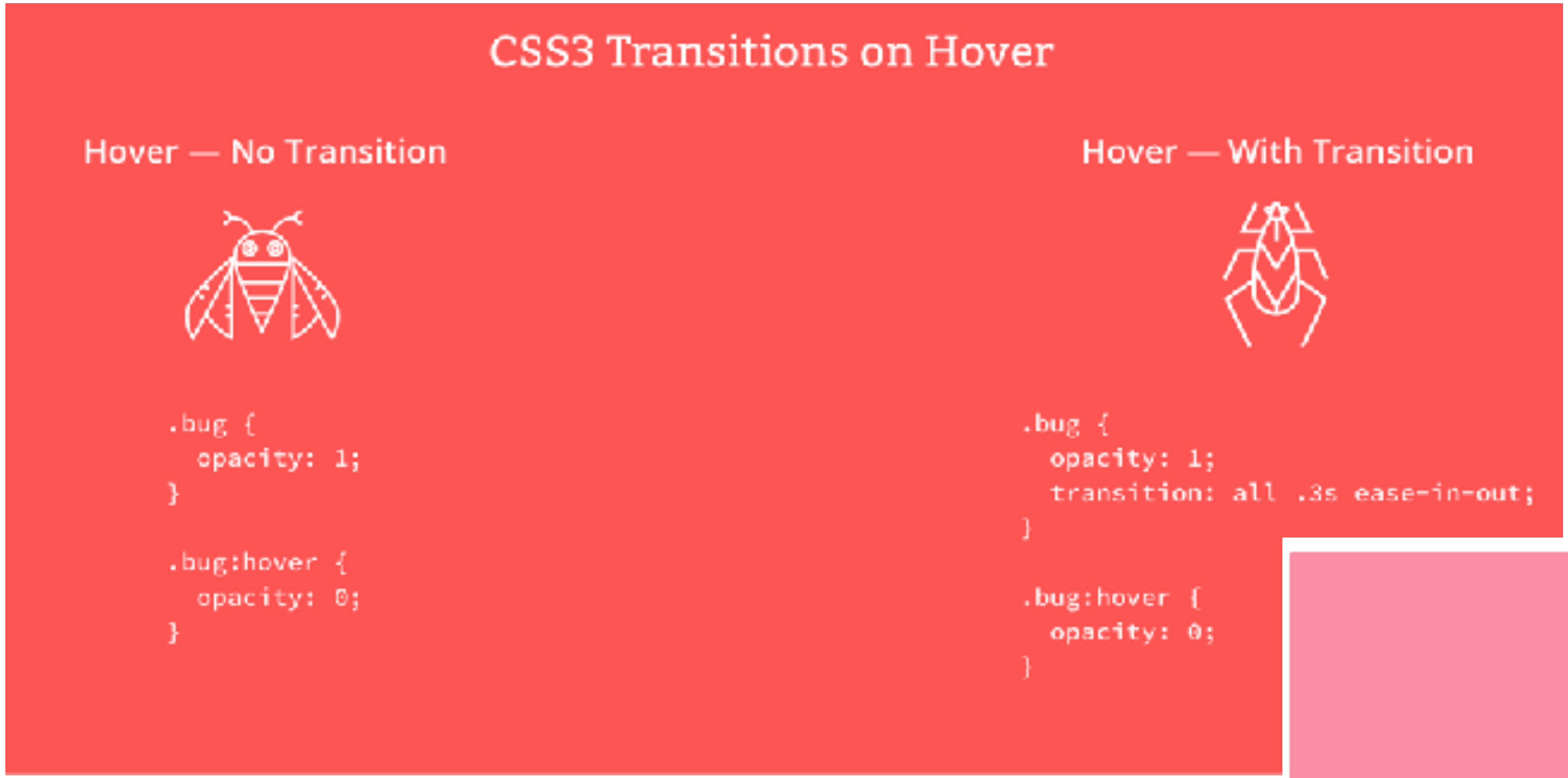


Starter code > image_overlay

ANIMATION

TRANSITIONS

DEMO: LET'S TAKE A CLOSER LOOK



CSS3 Transitions on Hover

Hover — No Transition

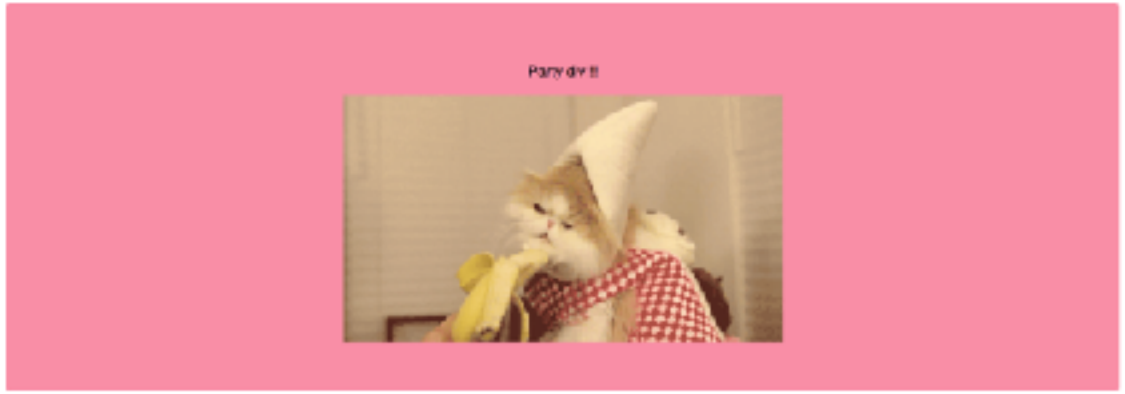
```
.bug {
  opacity: 1;
}

.bug:hover {
  opacity: 0;
}
```

Hover — With Transition

```
.bug {
  opacity: 1;
  transition: all .3s ease-in-out;
}

.bug:hover {
  opacity: 0;
}
```



Syntax: W3 Schools

TRANSITIONS

- Provide a way to control animation speed when changing properties
- Instead of having property changes take effect immediately, you can have them take place over a period of time.

```
yourSelectorHere {
    transition: what-to-transition animation-duration timing-function delay;
}
```

EXAMPLE:

```
transition: all 350ms ease-in-out;
```

TRANSITIONS - TRANSITION-PROPERTY

PROPERTIES TO ANIMATE

- Can specify a specific property to transition or "all" to transition all properties
- *Default:* all

```css
div {
    transition: opacity 0.5s;
}
```

```css
div {
    transition: all 0.5s;
}
```

TRANSITIONS - TRANSITION-DURATION

DURATION

- A time value, defined in seconds or milliseconds

```
div {
    transition: height 0.5s;
}
```

```
div {
    transition: height 300ms;
}
```

TRANSITIONS

TIMING FUNCTION

- Describes how a transition will proceed over its duration, allowing a transition to change speed during its course.
- Timing functions: **`ease, linear, ease-in, ease-out, ease-in-out`**

```css
div {
    transition: opacity 0.5s ease;
}
```

TRANSITIONS

DELAY

- Length of time before the transition starts

```
div {
    transition: background-color 0.5s ease 2s;
}
```

MORE FUN WITH TRANSITIONS — CODROPS

Fun CSS button styles: Creative buttons

Icon hover effects: Icon Hover Effects

Modal dialogue effects (advanced): Dialogue Effects

ACTIVITY — BUTTON LAB

EXERCISE

KEY OBJECTIVE

- Practice using CSS transitions

TYPE OF EXERCISE

- Partner Lab

TIMING

1. Work with a partner!
2. Open the starter code > transition_button_lab
3. Add styles to the button
4. Then, add :hover styles to the button
5. Use the CSS transitions to make it smooth!

TRANSFORMATIONS

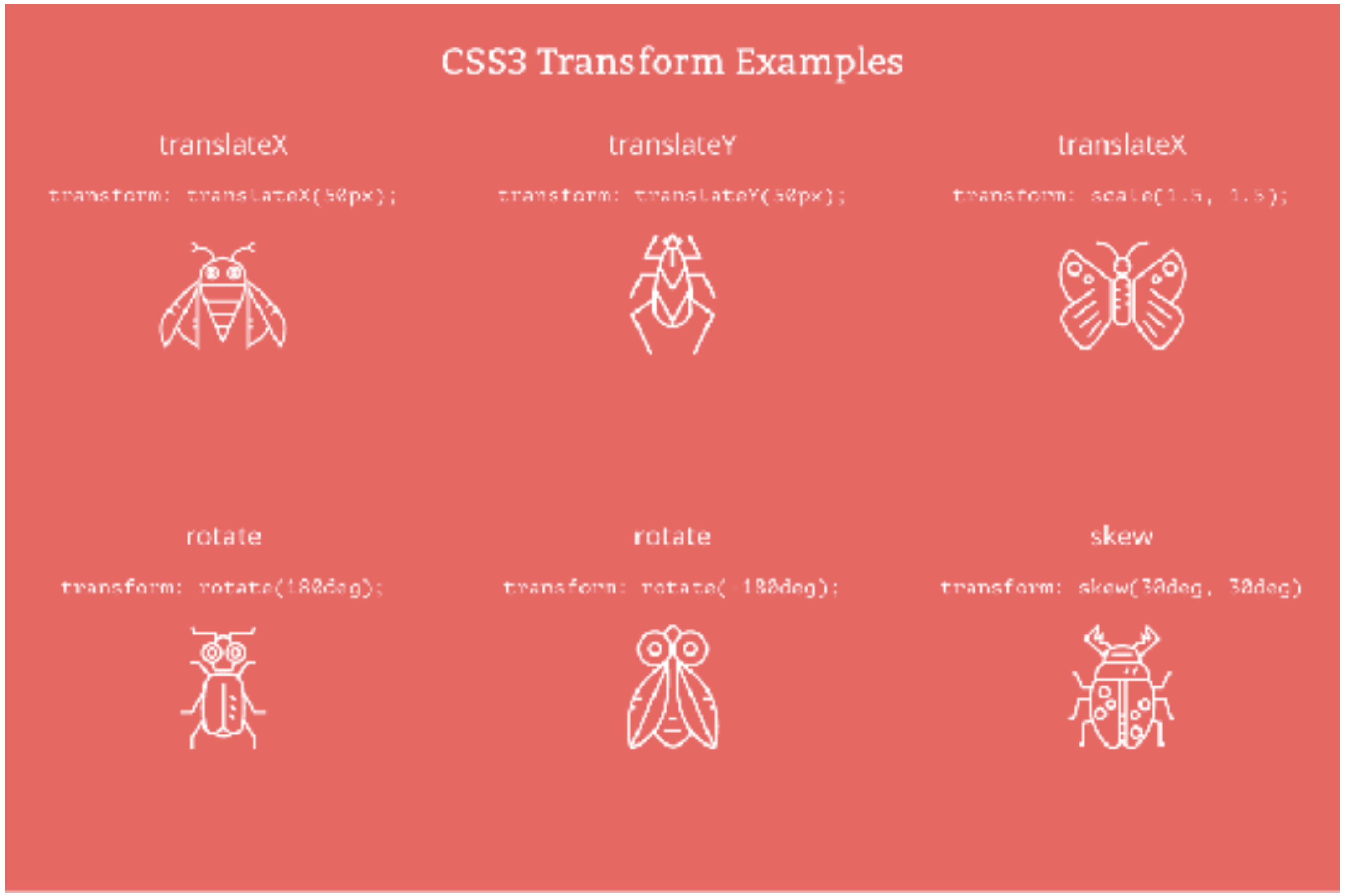
TRANSFORMATIONS

- Easy way to manipulate HTML elements
- Generally used in connection with animations (on hover, JS, or CSS keyframe animations)

```
yourSelectorHere:hover {
  transform: scale(1.1);
}
```

https://css-tricks.com/almanac/properties/t/transform/

DEMO — TRANSFORM



CSS3 Transform Examples

translateX

```
transform: translateX(50px);
```

translateY

```
transform: translateY(50px);
```

translateX

```
transform: scale(1.5, 1.5);
```

rotate

```
transform: rotate(180deg);
```

rotate

```
transform: rotate(-180deg);
```

skew

```
transform: skew(30deg, 30deg)
```

https://larissam.github.io/fewd_transform_examples/

TRIGGERING TRANSITIONS ON HOVER

ACTIVITY — TRIGGERING TRANSITIONS (HOVER)

EXERCISE

KEY OBJECTIVE

- Practice using CSS transitions

TYPE OF EXERCISE

- Individual/Partner Lab

TIMING

1. Follow the instructions in starter code > transform_bug > style.css
2. Use the “transform examples” site as a reference! https://larissam.github.io/fewd_transform_examples/

WEEKLY OVERVIEW

WEEK 4	Responsive Design / CSS Positioning
WEEK 5	Forms / Final Project Lab
WEEK 6	JS Intro

LEARNING OBJECTIVES

- Differentiate between inline, block, and inline-block elements
- Use the `position` property to position elements on the page
- Utilize transitions to add basic animations on hover

EXIT TICKETS