HTML & CSS: Fundamentals of Development

Instructor: Beck Johnson

Week 5

SESSION OVERVIEW

- Review Week Four
- CSS positioning
- Responsive design
- Fun CSS tricks
- Evaluations

REVIEW!

{} CLASSES AND IDS

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

- You decide the `class` and `id` values – be descriptive!

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

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

{} CLASSES

Multiple elements can have the same `class`

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

```
.kittens { color: gray; }

<p class="kittens">This will be gray.</p>
```

{} IDS

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

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

```
#kittenContainer { color: gray; }
```

```
<div id="kittenContainer">This will be
gray.</div>
```

{} HOW TO CHOOSE BETWEEN THEM

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`

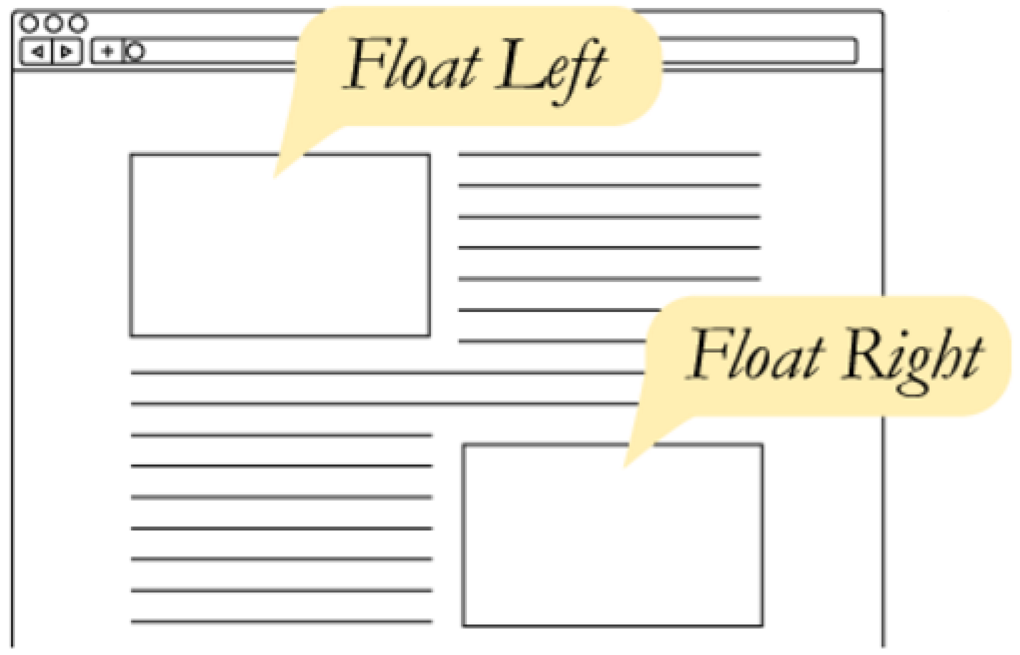
If your element needs to be linked to directly, use `id`

CSS FLOATS

The `float` property takes an element out of the normal flow and "floats" it to the left or right side of its container.

- This allows other content to flow around it

```
img { float: left; }
```

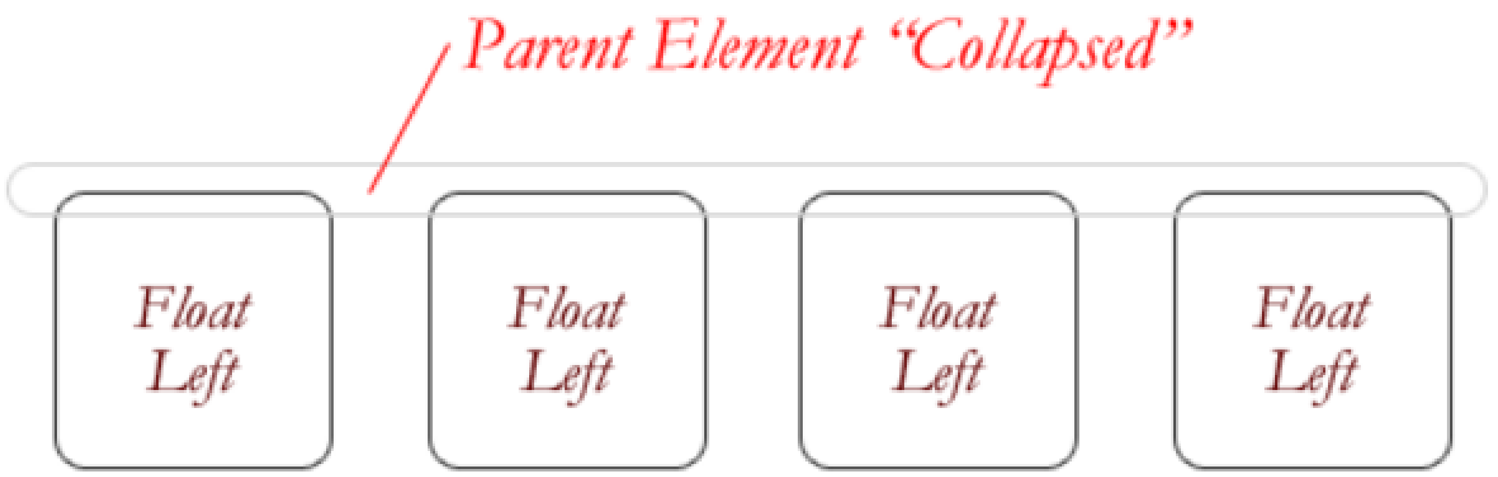


CSS FLOATS

The three values for `float` are:

- `left`
- `right`
- `none`

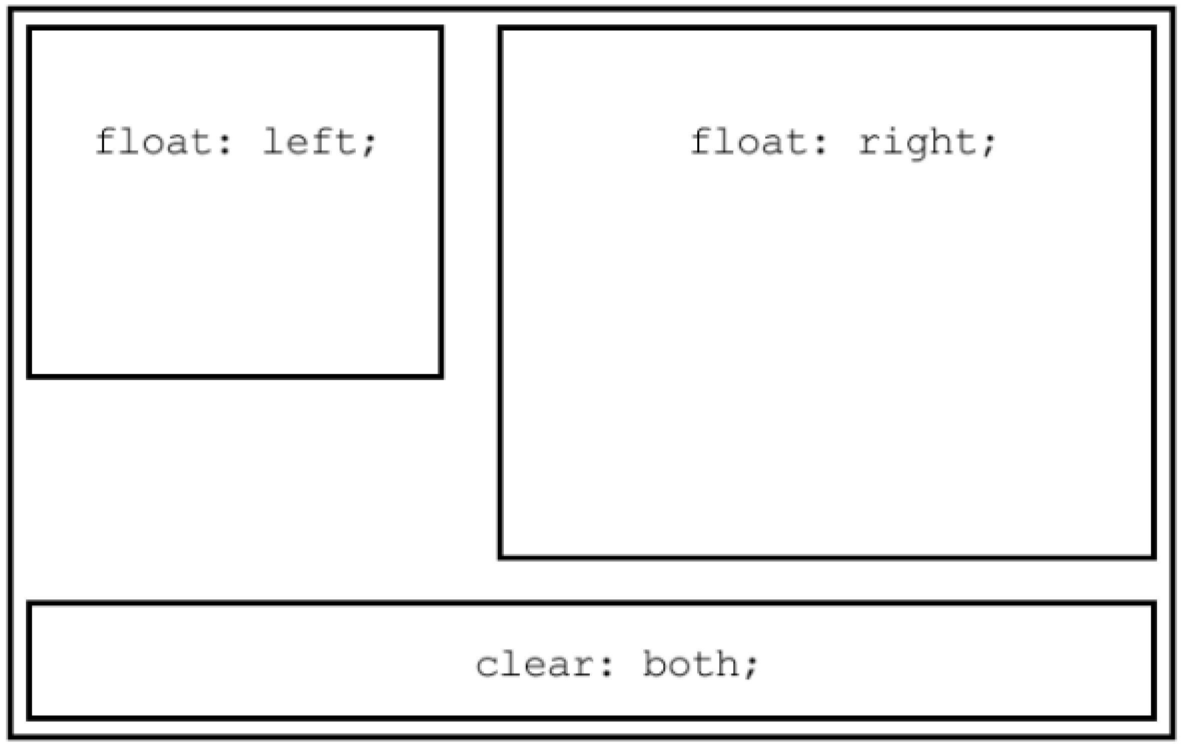
If everything in a container is floated, then the container thinks it's empty.



THE CLEAR PROPERTY

The `clear` property is the sister property to `float`

- It doesn't do much until there are floated elements on the page
- An element with `clear` applied to it will force itself **below** the floated element
- Everything after that will be back in the normal flow
- This "stretches" out the container and keeps it from collapsing



THE OVERFLOW PROPERTY

`overflow` is a CSS property that governs how content looks when it breaks out of its container.

By default, elements have `overflow: visible`, which means all content is fully visible.

`overflow: auto` adds scrollbars when the content is bigger than its container.

QUESTIONS?

CSS POSITIONING

3 WEB LAYOUT PROPERTIES

- `display:` dictates how elements behave within the box model
- `float:` moves elements around within the page flow
- `position:` takes elements entirely out of the page flow

CSS POSITIONING

The `position` property specifies how an element is positioned on the page. Possible values are:

- `static`
- `fixed`
- `absolute`
- `relative`

The default `position` is `static`, which just means that the element obeys whatever its box model rules tell it to do.

CSS POSITIONING

There are 4 directional properties that affect positioning:

- `left`
- `right`
- `top`
- `bottom`

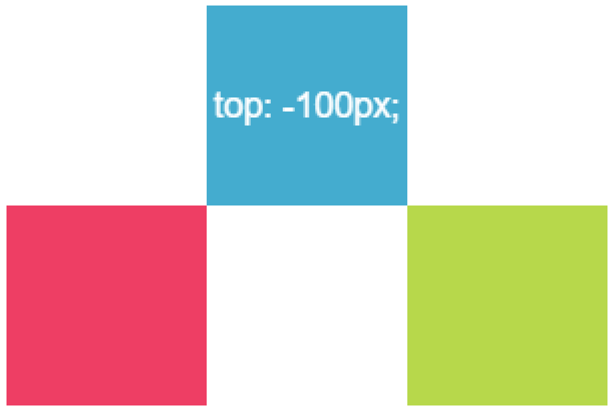
The value is a number (positive or negative) followed by a unit.

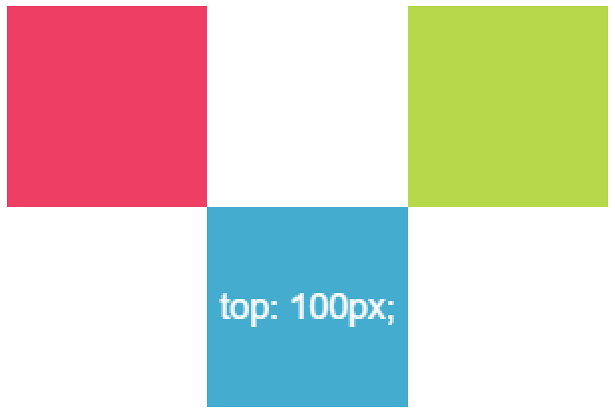
They define how far an element is offset that direction.

CSS POSITIONING

`top` defines how far an element is offset from its original top edge.

- Positive `top` values push an element **down**
- Negative `top` values push an element **up**

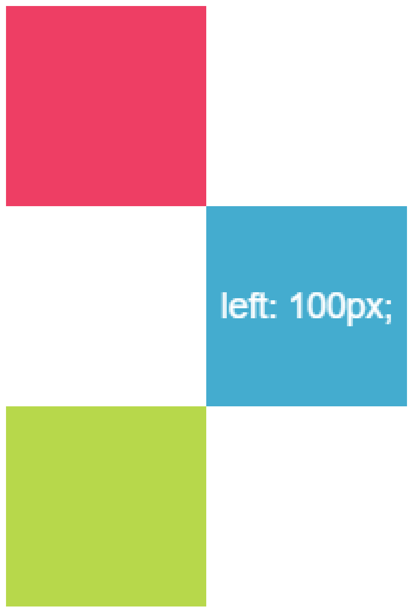


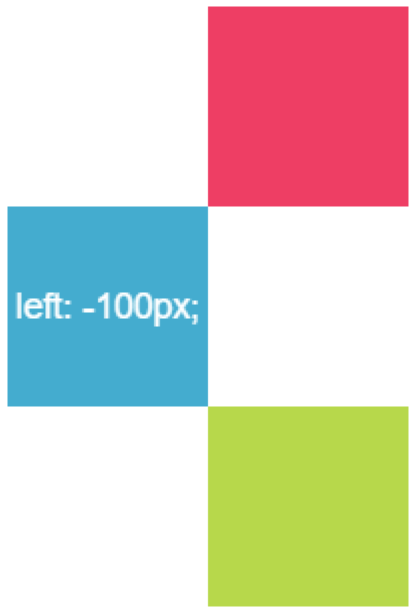


CSS POSITIONING

Similarly, `left` defines how far it's offset from its original left edge.

- Positive `left` values push an element **right**
- Negative `left` values push an element **left**





POSITION: FIXED

`position: fixed` makes content "stick" to the browser window, regardless of where the user scrolls.

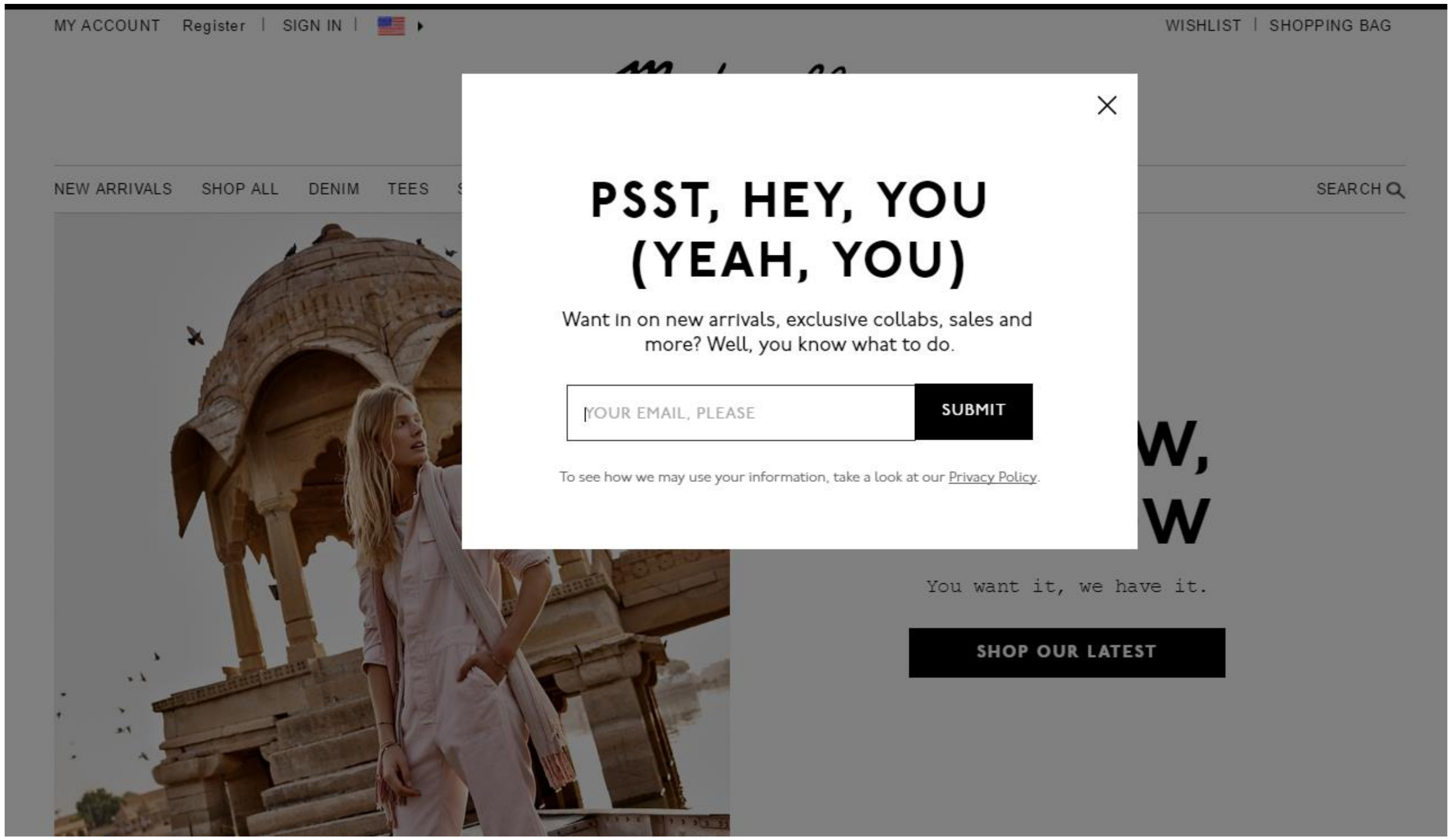
Commonly used to make headers, navigation menus, or sidebars that follow the page as it scrolls.

```
nav {
    position: fixed;
    left: 0;
    top: 0;
}
```

Hard to describe, see a live demo

FIXED

This popup background uses `position: fixed` to grey-out the entire page, even if the user scrolls.



PSST, HEY, YOU (YEAH, YOU)

Want in on new arrivals, exclusive collabs, sales and more? Well, you know what to do.

To see how we may use your information, take a look at our Privacy Policy.

You want it, we have it.

ABSOLUTE

`position: absolute` is a powerful tool that allows you to place any page element exactly where you want it, down to the pixel.

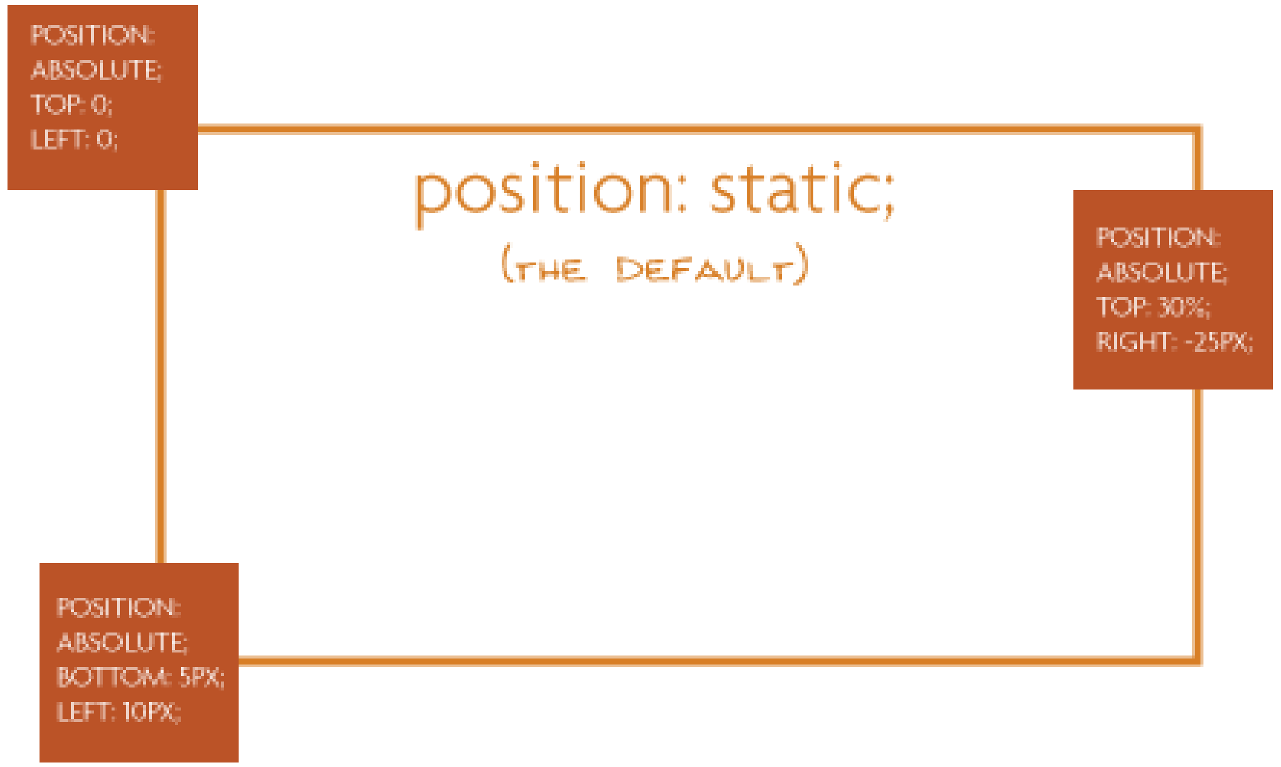
When an element has `position: absolute`, it is entirely removed from the normal flow of the page.

- That means its padding, margins, and borders no longer affect the elements around it

ABSOLUTE

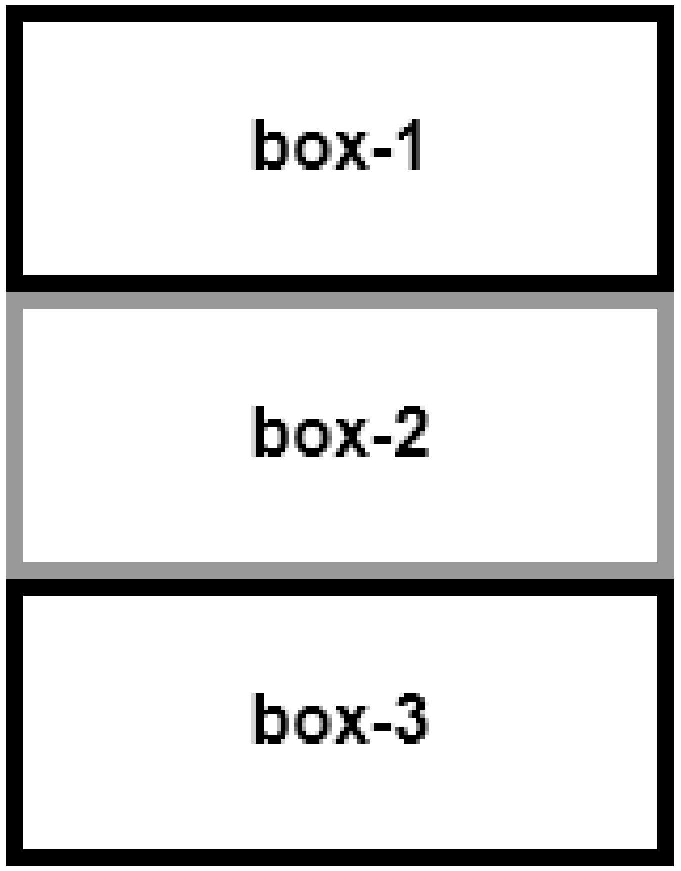
An element with `position: absolute` is absolutely positioned:

- To its closest parent with positioning
- Or if it has no parents with positioning (other than `static`), to the `body` of the page


POSITION: ABSOLUTE; TOP: 0; LEFT: 0;
position: static;
(THE DEFAULT)
POSITION: ABSOLUTE; TOP: 30%; RIGHT: -25PX;
POSITION: ABSOLUTE; BOTTOM: 5PX; LEFT: 10PX;

DEFAULT POSITIONING

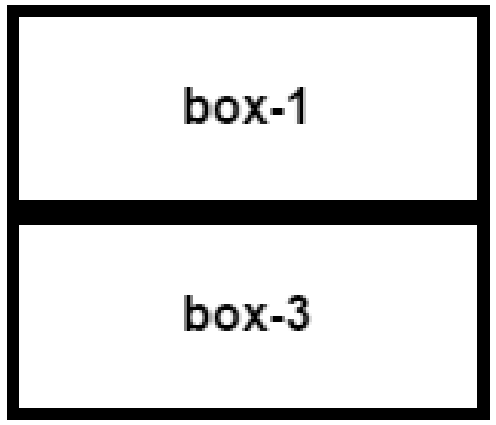
Remember that by default, block elements full up their entire row, and push any content to the next line, like this:



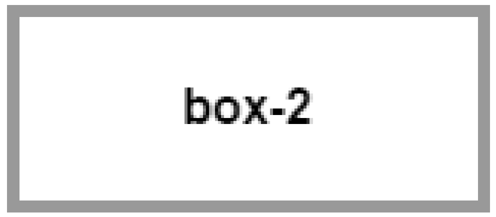
ABSOLUTE

If we give box-2 absolute positioning:

```
#box-2 {
    position: absolute;
    right: 0px;
    bottom: 0px;
}
```

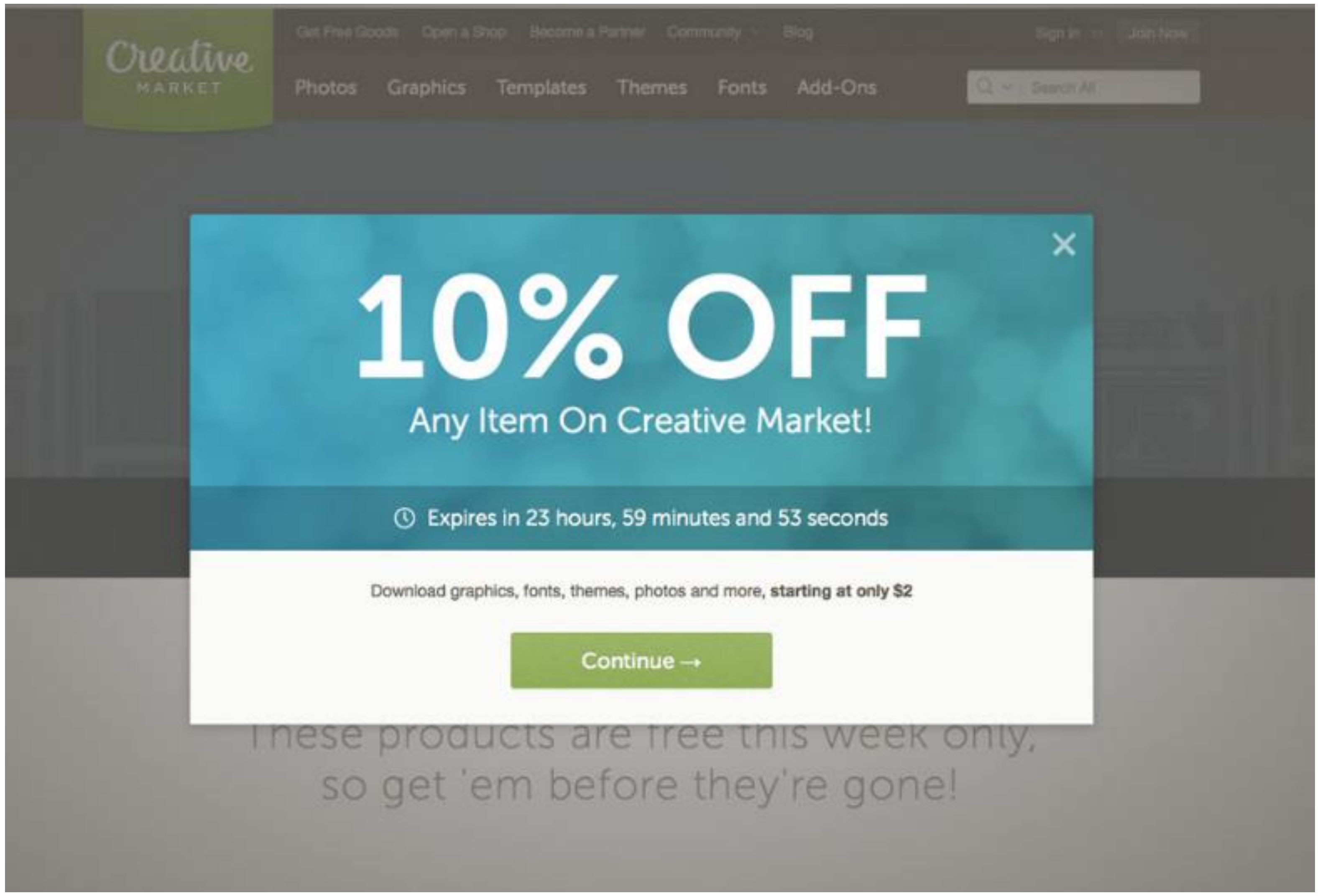


- box-2 is moved to the bottom right of the page
- box-3 moves up to occupy the space vacated by box-2



ABSOLUTE POWER

`position: absolute` is commonly used when creating page modals that pop up over other content



10% OFF

Any Item On Creative Market!

Expires in 23 hours, 59 minutes and 53 seconds

Download graphics, fonts, themes, photos and more, **starting at only $2**

These products are free this week only, so get 'em before they're gone!

GET YOUR Z'S

When you start changing how elements are positioned, you may need to specify which ones should be "on top", if they begin to overlap.

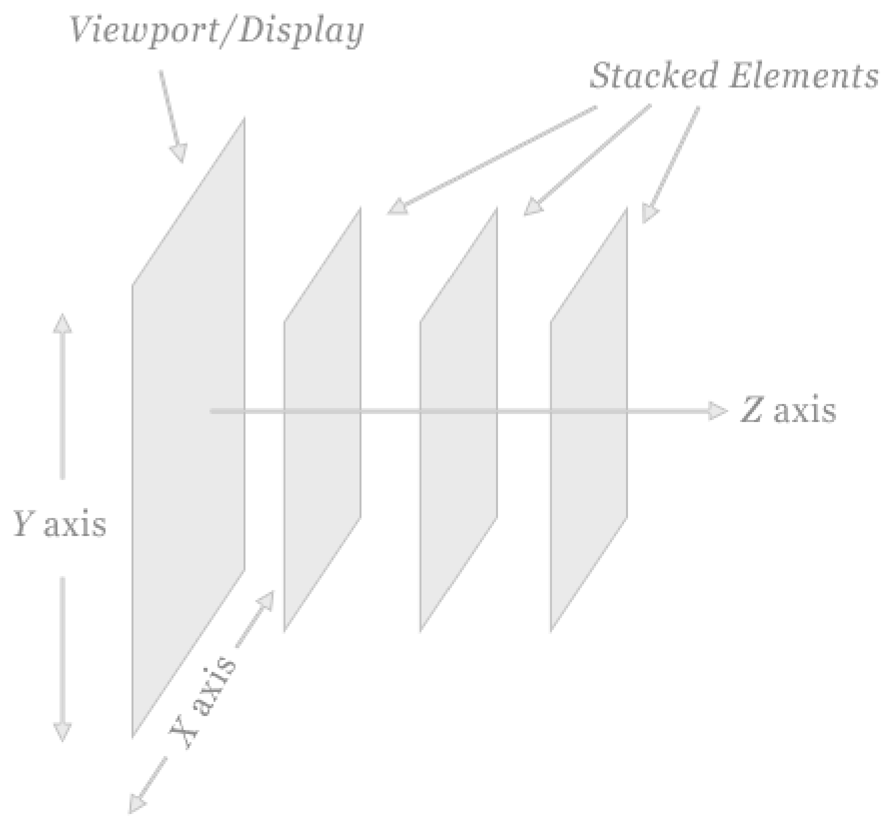
- Elements with no `position` declared will appear under elements that have absolute, fixed, or relative positioning

- By default, the *last* item on the webpage will appear on top of earlier elements that have the same type of `position`

GET YOUR Z'S

The CSS property `z-index` can be applied to force an element with position on top of any other by giving it a ***higher*** number

- By default all elements have `z-index: 0`
- Can assign negative or positive values to `z-index`



RELATIVE

`position: relative` allows you to move an element using directional attributes (`top`, `bottom`, etc).

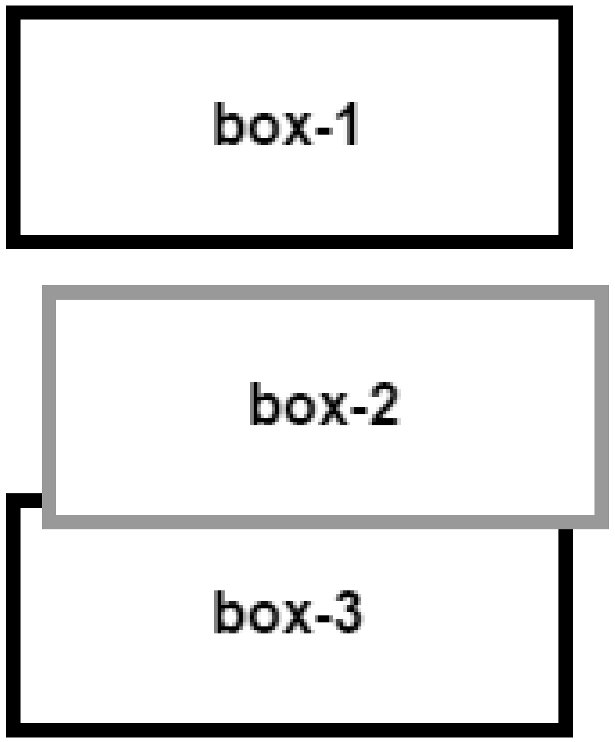
- If you set `position: relative` but no directional attributes, the element won't change at all

- If you set `top: 10px;` then it will be shifted 10 pixels down from where it would normally be

RELATIVE

box-2 has the following style, which moves it down and right:

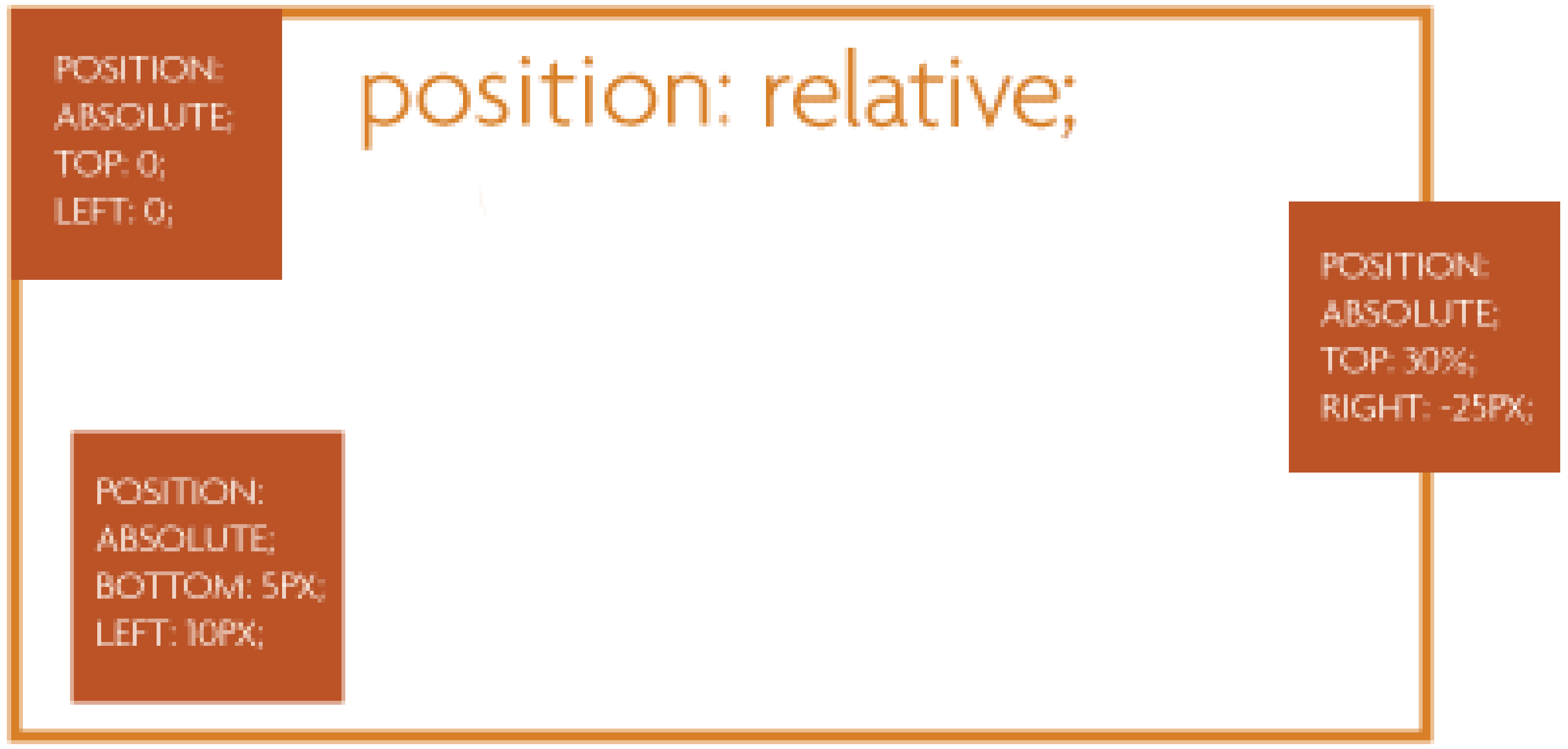
```
#box-2 {
    position: relative;
    left: 10px;
    top: 10px;
}
```



The other boxes behave like box-2 was in its original position (unlike using negative margins).

RELATIVE

More commonly, `position: relative` is used to position other absolutely-positioned elements inside the container


position: relative;
POSITION: ABSOLUTE; TOP: 0; LEFT: 0;
POSITION: ABSOLUTE; TOP: 30%; RIGHT: -25PX;
POSITION: ABSOLUTE; BOTTOM: 5PX; LEFT: 10PX;

PRACTICE TIME!

ASSIGNMENT

Create a new page using this page as a template:

http://kweeket.github.io/dev-101/demos/modal.html

Write CSS so that the modal (the `div` with the class `modal`) appears over the boilerplate text

- Apply box model properties to the content until it looks nice

Hint: Recall that you target a class in CSS like this:

```
.modal
```

RESPONSIVE DESIGN

MOBILE FIRST

An important principle of responsive design is "Mobile First"

- Both design and code should default to mobile resolution, adding progressive enhancements as the screen gets larger

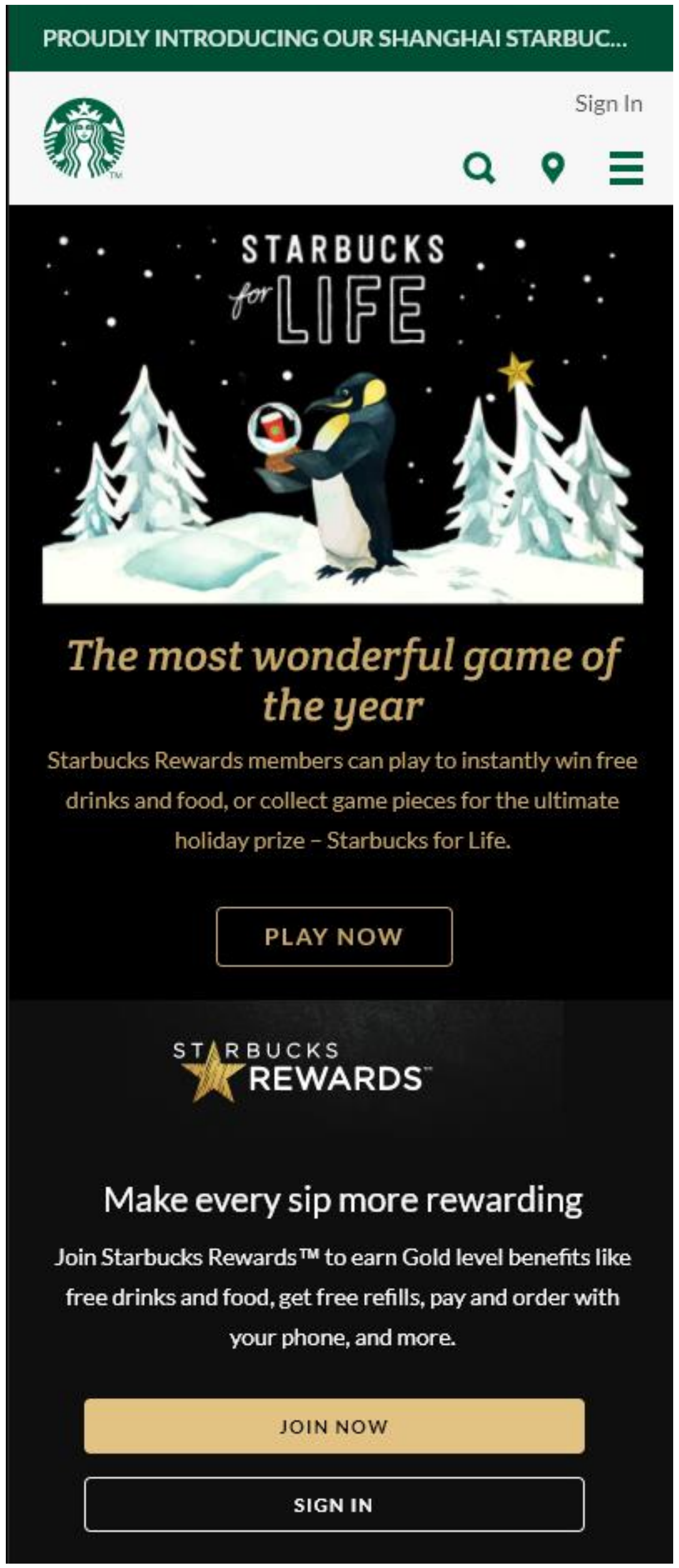
- One benefit of considering mobile first is that it trims down website content to its most vital elements
    - Mobile first = content first

RESPONSIVE != ADAPTIVE

Responsive design means that your design (and code) needs to function on a continuum of devices and screen sizes

- Although you should make care that things look great at specific “breakpoints” (640px wide for iPhone 4/5, 768px for iPad), it’s just as important to make sure things look good at *any* resolution
  - Users resize their browser windows
  - Technology changes

EXAMPLE – MOBILE



PROUDLY INTRODUCING OUR SHANGHAI STARBUC...

Sign In

STARBUCKS for LIFE

The most wonderful game of the year

Starbucks Rewards members can play to instantly win free drinks and food, or collect game pieces for the ultimate holiday prize – Starbucks for Life.

PLAY NOW

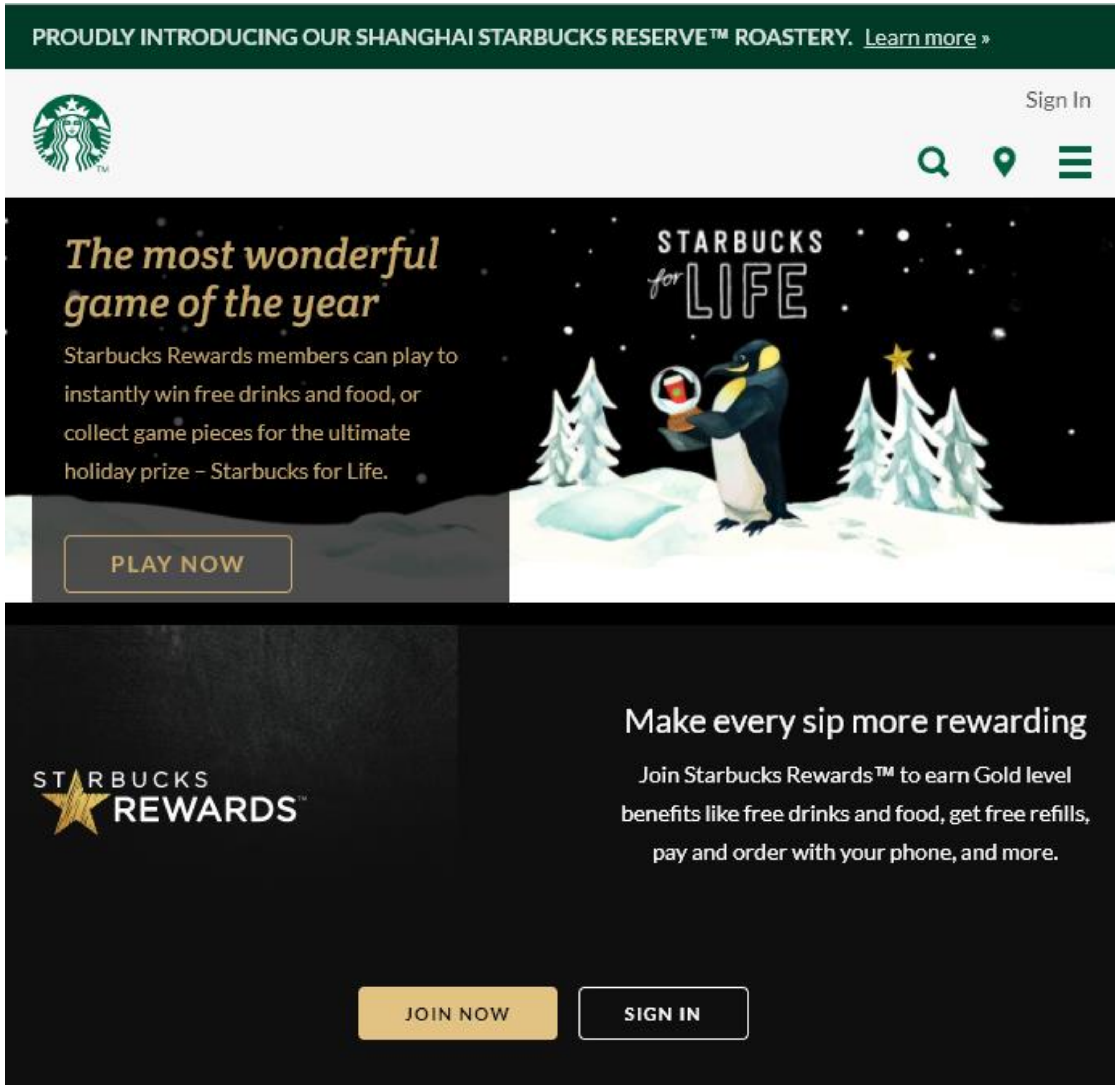
STARBUCKS REWARDS

Make every sip more rewarding

Join Starbucks Rewards™ to earn Gold level benefits like free drinks and food, get free refills, pay and order with your phone, and more.

JOIN NOW

SIGN IN



PROUDLY INTRODUCING OUR SHANGHAI STARBUCKS RESERVE™ ROASTERY. Learn more »

Sign In

The most wonderful game of the year

Starbucks Rewards members can play to instantly win free drinks and food, or collect game pieces for the ultimate holiday prize – Starbucks for Life.

PLAY NOW

STARBUCKS for LIFE

STARBUCKS REWARDS

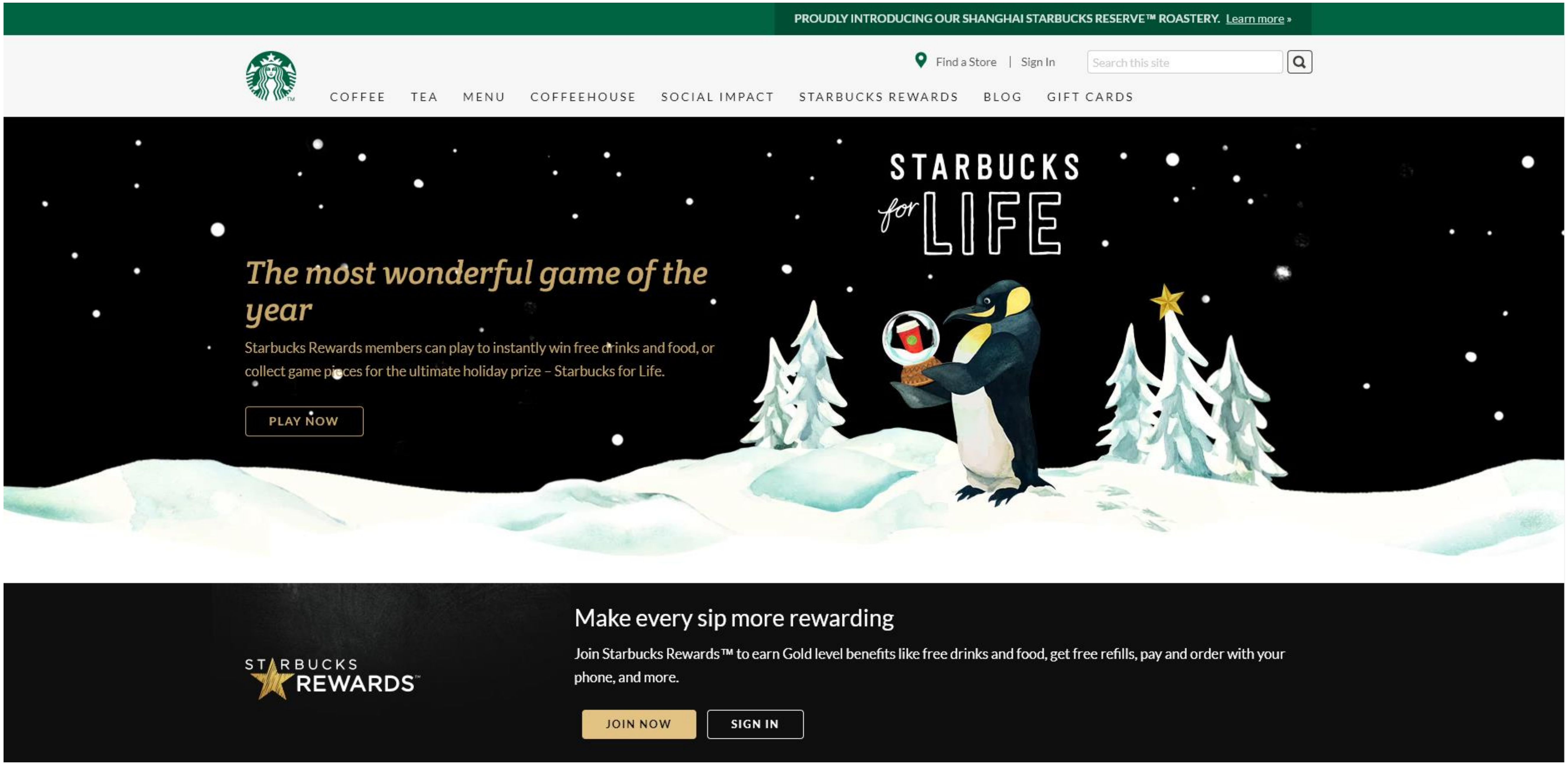
Make every sip more rewarding

Join Starbucks Rewards™ to earn Gold level benefits like free drinks and food, get free refills, pay and order with your phone, and more.

JOIN NOW

SIGN IN

EXAMPLE – DESKTOP



PROUDLY INTRODUCING OUR SHANGHAI STARBUCKS RESERVE™ ROASTERY. Learn more »

Find a Store | Sign In

Search this site

COFFEE TEA MENU COFFEEHOUSE SOCIAL IMPACT STARBUCKS REWARDS BLOG GIFT CARDS

STARBUCKS for LIFE

The most wonderful game of the year

Starbucks Rewards members can play to instantly win free drinks and food, or collect game pieces for the ultimate holiday prize – Starbucks for Life.

PLAY NOW

STARBUCKS REWARDS™

Make every sip more rewarding

Join Starbucks Rewards™ to earn Gold level benefits like free drinks and food, get free refills, pay and order with your phone, and more.

JOIN NOW SIGN IN

MEDIA QUERIES

Media queries are used to apply different CSS to different devices.

Some things you can use a media query to detect:

- The minimum or maximum screen height or width
- Whether the screen is rotated (in “landscape view”)
- If the page is being printed
- If the user is on a touch screen device
- The screen’s resolution

MEDIA QUERIES

Media queries have a different format than any other CSS we've seen so far

They always start with `@media` and have curly braces that contain all the CSS that applies to that media query rule

Multiple rules can be tested for, separated by `and`

```
@media (max-width: 480px) and (orientation: landscape) {
}
```

MEDIA QUERIES

To specify different styles when a webpage is being printed, use this media query:

```
@media print {}
```

This allows you to format your page so that it looks better on paper

- Remove dark colored backgrounds
- Make the page full screen
- Remove non-essential page elements (such as navigation links)

MEDIA QUERIES

Most modern “mobile first” websites have CSS that applies to phone-sized screens first

Then, anything specific to bigger screens goes in media queries that test for a minimum screen width, like this:

```
@media (min-width: 768px) {
}
```

MEDIA QUERIES

```css
/*   CSS that is used for phones, and also
     applies generally
*/
@media (min-width: 768px) {
     /* CSS that is different for tablets */
}

@media (min-width: 920px) {
     /* CSS that is different for desktops */
}
```

MEDIA QUERIES - EXAMPLE

```css
h1 {
     font-size: 12px;
     color: gray;
}


@media (min-width: 768px) {

     h1 { font-size: 20px; }

}


@media (min-width: 920px) {

      h1 { font-size: 25px; }

}
```

PRACTICE TIME!

ASSIGNMENT

Apply a media query to an element on your page so that it looks different when you resize your browser screen

Things to consider doing:

- Float a section right only in desktop
- Give buttons or links in navigation a bigger "click target" in mobile

FUN CSS TRICKS

FILTER

The `filter` property can be used to apply graphical effects similar to Photoshop filters on images, backgrounds, or borders.

Some options include blur, grayscale, brightness, saturate, and sepia.

Its value is one of the above options, followed by parenthesis indicating how much to apply the effect, like this:

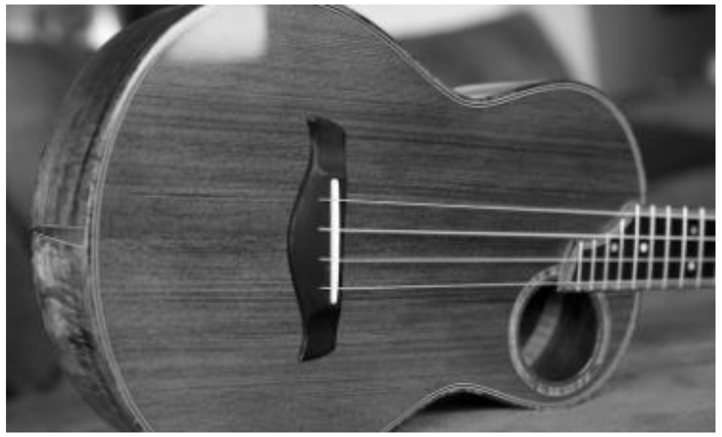
```
img { filter: grayscale(1); }
```

FILTER

Original image

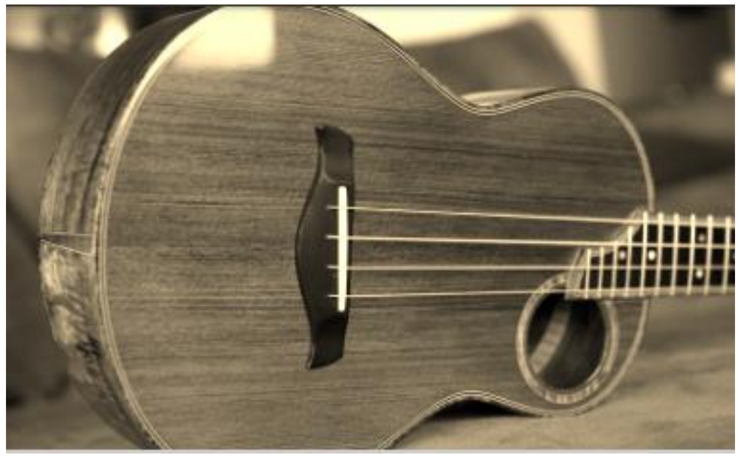
`filter: grayscale(1);`

Converts the image to gray. 1 is fully gray, 0 is original. Any value between 0 and 1 is allowed



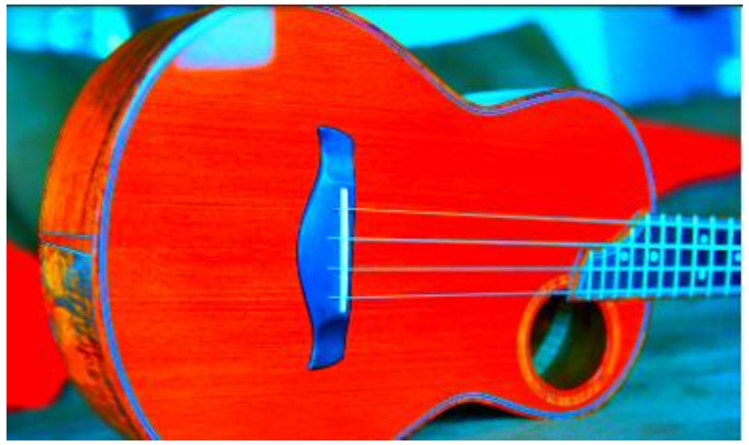
`filter: sepia(1);`

Converts the image to sepia. 1 is fully sepia, 0 is original. Any value between 0 and 1 is allowed

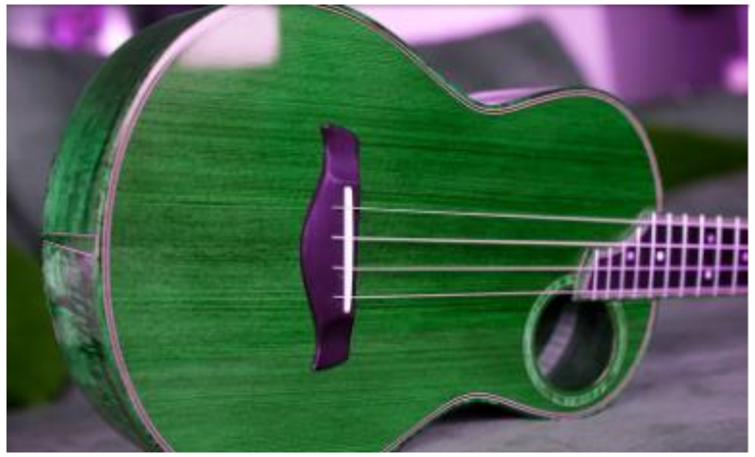


FILTER

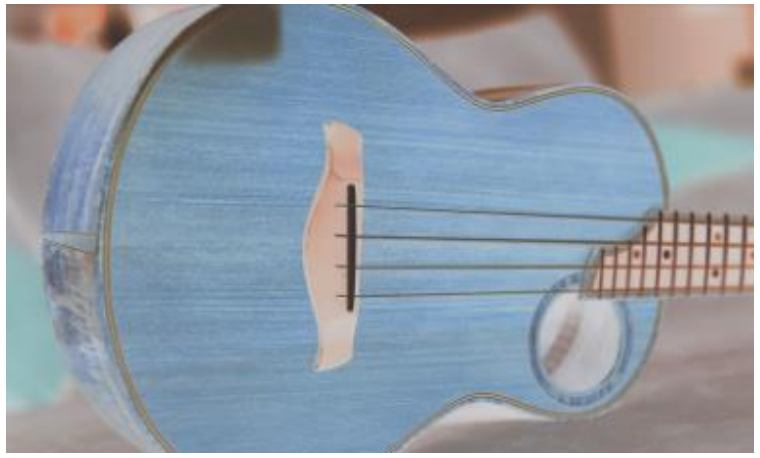
`filter: saturate(8);`
Larger values are more saturated. Anything under 1 will make the element less saturated than the original



`filter: hue-rotate(90deg);`
Must be an angle of rotation in degrees, that will affect how far around the color circle the input is adjusted



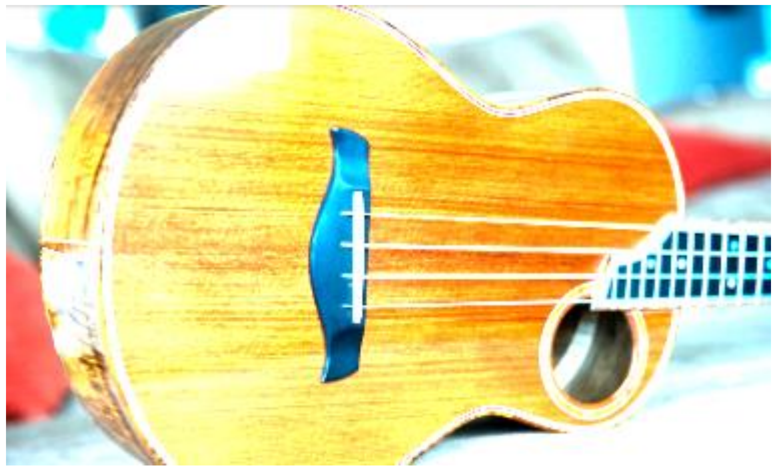
`filter: invert(.8);`
Must be a percentage. Flips RGB values by that amount



FILTER

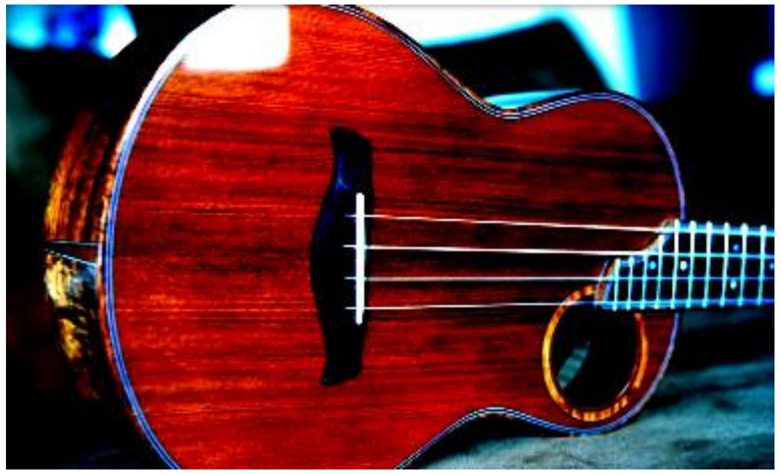
`filter: brightness(3);`

Larger values are brighter. Anything under 1 will make the element darker



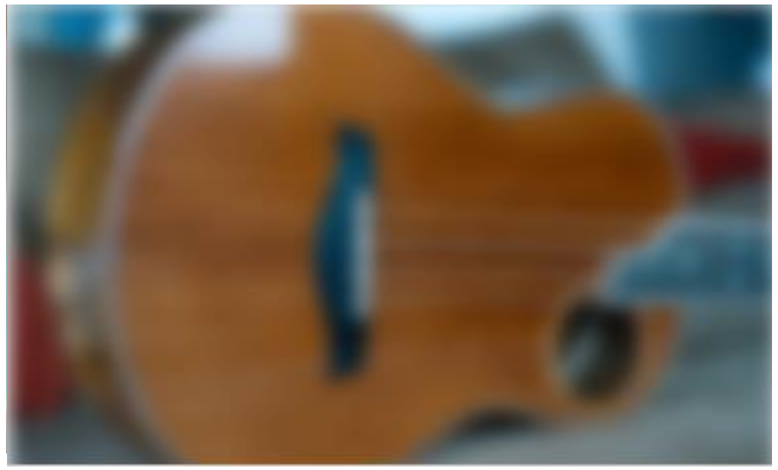
`filter: contrast(4);`

Larger values apply greater contrast. Anything under 1 will apply less contrast than the original



`filter: blur(5px);`

Applies a Gaussian blur. The px value is how many pixels will blend into each other



ANIMATE

To allow elements on your page to animate using CSS, use the `transition` property

- By default, an element that transforms will change abruptly – for example, when you changed the text color of links using `:hover`

- `transition` makes those changes occur smoothly over time, instead of suddenly

ANIMATE

```
nav { transition: all 1s; }
```

This means, animate all CSS properties that happen to `nav` for 1 second

- The first value: *which* CSS properties can be animated
- The second value: *how long* the animation should take to finish
- Separate the values with spaces

ANIMATE

Until there is a CSS property that changes, `transition` won't have any noticeable effect

Remember that you can give any element a different style when the user hovers on it

- Use the CSS rule `:hover`
- This will allow us to see the animation effect as the style changes from one value to another

FILTER ON HOVER

```
img {
   filter: grayscale(0);
   transition: all 1s;
}

img:hover {
   filter: grayscale(1);
}
```

Note that a `grayscale` of `0` (ie, none) is applied to the "normal" image. This allows the `transition` to run both when the user hovers *and* when they move their mouse away

TRANSFORMATION

The `transform` property lets you manipulate an element by skewing, rotating, moving, or scaling

Like `filter`, the value is the type of transformation you want to apply, with the degree of transformation inside parentheses

```
.bigger {
  transform: scale(20);
}
```

TRANSFORMATION

`transform` options include:

`scale` – changes the size of an element

`skew` – tilts the element

`rotate` – rotates the element a specified number of degrees

`translate` – moves an element

TRANSFORMATION

Potential uses for `transform`:

- Flip an arrow when sorting or expanding a menu

```
.toggle {
      transform: rotateZ(0deg);
      transition: all .11s;
}

.toggle:hover {
      transform: rotateZ(180deg);
}
```

TRANSFORMATION

- “Lift” a card when the user interacts with it

```
.card {
      transform: scale(1);
      transition: all .15s ease-in;
}

. card:hover {
       transform: scale(1.01);
}
```

PRACTICE TIME!

PLAY WITH ANIMATION

Apply `transition` to at least one element on your page

- Give that element a different style on hover so that you can see the animation occur

Apply `filter` or `transform` to at least one element on your page

- Play with all the possibilities!

Reference https://robots.thoughtbot.com/transitions-and-transforms to see more options for `transform`

“THE END”

END TIMES

I will be keeping all slides and demos up on the class site indefinitely

You can continue to ask me questions anytime at beckjohnson@gmail.com

THATS ALL FOLKS!

Please provide feedback for this class!

http://svcseattle.com/evaluation