CS193X: Web Programming Fundamentals

Spring 2017

Victoria Kirst
(vrk@stanford.edu)

Today's schedule

Today:

- Keyboard events
- Mobile events
- Simple CSS animations
- Victoria's office hours once again moved to Friday...

Friday

- Classes and objects in JavaScript
- `this` keyword and `bind`
- **HW2 due**
- **HW3 assigned**
- Victoria has office hours 2:30 - 4pm

Other JavaScript events?

We've been doing a ton of JavaScript examples that involve `click` events...

Aren't there other types of events?

Other JavaScript events?

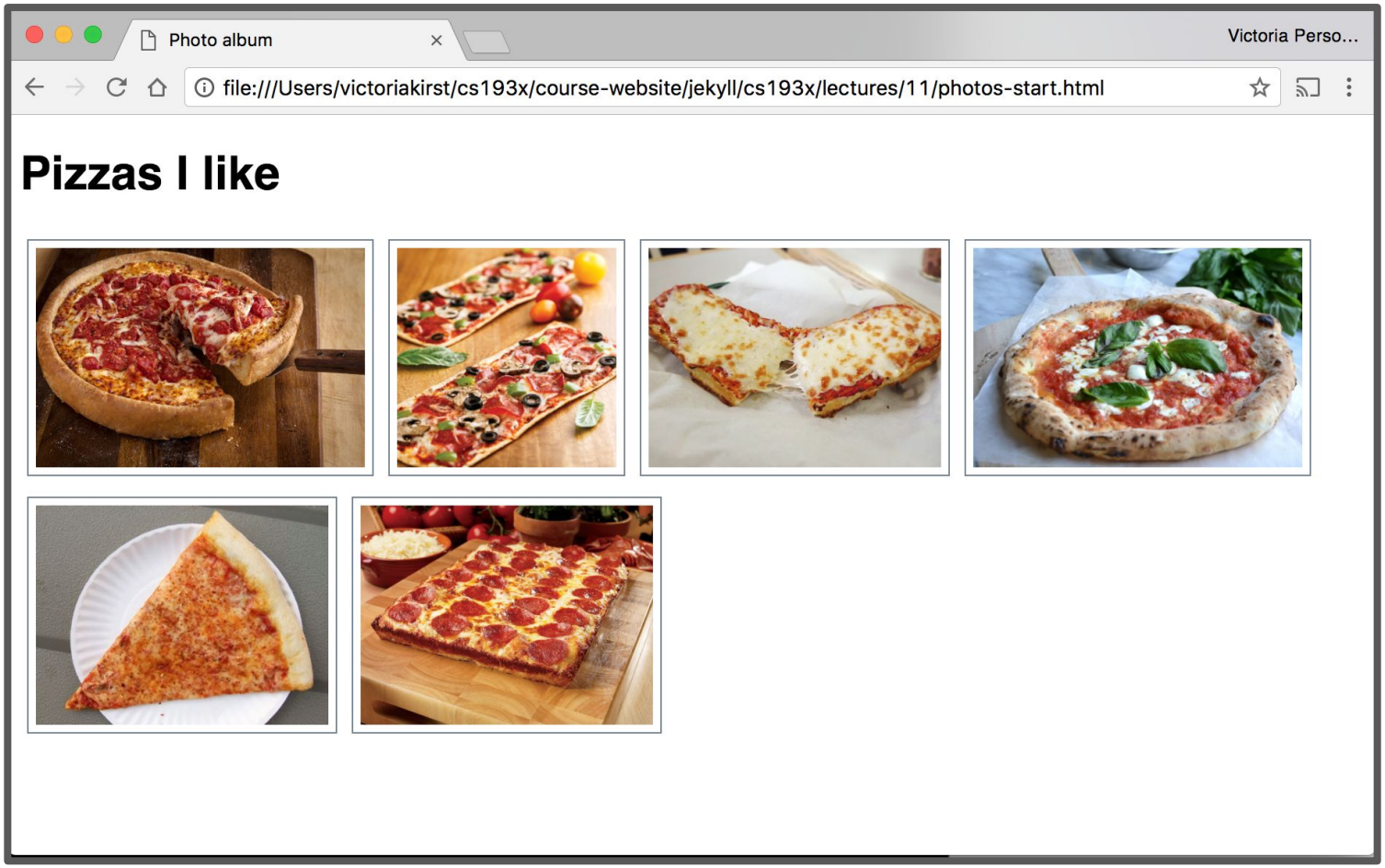
We've been doing a ton of JavaScript examples that involve `click` events...

Aren't there other types of events?

- Of course!
- Today we'll talk about:
  - **Keyboard events**
  - **Pointer / mobile events**
  - (possibly) **Animation events**

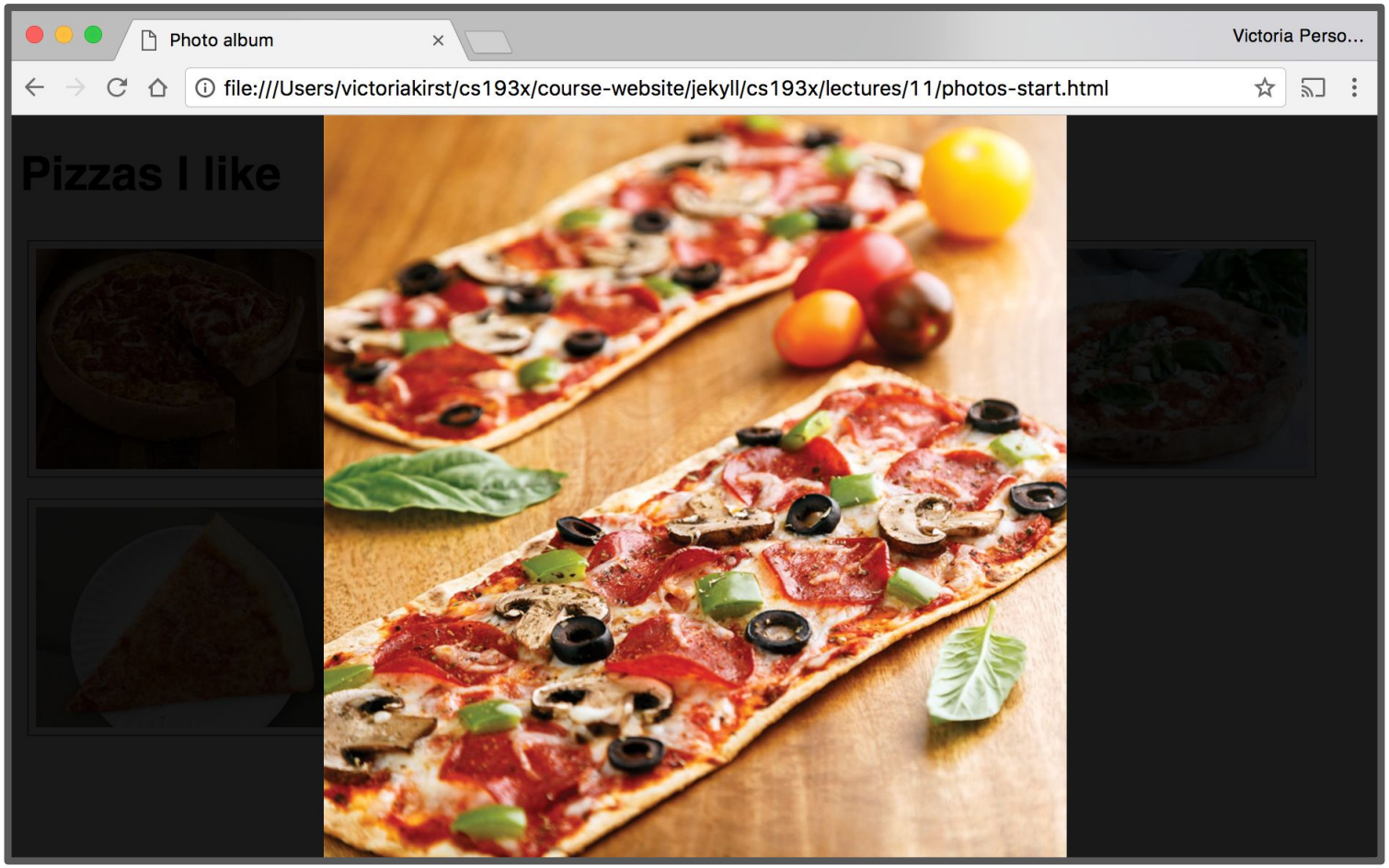
Example: Photo Album

We're going to add a few features to this photo album:



Example: Photo Album

We're going to add a few features to this photo album:



Code walkthrough:
photo-start.html
photo.js
photo.css

General setup

```
<body>
  <h1>Pizzas I like</h1>
  <section id="album-view">
  </section>

  <section id="modal-view" class="hidden">
  </section>
</body>
```

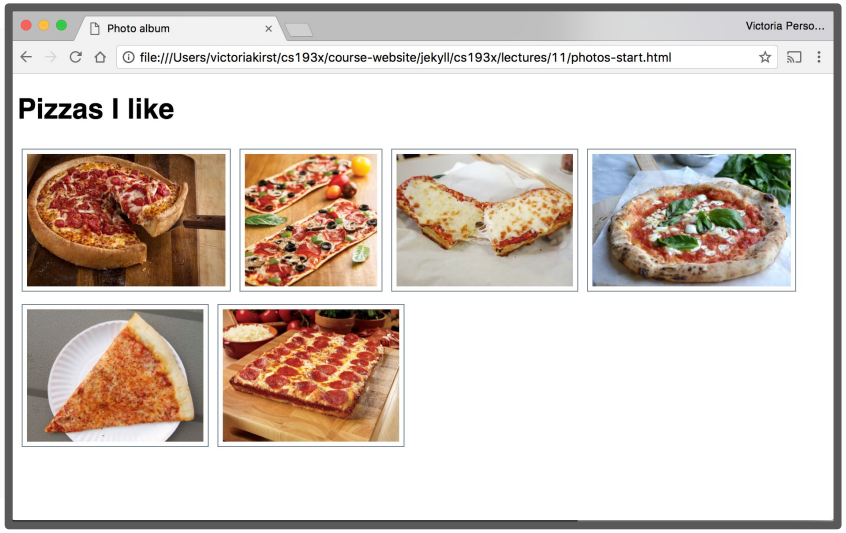
photo.html contains both "screens":

- The album view: Thumbnails of every photo
- The "modal" view: A single photo against a semi-transparent black background
  - Hidden by default

CSS: Album

photo.css: The album view CSS is pretty straightforward:

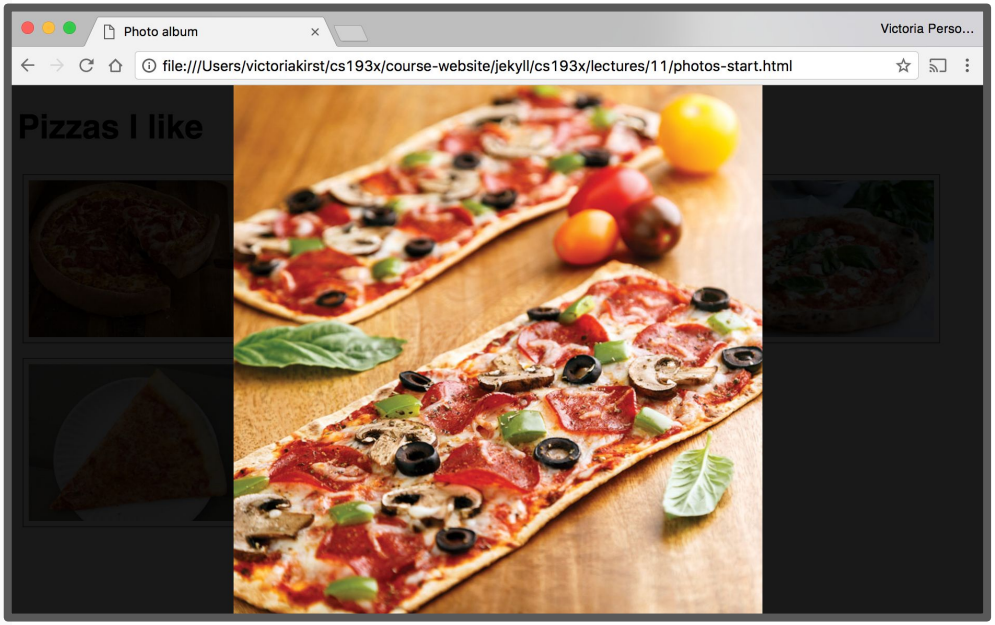
```
#album-view img {
  border: 1px solid slategray;
  margin: 5px;
  padding: 5px;
  height: 150px;
}
```



CSS: Modal

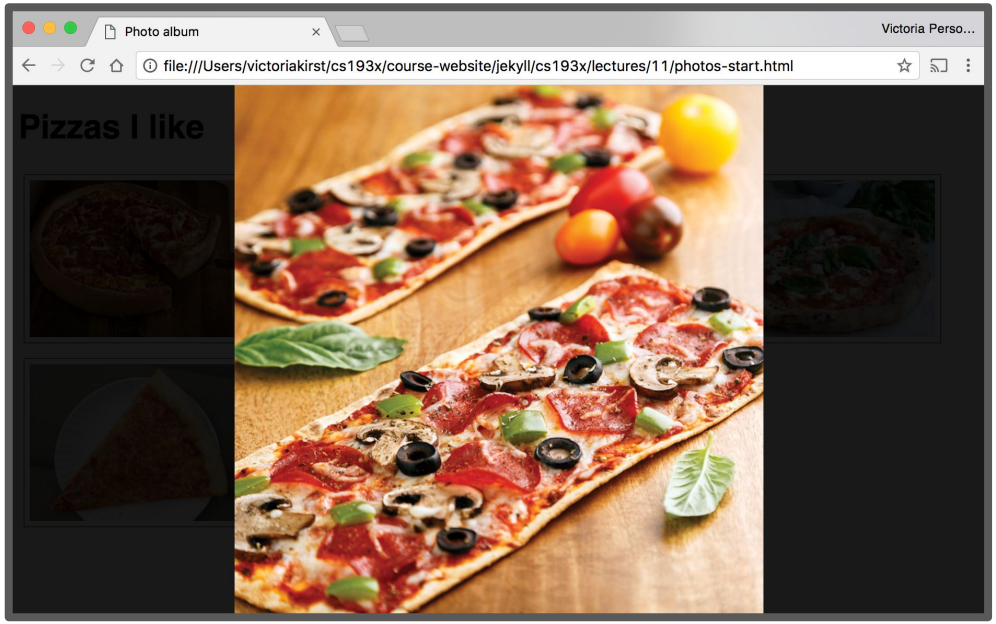
Modal view is a little more involved, but all stuff we've learned:

```
#modal-view {
  position: absolute;
  top: 0;
  left: 0;
  height: 100vh;
  width: 100vw;

  background-color: rgba(0, 0, 0, 0.9);
  z-index: 2;

  display: flex;
  justify-content: center;
  align-items: center;
}
```



CSS: Modal image

```
#modal-view img {
   max-height: 100%;
   max-width: 100%;
}
```

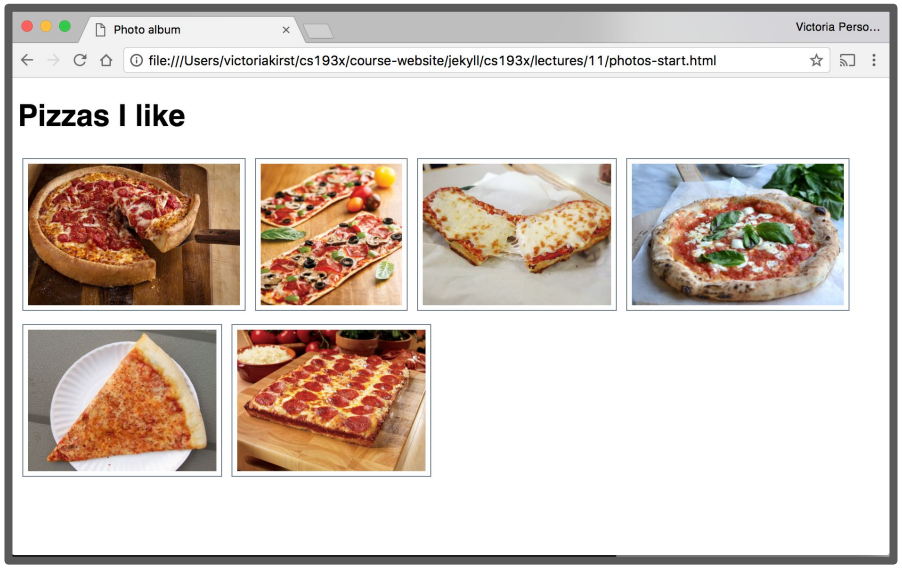


- Image sizes are constrained to the height and width of the parent, `#modal-view` (whose height and width are set to the size of the viewport)

CSS: Hidden modal

```
<body>
  <h1>Pizzas I like</h1>
  <section id="album-view">
  </section>

  <section id="modal-view" class="hidden">
  </section>
</body>
```

```
#modal-view.hidden {
  display: none;
}
```



Even though both the album view and modal view are in the HTML, the model view is set to `display: none;` so it does not show up.

Global List of Photos

```
<head>
  <meta charset="utf-8">
  <title>Photo album</title>
  <link rel="stylesheet" href="css/photo.css">
  <script src="js/photo-list.js" defer></script>
  <script src="js/photo.js" defer></script>
</head>
```

```
const PHOTO_LIST = [
  'images/deepdish.jpg',
  'images/flatbread.jpg',
  'images/frenchbread.jpg',
  'images/neapolitan.jpg',
  'images/nypizza.jpg',
  'images/squarepan.jpg'
];
```

photo-list.js: There is a global array with the list of string photo sources called PHOTO_LIST.

Photo thumbnails

```
function createImage(src) {
  const image = document.createElement('img');
  image.src = src;
  return image;
}
```

```
const albumView = document.querySelector('#album-view');
for (let i = 0; i < PHOTO_LIST.length; i++) {
  const photoSrc = PHOTO_LIST[i];
  const image = createImage(photoSrc);
  image.addEventListener('click', onThumbnailClick);
  albumView.appendChild(image);
}
```

photo.js: We populate the initial album view by looping over `PHOTO_LIST` and appending `<img>`s to the `#album-view`.

Clicking a photo

```
function onThumbnailClick(event) {
  const image = createImage(event.currentTarget.src);
  modalView.appendChild(image);
  modalView.classList.remove('hidden');
}
```

When the user clicks a thumbnail:

- We create another `<img>` tag with the same src
- We append this new `<img>` to the `#modal-view`
- We unhide the `#modal-view`

Closing the modal dialog

```
function onModalClick() {
  modalView.classList.add('hidden');
  modalView.innerHTML = '';
}
```

```
const modalView = document.querySelector('#modal-view');
modalView.addEventListener('click', onModalClick);
```

When the user clicks the modal view:

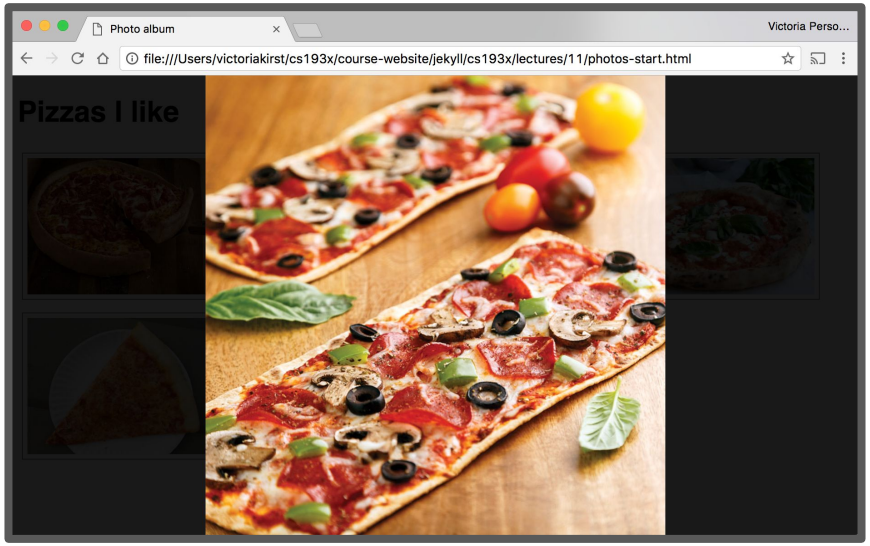
- We hide the modal view again
- We clear the image we appended to it by setting `innerHTML = '';`

Keyboard navigation

Navigating photos

Let's add some keyboard events to navigate between photos in the Modal View:

- Left arrow: Show the "i - 1"th picture
- Right arrow: Show the "i + 1"th picture
- Escape key: Close dialog



Keyboard events

Event name	Description
`keydown`	Fires when any key is pressed. Continues firing if you hold down the key. (mdn)
`keypress`	Fires when any **character** key is pressed, such as a letter or number. Continues firing if you hold down the key. (mdn)
`keyup`	Fires when you stop pressing a key. (mdn)

You can listen for keyboard events by adding the event listener to `document`:

```
document.addEventListener('keyup', onKeyUp);
```

KeyboardEvent.key

```
function onKeyUp(event) {
  console.log('onKeyUp:' + event.key);
}
document.addEventListener('keyup', onKeyUp);
```

Functions listening to a key-related event receive a parameter of `KeyboardEvent` type.

The `KeyboardEvent` object has a `key` property, which stores the string value of the key, such as `"Escape"`

- List of key values

Useful key values

Key string value	Description
`"Escape"`	The Escape key
`"ArrowRight"`	The right arrow key
`"ArrowLeft"`	The left arrow key

Example: key-events.html

Let's finish the feature!

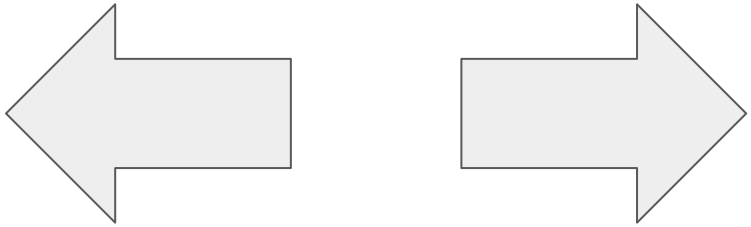
Finished result:
photo-desktop-finished.html

Mobile?

Keyboard events work well on desktop, but keyboard navigation doesn't work well for mobile.

On your phone, you can usually navigate photo albums using **gestures**:

- **Left swipe** reveals the next photo
- **Right swipe** reveals the previous photo



Next Previous

Mobile?

Keyboard events work well on desktop, but keyboard navigation doesn't work well for mobile.

On your phone, you can usually navigate photo albums using **gestures**:

- **Left swipe** reveals the next photo
- **Right swipe** reveals the previous photo

How do we implement the swipe gesture on the web?

Custom swipe events

- There are no gesture events in JavaScript (yet).
- That means there is no "Left Swipe" or "Right Swipe" event we can listen to. (Note that `drag` does not do what we want, nor does it work on mobile)

To get this behavior, we must implement it ourselves.

To do this, it's helpful to learn about a few more JS events:

- MouseEvent
- TouchEvent
- PointerEvent

MouseEvent

Event name	Description
`click`	Fired when you click and release (mdn)
`mousedown`	Fired when you click down (mdn)
`mouseup`	Fired when when you release from clicking (mdn)
`mousemove`	Fired repeatedly as your mouse moves (mdn)

*__`mousemove`__ only works on desktop, since there's no concept of a mouse on mobile.

TouchEvent

Event name	Description
`touchstart`	Fired when you touch the screen (mdn)
`touchend`	Fired when you lift your finger off the screen (mdn)
`touchmove`	Fired repeatedly while you drag your finger on the screen (mdn)
`touchcancel`	Fired when a touch point is "disrupted" (e.g. if the browser isn't totally sure what happened) (mdn)

*__`touchmove`__ only works on mobile (example)

clientX and clientY

```
function onClick(event) {
  console.log('x' + event.clientX);
  console.log('y' + event.clientY);
}
element.addEventListener('click', onClick);
```

MouseEvents have a `clientX` and `clientY` :

- `clientX`: x-axis position relative to the left edge of the browser viewport
- `clientY`: y-axis position relative to the top edge of the browser viewport

Implementing drag

When a user clicks down/touches an element…

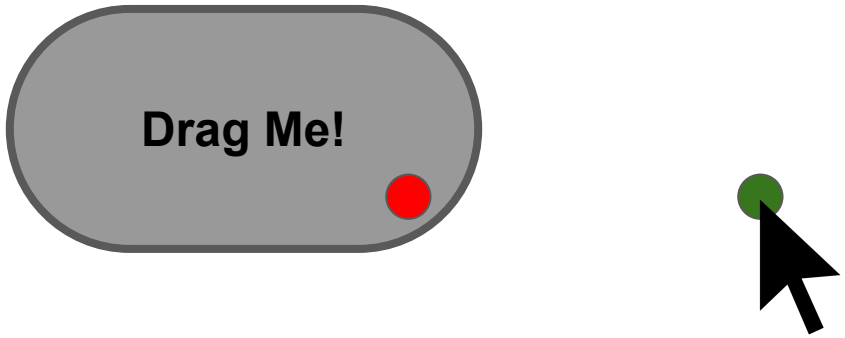
Implementing drag

```
originX = 100;
```

Take note of the starting position.

Implementing drag

```
originX = 100;
newX = 150;
```



Then on `mousemove` / `touchmove`, make note of the new mouse position

Implementing drag

```
originX = 100;
newX = 150;
```

Move the element by the difference between the old and new positions.

Implementing drag

Then on release...

Implementing drag

… stop listening to `mousemove` / `touchmove`.

Dragging on mobile and desktop

Wouldn't it be nice if we didn't have to listen to different events for mobile and desktop?

PointerEvent

`PointerEvent`: "pointer" events that work the same with for both mouse and touch

- Not to be confused with `pointer-events` CSS property (completely unrelated)
- **Note:** In this case, Mozilla's documentation on `PointerEvent` is not great.
  - A Google blog post on `PointerEvent`

`PointerEvent` inherits from `MouseEvent`, and therefore has `clientX` and `clientY`

PointerEvent

Event name	Description
`pointerdown`	Fired when a "pointer becomes active" (touch screen or click mouse down) (mdn)
`pointerup`	Fired when a pointer is no longer active (mdn)
`pointermove`	Fired repeatedly while the pointer moves (mouse move or touch drag) (mdn)
`pointercancel`	Fired when a pointer is "interrupted" (mdn)

*__`pointermove`__ works on mobile and desktop!

… Except…

Our first controversial feature!

`PointerEvent` is **not** implemented on all browsers yet:

- Firefox implementation is in progress
- Safari outright opposes this API… since 2012.

Argh!!! Does this mean we can't use it?

Polyfill library

A polyfill library is code that implements support for browsers that do not natively implement a web API.

Luckily there is a polyfill library for `PointerEvent`: https://github.com/jquery/PEP

PointerEvent Polyfill

To use the PEP polyfill library, we add this `script` tag to our HTML:

```
<script src="https://code.jquery.com/pep/0.4.1/pep.js"></script>
```

And we'll add need to add `touch-action="none"` to the area where we want `PointerEvents` to be recognized*:

```
<section id="photo-view" class="hidden" touch-action="none">
</section>
```

*Technically what this is doing is it is telling the browser that we do not want the default touch behavior for children of this element, i.e. on a mobile phone, we don't want to recognize the usual "pinch to zoom" type of events because we will be intercepting them via PointerEvent. This is normally a CSS property, but the limitations of the polyfill library requires this to be an HTML attribute instead.

Moving an element

We are going to use the `transform` CSS property to move the element we are dragging from its original position:

```
originX = 100;
newX = 150;
delta = newX - originX;
```


```
element.style.transform = 'translateX(' + delta + 'px)';
```

transform

`transform` is a strange but powerful CSS property that allow you to translate, rotate, scale, or skew an element.

`transform: translate(`***x***`, `***y***`)`	Moves element relative to its natural position by ***x*** and ***y***
`transform: translateX(`***x***`)`	Moves element relative to its natural position horizontally by ***x***
`transform: translateY(`***y***`)`	Moves element relative to its natural position vertically by ***y***
`transform: rotate(`***deg***`)`	Rotates the element clockwise by ***deg***
`transform: rotate(10deg) translate(5px, 10px);`	Rotates an element 10 degrees clockwise, moves it 5px down, 10px right

Examples

`translate` vs `position`

Can't you use `relative` or `absolute` positioning to get the same effect as `translate`? What's the difference?

- `translate` is much faster
- `translate` is optimized for animations

See comparison (article):

- Absolute positioning (click "10 more macbooks")
- transform: translate (click "10 more macbooks")

Finally, let's code!

preventDefault()

On desktop, there's a default behavior for dragging an image, which we need to disable with `event.preventDefault()`:

```
function startDrag(event) {
  event.preventDefault();
```

`style` attribute

Every `HTMLElement` also has a `style` attribute that lets you set a style directly on the element:

```
element.style.transform =
        'translateX(' + value + ')';
```

Generally **you should not use the `style` property**, as adding and removing classes via `classList` is a better way to change the style of an element via JavaScript

But when we are dynamically calculating the value of a CSS property, we have to use the `style` attribute.

`style` attribute

The `style` attribute has **higher precedence** than any CSS property.

To undo a style set via the `style` attribute, you can set it to the empty string:

```
element.style.transform = '';
```

Now the element will be styled according to any rules in the CSS file(s).

CSS animations