Hamburg Coding School

Henning Muszynski

Session 3

JavaScript for Web

Feedback

What did we learn last time

Re-examination

Query Selectors

Selecting by id

```
<p id="first-text">Lorem Ipsum ...</p>

document.querySelector("#first-text")
```

Query Selectors

Selecting by class

```
<p class="text">Lorem Ipsum ...</p>

document.querySelector(".text")
```

Query Selectors

Selecting by tag

```
<p>Lorem Ipsum ...</p>

document.querySelector("p")
```

Query Selectors

Selecting by attributes

```
<p aria-hidden="true">Lorem Ipsum ...</p>

document.querySelector("[aria-hidden]")
document.querySelector('[aria-hidden="true"]')
```

Query Selectors

Selecting by order

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>

// Select Item 2 and Item 3
document.querySelector('li + li')
```

Query Selectors

Multiple selectors

```
<h1>Header</h1>
<h2>Sub Header</h2>

document.querySelector("h1, h2")
```

Query Selectors

How to use them

```
// Get the first match:
document.querySelector()

// Get all matches:
document.querySelectorAll()

// Match on children:
const navigation = document.querySelector("#nav")
navigation.querySelector(‘li')
```

Query Selectors

Shorthands

```
document.getElementById()
document.getElementByClassName()
document.getElementByTagName()

document.getElementsByClassName()
document.getElementsByTagName()
```

Query Selectors

Special accessors

```
// array of all children elements
element.childNodes

element.firstChild
element.lastChild
element.parentNode

// checks all parents for a match
element.closest('p')
```

NodeList vs. Array

```
const divs = document.querySelectorAll('div')
divs.forEach(div => {
    /* ... */
})

divs.map(div => { /* ... */ }) // throws exception

// turn NodeList into array first:
[...divs].map(div => { /* ... */ })
```

Manipulating the DOM

Creating new elements

```
const div = document.createElement('div')
div.innerHTML = '<p>Text</p>'
// or
div.innerText = 'Text'

document.createTextNode('Text')
```

Manipulating the DOM

Adding, removing and replacing elements

```
document.body.appendChild(div)
element.appendChild(div)

document.insertBefore(newElement, referenceElement)

document.replaceChild(newElement, oldElement)
document.removeChild(element)

// emptying element
element.innerHTML = '' // removes all children + listeners
```

Manipulating the DOM

Working with CSS classes

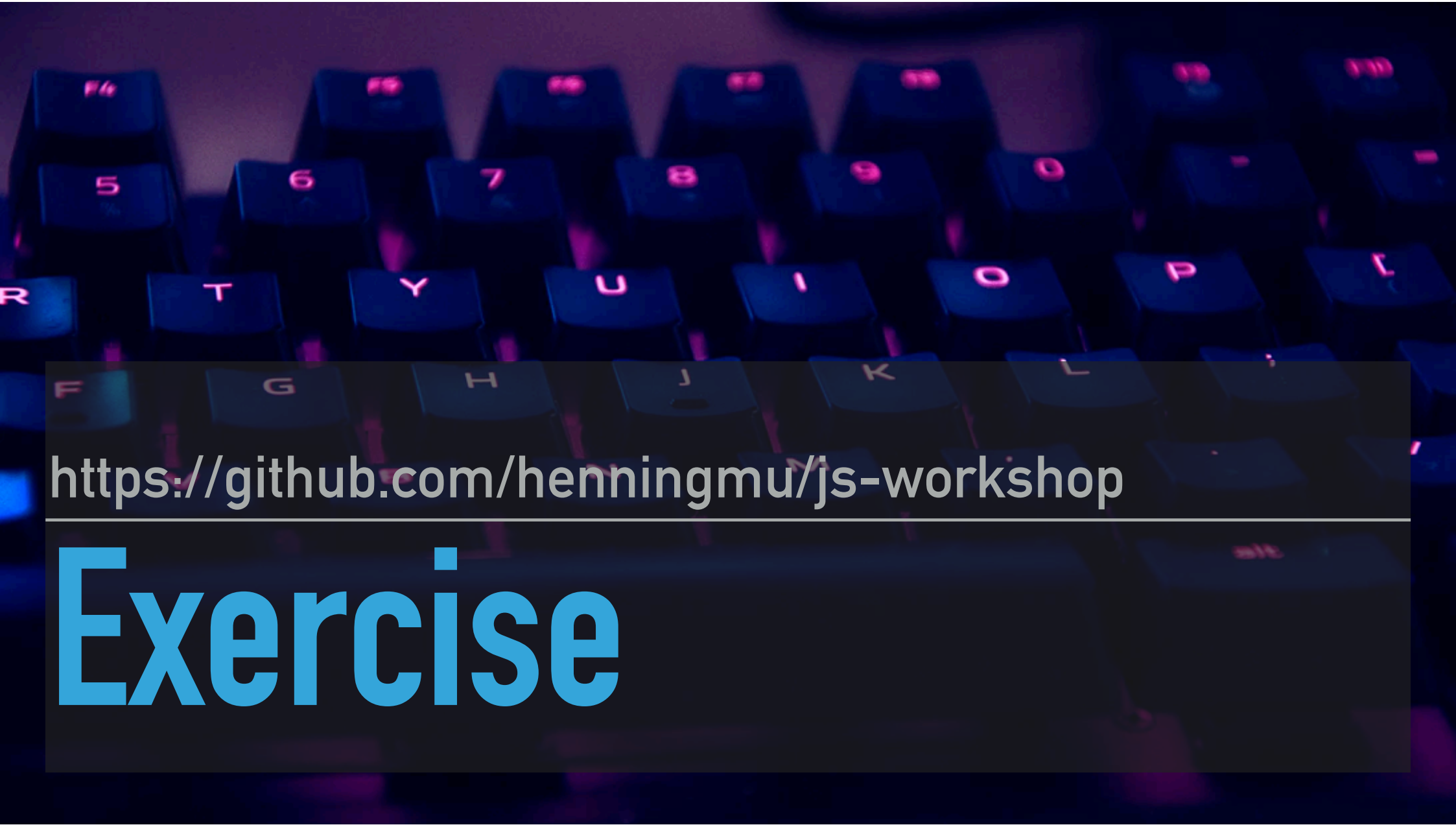
```
element.classList.add('cool-class')
element.classList.remove('bad-class')
element.classList.contains('another-class')
```

Manipulating the DOM

Adding event listeners

```
element.addEventListener('click', (event) => {
    /* ... */
})

element.addEventListener('change', (event) => {
    /* ... */
})
```



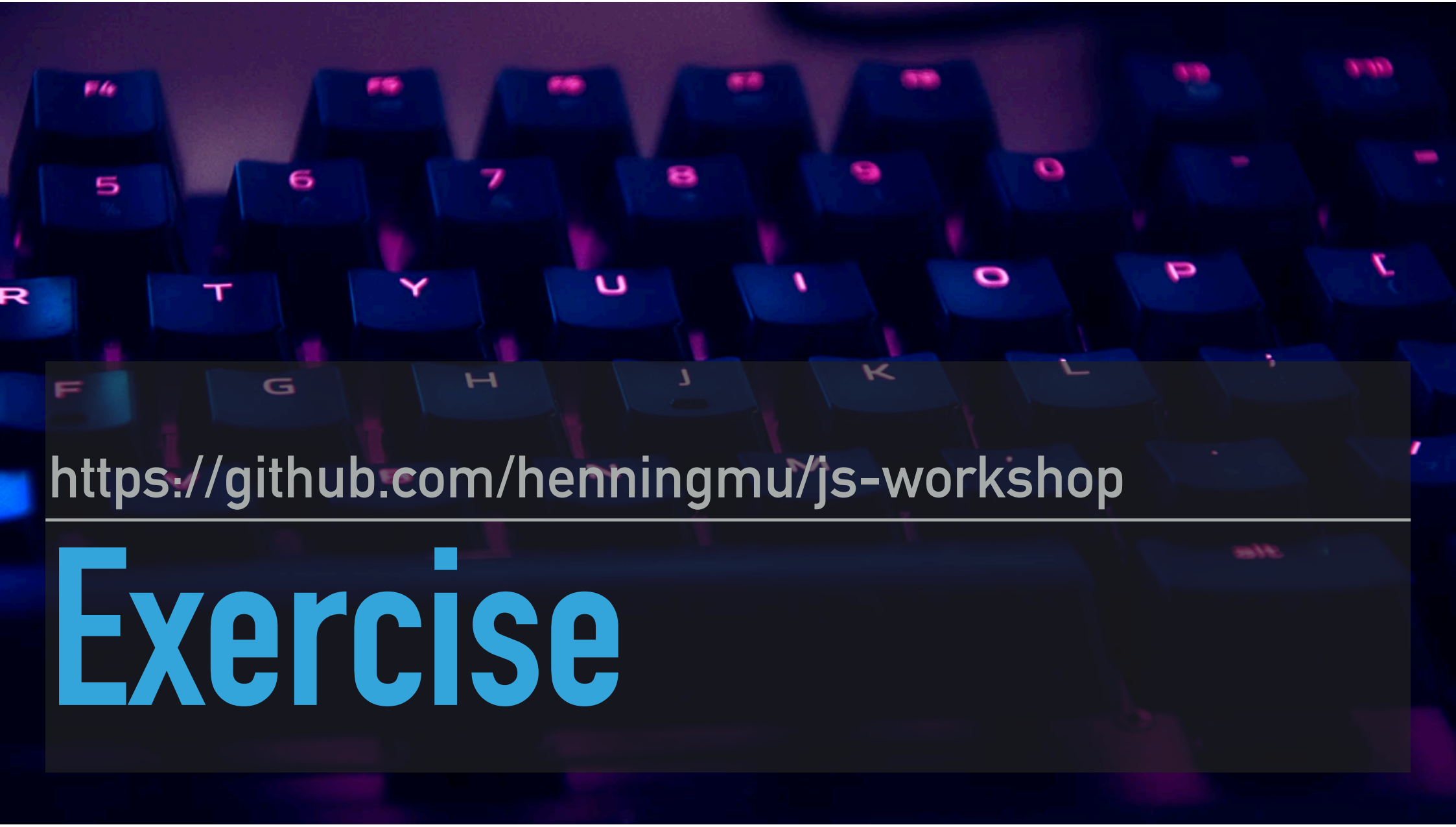
https://github.com/henningmu/js-workshop

Exercise

Load data with fetch

Interact with APIs – the modern way of making HTTP requests

```
fetch("https://api.punkapi.com/v2/beers")
    .then(response => response.json())
    .then(beers => {
        // Do something with the loaded data
    })
    .catch(error => {
        // Handle the error
    })
```



https://github.com/henningmu/js-workshop

Exercise

Store data in LocalStorage

- Synchronous Key-Value Storage
- Only Strings as values

```
localStorage.setItem('key', value)

localStorage.getItem(‘key')

localStorage.removeItem('key')
```

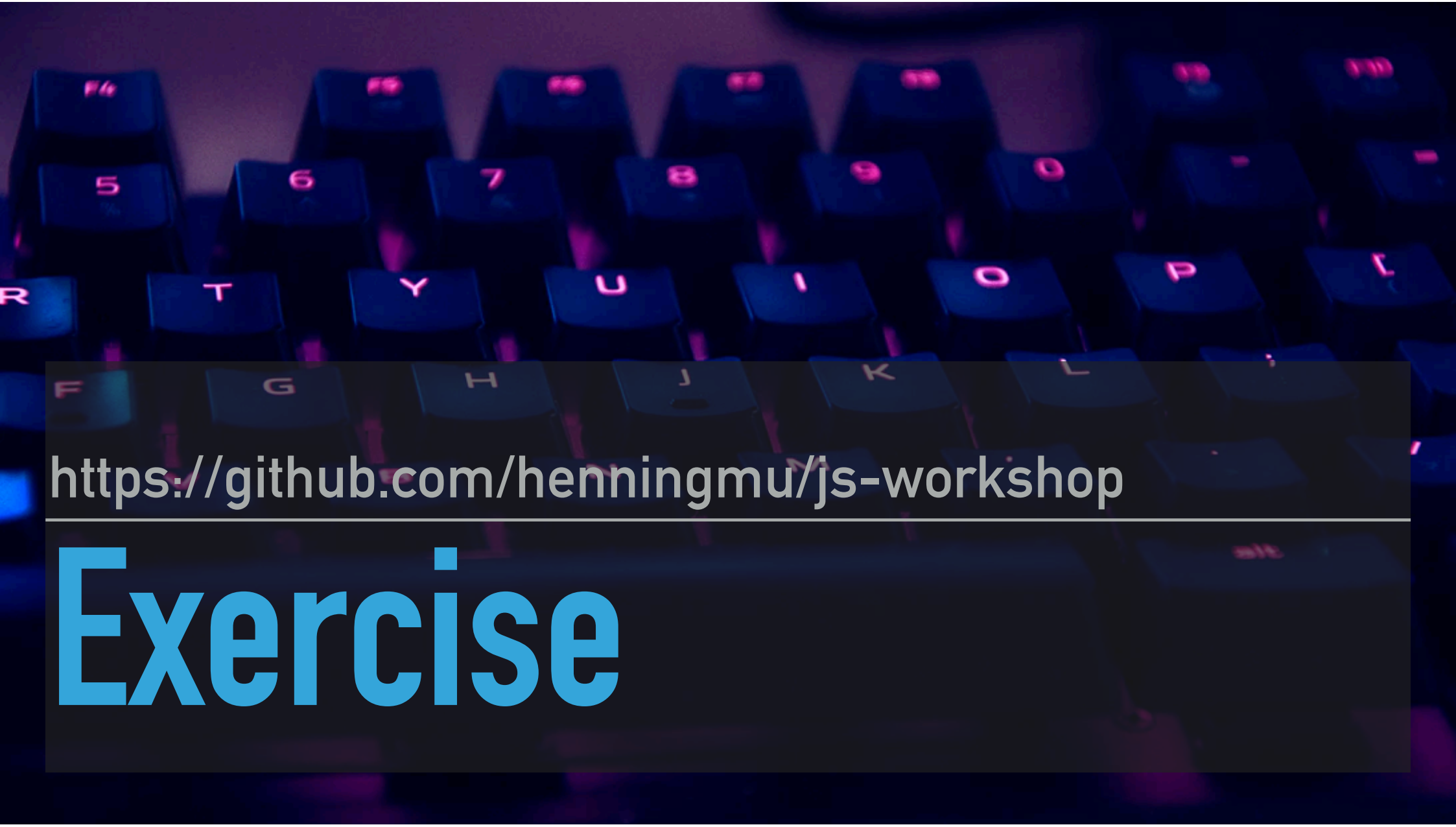
Store data in LocalStorage

Read and write complex data

```
const data = [{ id: 1, text: 'Milk' }, { id: 2, text: 'Bread' }]

setItem('data', JSON.stringify(data))

JSON.parse(getItem('data'))
```



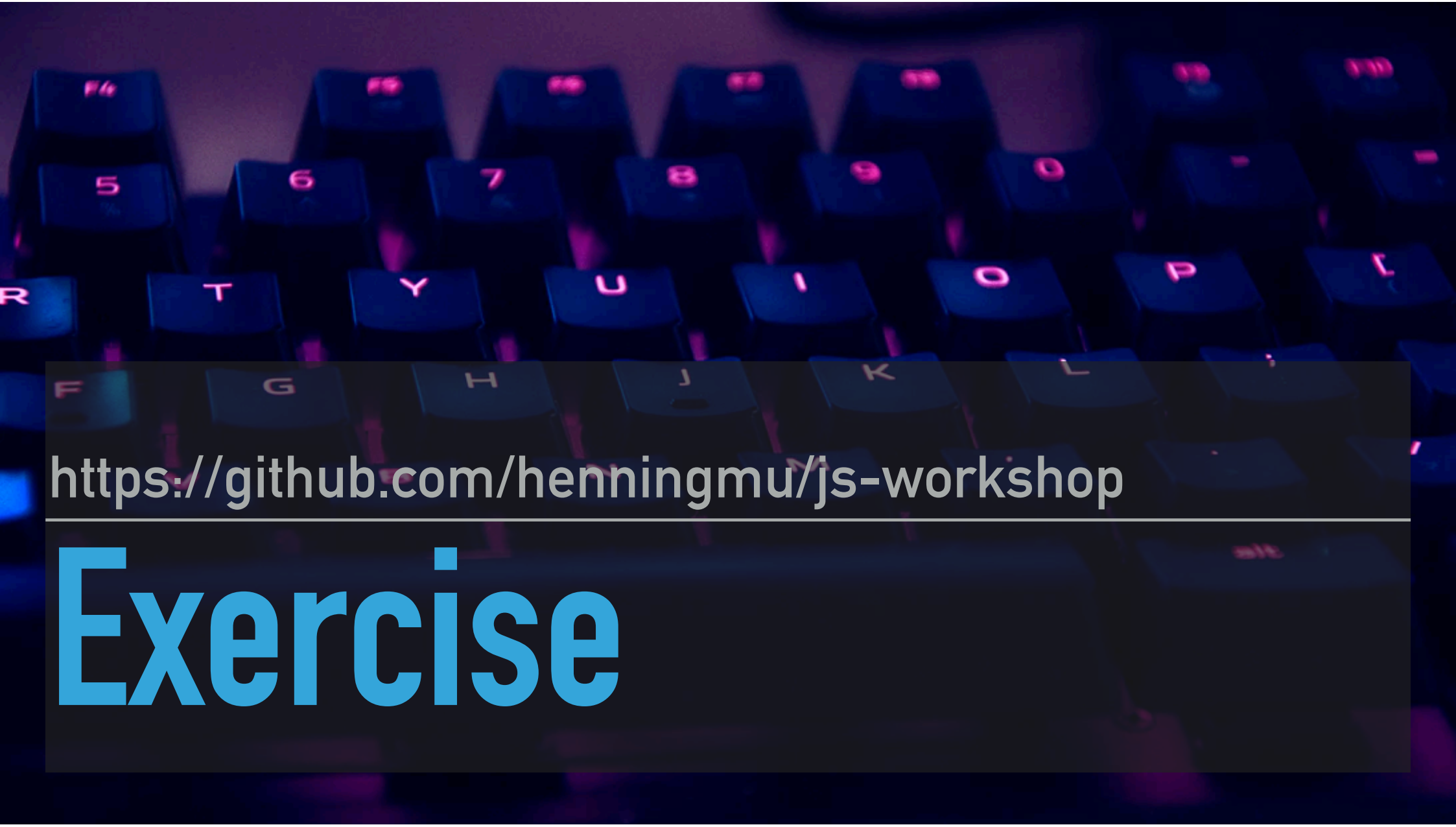
https://github.com/henningmu/js-workshop

Exercise

Get the current location with the Geolocation API

Requires user permission – so we should always handle errors gracefully

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
        (position) => {
            const { latitude, longitude } = position.coords
            console.log('Yeah:', latitude, longitude)
        },
        (error) => {
            console.log('Oh no:', error)
        }
    )
}
```

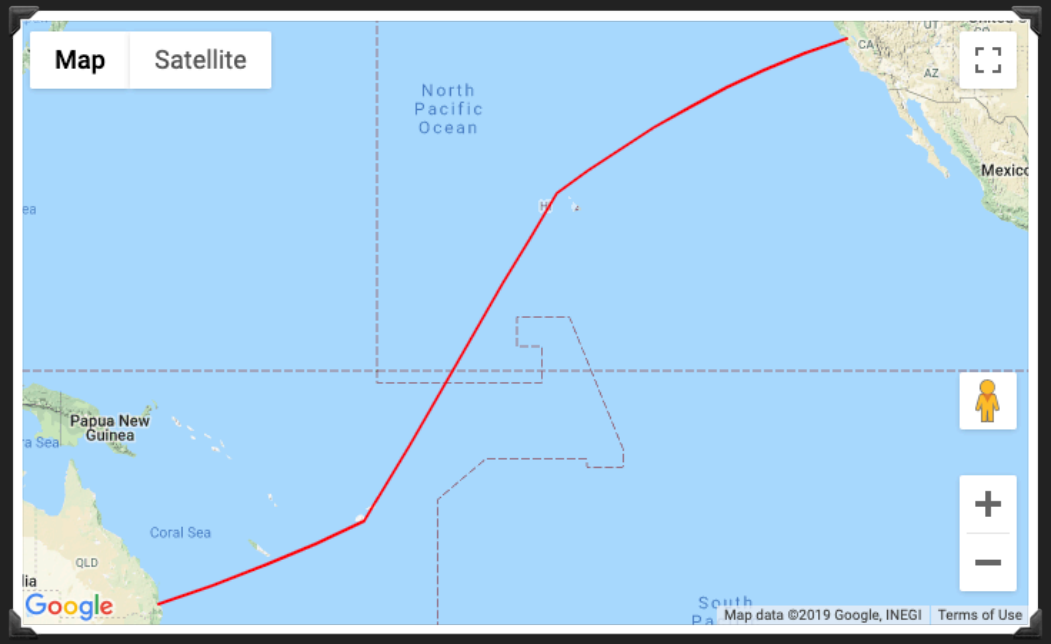


https://github.com/henningmu/js-workshop

Exercise

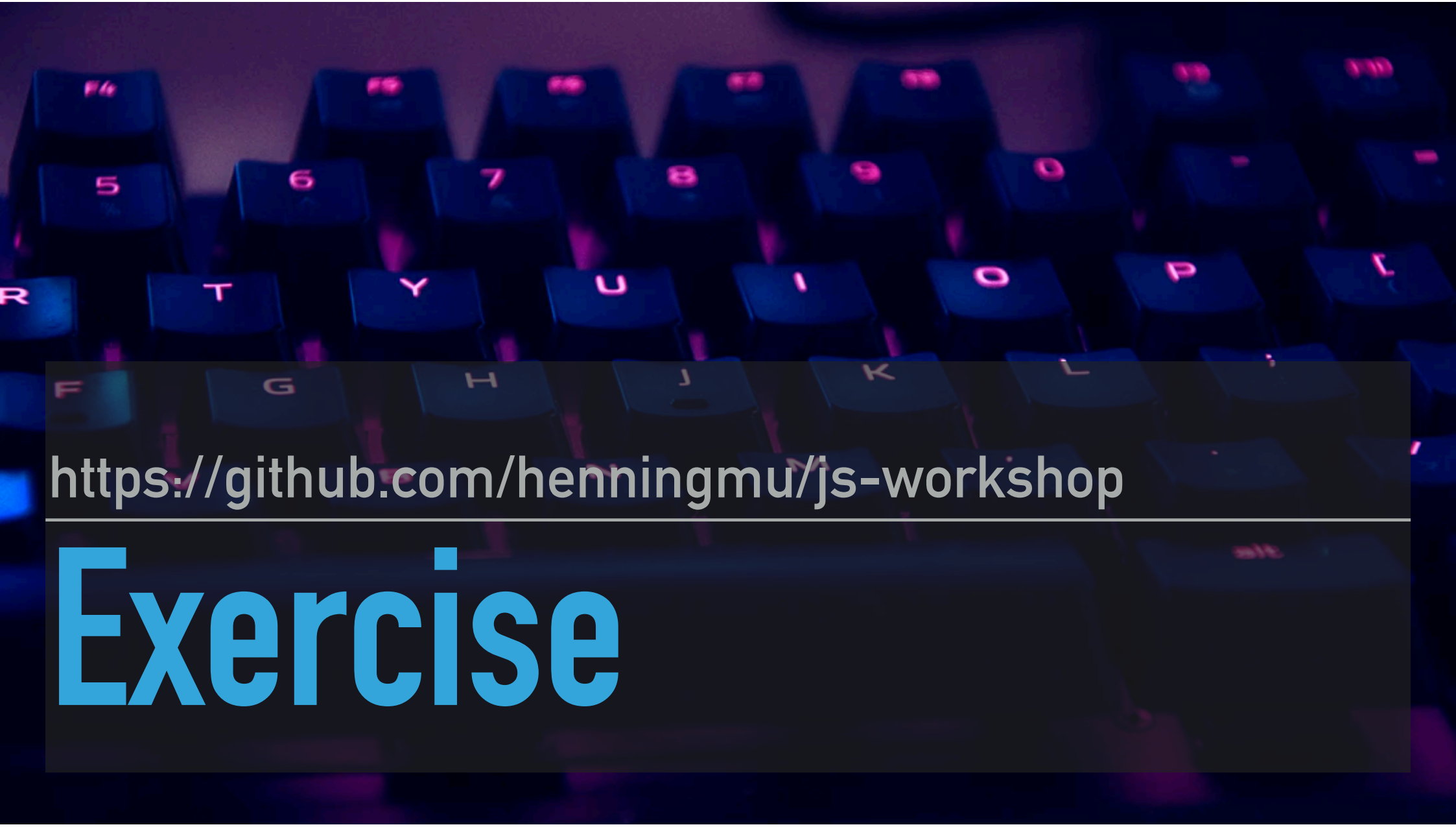
Display a map with Google Maps API

- Interactive map of any location
- Zoom levels (continents to houses)
- Markers and Lines



```
<html lang="en">
  <head>
    <script src="googleapis.com?callback=initMap" async defer></script>
    <script>
      let map

      function initMap() {
        const mapElement = document.querySelector('#map')
        map = new google.maps.Map(mapElement, {
          center: { lat: -33.9249, lng: 18.4241 },
          zoom: 12
        })
      }
    </script>
  </head>
  <body>
    <div id="map"></div>
  </body>
</html>
```



https://github.com/henningmu/js-workshop

Exercise

Quick Feedback