Hamburg Coding School

Henning Muszynski

Session 2

JavaScript for Web

Feedback

What did we learn last time

Re-examination

Modern JavaScript: Arrow Functions

```
function sumOfApples(bucket1, bucket2) {
    const sum = bucket1 + bucket2
    return sum
 }
```

```
const sumOfApples = (bucket1, bucket2) => {
    const sum = bucket1 + bucket2
    return sum
}
// or with implicit return:
const sumOfApples = (bucket1, bucket2) => bucket1 + bucket2
```

Modern JavaScript: Arrow Functions

Special case: Single parameter can omit brackets

```
const log = value => console.log('Value:', value)
```

Special case: No parameter have empty brackets

```
const sayHello = () => console.log('Hello')
```

Modern JavaScript: Template Literals

String concatenation is cumbersome:

```
const person = { name: 'Henning', age: 28, role: 'Engineer' }

console.log(person.name + ' is ' + person.age + ' years old.')

// logs: Henning is 28 years old
```

```
const person = { name: 'Henning', age: 28, role: 'Engineer' }

console.log(`${person.name} is ${person.age} years old.`)

// logs: Henning is 28 years old
```

Modern JavaScript: Array Destructuring

Select the values you're interested in

```
const values = ['Henning', 28, 'Engineer']
const [name, age] = values
```

Modern JavaScript: Object Destructuring

Select the properties you're interested in

```
const person = { name: 'Henning', age: 28, role: 'Engineer' }
const { name, age } = person
```

Supply default values for unknown properties

```
const person = { age: 28, role: 'Engineer' }
const { name = 'Henning', age = 20 } = person
```

Modern JavaScript: Object Spreading

Problem: Copying objects

```
const person = { name: 'Henning', age: 28 }
const person2 = person
person2.name = 'Nina'

console.log(`${person.name} is ${person.age} years old.`)
// prints: Nina is 28 years old
```

Modern JavaScript: Object Spreading

```
const person = { name: 'Henning', age: 28 }
const person2 = { ...person }
person2.name = 'Nina'

console.log(`${person.name} is ${person.age} years old.`)
// prints: Henning is 28 years old

console.log(`${person2.name} is ${person2.age} years old.`)
// prints: Nina is 28 years old
```

Modern JavaScript: Array Spreading

```
const values1 = [1, 2, 3]
const values2 = [4, 5, 6]
const allValues = [...values1, ...values2] // [1, 2, 3, 4, 5, 6]

// used less often as there's also:
const allValues = values1.concat(values2)
```

Modern JavaScript: Array.map()

Execute a function for each element in the array and return a new array

```
const numbers = [1, 2, 3, 4]
const doubles = numbers.map(number => number * 2)
// [2, 4, 6, 8]
```

Modern JavaScript: Array.filter()

Evaluate a condition for each element in an array and return array with elements that evaluate to true

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8]
const evens = numbers.filter(number => number % 2 === 0)
// [2, 4, 6, 8]
```

Modern JavaScript: Array.reduce()

Execute a function for each element in the array, return a single reduced result

```
const numbers = [1, 2, 3, 4]
const initialValue = 0
const sum = numbers.reduce((result, number) => {
    return result + number
}, initialValue)

// sum is 10
```

Recap Array Methods

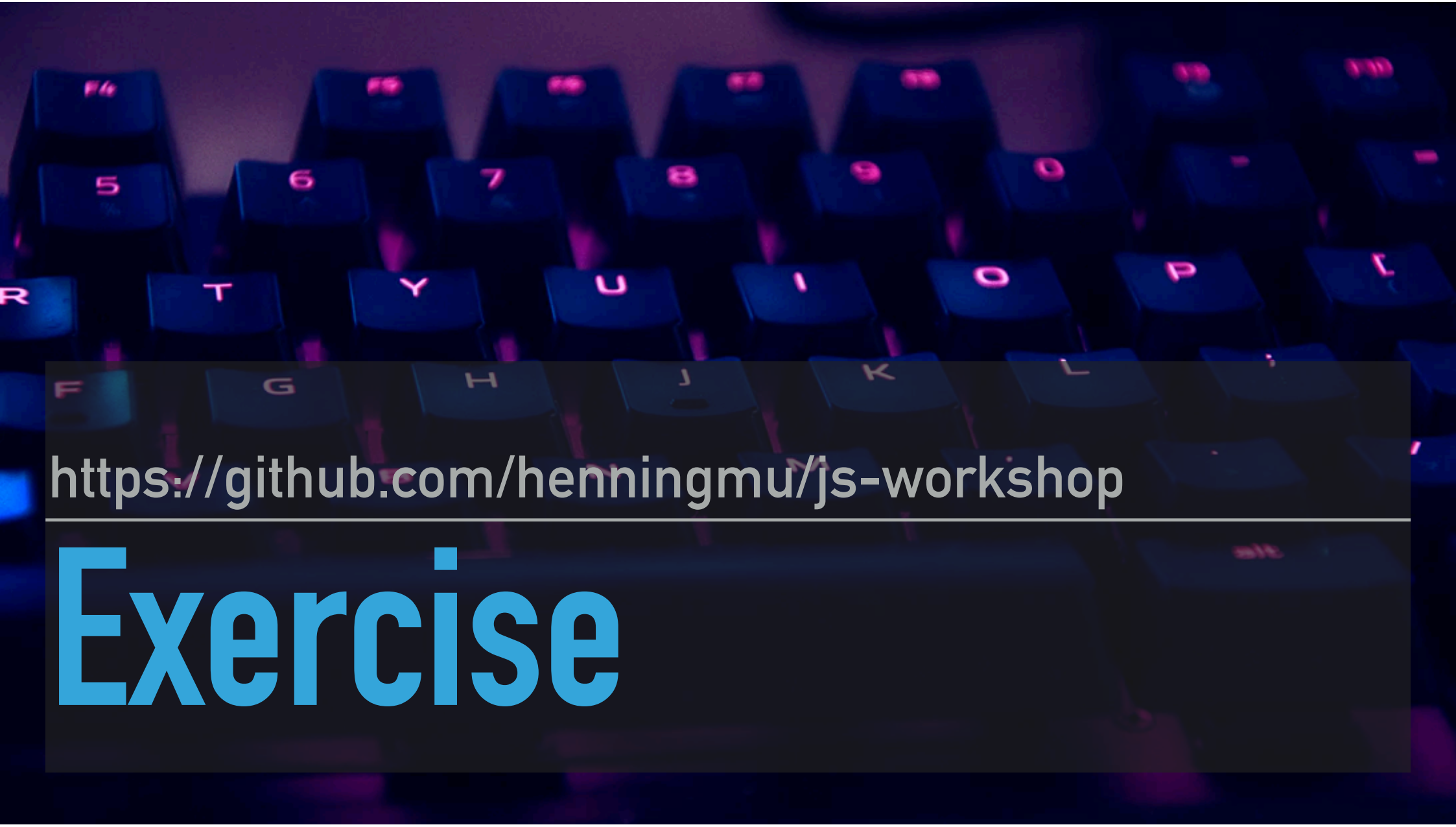
- **Array.map** Change each element, return array of same size
- **Array.filter** Filter only relevant values from array, return new array
- **Array.reduce** Create a single value from array

Modern JavaScript: Ternary Operator

Not really new, but experiences a revival

```
const number = 3
if (number > 5) {
    console.log('Larger than 5')
} else {
    console.log('Lesser or equal 5')
}
// Lesser or equal 5

number > 5
    ? console.log('Larger than 5')
    : console.log('Lesser or equal 5')
// prints: Lesser or equal 5
```

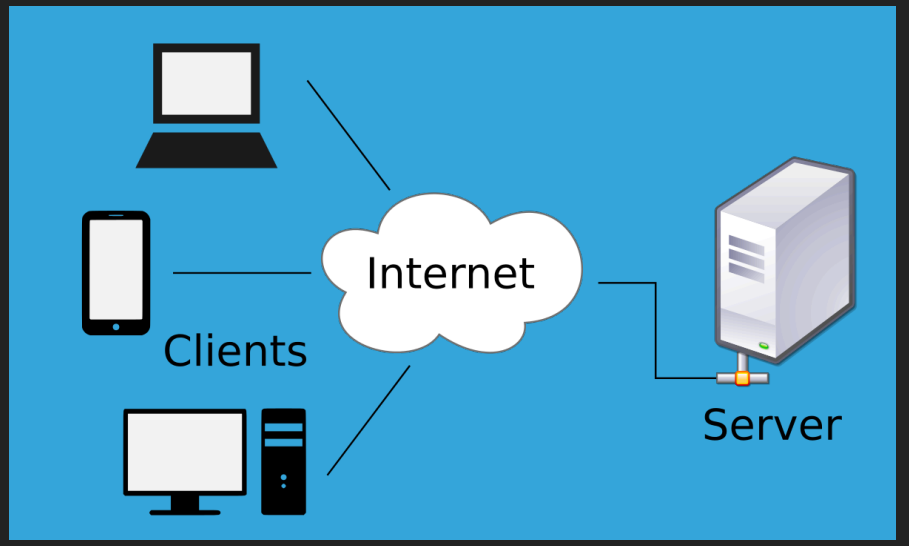


https://github.com/henningmu/js-workshop

Exercise

Frontend Architectures

- Server-Rendered Apps
- Single Page Apps
- Isomorpic / Universal Apps



- It always depends – in this course we build a server-rendered app

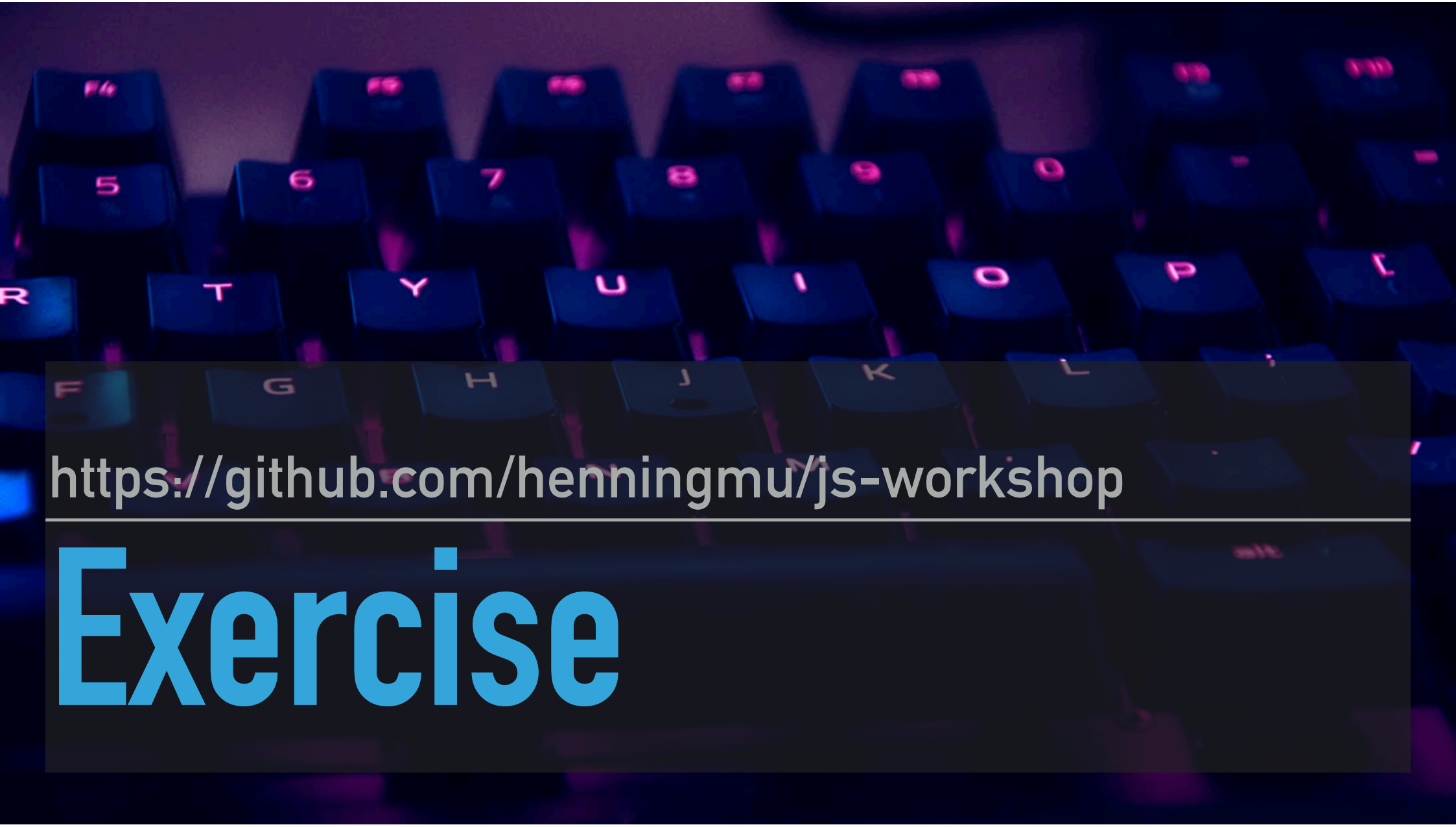
Parts of a web application

- HTML
- CSS
- JavaScript

```
<!DOCTYPE html>
<html>
    <head>
        <title>Web Application</title>
        <link rel="stylesheet" href="styles.css" />
        <script type="javascript">
            /* ... */
        </script>
    </head>
    <body>
        <h1>Hello World</h1>
    </body>
</html>
```

JavaScript and the DOM

```
<html>
    <head>
        <script>
            function setValue() {
                const valueElm = document.getElementById('value')
                valueElm.innerHTML = 13
            }
        </script>
    </head>
    <body>
        <button onclick="setValue()">Set Value to 13</button>
        <p id="value">0</p>
    </body>
</html>
```



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

Exercise

Recap JSON

Data in JavaScript

```
const data = {
    keyString: 'Text value',
    keyNumber: 123,
    keyObject: {
        key: 'value'
    },
    keyArray: [1, 'a', { key: 'value' }]
}
```

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) => {
    /* ... */
})
```

Quick Feedback