DOM MANIPULATION

CONDITIONALLY SHOWING DATA

AGENDA

- Navigating the DOM
- Attributes
- Arrays / `.querySelectorAll`
- Removing content
- Conditional data
- Data attributes
- `productlist.html.#1`

PURPOSE OF TODAY

Navigating and controlling the DOM, remember

We can find ANYTHING in the DOM

We can create ANYTHING in the DOM

We can modify ANYTHING in the DOM

NAVIGATING THE DOM

LOCATING AN ELEMENT

```
1  //Our good old friend, grab the first nested element
2  //that match the selector
3  elem.querySelector(selector);
4
5  //Our new friend, grab all nested elements
6  //that match the selector
7  elem.querySelectorAll(selector);
8
9  //With one element selected, grab it's parent
10 elem.parentElement;
11
12 //With one element selected, grab it's next sibling
13 elem.nextElementSibling;
14
15 //With one element selected, grab it's previous sibling
16 elem.previousElementSibling;
```

elem.querySelector

With something selected, we can search through it's children

```
...
<header>
  <nav id="main">
    <a href="#">Home</a>
    <a href="#">About us</a>
  </nav>
  <p>Please click something</p>
</header>
<article>
  <header>
    <h1>An ode to JS</h1>
    <h2>By <a href="#">Jonas</a></h2>
  </header>
  <p>content goes here</p>
</article>
```

```
const header = document.querySelector("header");

/*
with header selected we can
```

elem.querySelectorAll

```
...
<header>
  <nav id="main">
    <a href="#">Home</a>
    <a href="#">About us</a>
  </nav>
  <p>Please click something</p>
</header>
<article>
  <header>

    <h1>An ode to JS</h1>
    <h2>By <a href="#">Jonas</a></h2>
  </header>
  <p>content goes here</p>
</article>
```

```
const allLinks = document.querySelectorAll("a");

const paragraphs = document.querySelectorAll("p");
```

This one is a bit harder, we'll get back to it

elem.parentElement

```
...
<header>
  <nav id="main">
    <a href="#">Home</a>
    <a href="#">About us</a>
  </nav>
  <p>Please click something</p>
</header>

<article>
  <header>
    <h1>An ode to JS</h1>
    <h2>By <a href="#">Jonas</a></h2>
  </header>
  <p>content goes here</p>
</article>
```

```
const header = document.querySelector("article header");

const article = header.parentElement;
```

elem.nextElementSibling

```
...
<header>
  <nav id="main">
    <a href="#">Home</a>
    <a href="#">About us</a>
  </nav>
  <p>Please click something</p>
</header>
<article>
  <header>

    <h1>An ode to JS</h1>
    <h2>By <a href="#">Jonas</a></h2>
  </header>
  <p>content goes here</p>
</article>
```

```
const header = document.querySelector("header");

//what does 'header' contain now?

const next = header.nextElementSibling;

//what's inside 'next' now?

```

CHAINING

We often see these elements used "chained"

```
1 const h2 = document
2   .querySelector("header h2")
3   .previousElementSibling
4   .parentElement
5   .parentElement
6   .previousElementSibling
7   .querySelector("p");
```

ATTRIBUTES

setAttribute

With `elem.setAttribute('atr', 'new')` we can set the "attributes" on any element in the DOM

```
1 const image = document.querySelector("img.gallery");
2
3 image.addEventListener("click", swapImg);
4
5 function swapImg() {
6   image.setAttribute("src", "selfie.jpg");
7   image.setAttribute("alt", "A selfie");
8 }
```

we have changed attributes with `.src` which works just fine

getAttribute

We can also "read" attributes

```
1 const current = document.querySelector('img.gallery');
2 console.log(
3     current.getAttribute('src');
4 );
```

We could also read the src with `.src` but it behaves strange

ARRAYS

Well, actually, `NodeLists`

- Arrays are still a bit hard
- But oh so useful
- `.querySelectorAll()` gives us a NodeList
- Which works almost like an array
- e.g. we can use `forEach`

`forEach` WITH NAMED FUNCTIONS

```
1  const products = document.querySelectorAll("div.product");
2
3  products.forEach(handleProduct);
4
5  function handleProduct(elem) {
6    //we now have the dom elements one at a time, as 'elem'
7    elem.addEventListener("click", productClicked);
8  }
9
10 function productClicked(e) {
11   //e is the event object
12   //e.target is the element that was clicked
13   e.target.classList.add("selected");
14   //this.classList.add("selected"); // is the same
15 }
```

forEach WITH ARROW FUNCTIONS

```
1  const products = document.querySelectorAll("div.product");
2
3  products.forEach((elem) => {
4    //we now have the dom elements one at a time, as 'elem'
5    elem.addEventListener("click", productClicked);

6  });
7
8  function productClicked(e) {
9    //e is the event object
10   //e.target is the element that was clicked
11   e.target.classList.add("selected");
12   //this.classList.add("selected"); // is the same
```

`forEach` WITH ARROW FUNCTIONS, CONDENSED

```
1 document.querySelectorAll("div.product").forEach((elem) => {
2   //we now have the dom elements one at a time, as 'elem'
3   elem.addEventListener("click", (e) => {

4     //e is the event object
5     //e.target is the element that was clicked
6     e.target.classList.add("selected");
7     //this.classList.add("selected"); // is the same
8   });
```

REMOVING CONTENT

One of the easier ones :-)

```
elem.remove()
```

REMOVING SOME <DIV>

```
const div = document.querySelector("div");
div.remove();
//or just
document.querySelector("div").remove()
```

EXERCISE

Fronter: 1. DOM Manipulation

You get 20 minutes, finish the rest as homework

A quick look at the exercise

Remember, you can discuss this on Teams

CONDITION AL DATA?

What's that?

- Non-static sites are full of it.
- if you're logged in, you see one thing, otherwise you see something else
- Based on your actions on a site, things can change
- Some products are sold out, some are discontinued etc
- Designing for these circumstances require a little

TECHNICAL APPROACHES

1. Have the DOM element in your template, and hide it if you do not need it
2. Have the DOM element, hidden in your template, and show it if you need it
3. Have the DOM element in your template, remove it if you do not need it
4. Make JS create the element when needed

USING THE CASCADE

REMEMBER THE C IN CSS?

If you want to hide/show something, you don't HAVE to put the class on that element, you can use the cascade

Two versions:

DIRECTLY ON THE ELEMENT

```
<article>
  <h1>Some product</h1>
  <p class="discount hidden">On discount</p>
</article>
```

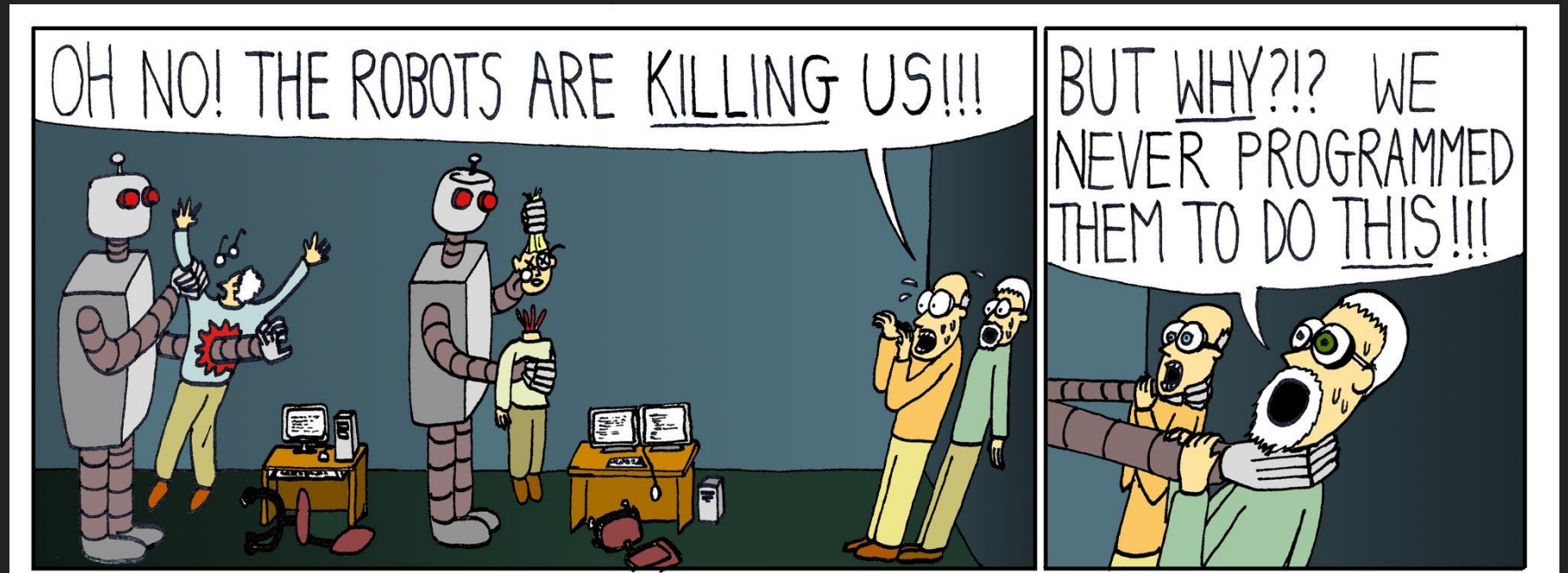
```
.hidden {
  display: none;
}
```

ON THE PARENT

```html
<article class="discount">
  <h1>Some product</h1>
  <p>On discount</p>
</article>
```

```css
p {
  display: none;
}
.discount p {
  display: block;
}
```

CONDITIONS & SAMPLE CODE


OH NO! THE ROBOTS ARE KILLING US!!!
BUT WHY?!? WE NEVER PROGRAMMED THEM TO DO THIS!!!

Remember `if`-statements?

```
let name = "";
if (name) {
  //1. will this run?
}

let age;
if (age) {
  //2. will this?
}


let items = ["beer", "cabbage"];
if (items.length) {
  //3. me?
}

if (hobby) {
  //4. or this?
}
```

```
let drinksLeft = 0;
if (drinksLeft) {
  //5. what about me?
}
```

SAMPLE CODE

Hide / show depending on data

```
1 if (product.discount) {
2   clone.querySelector(".product").classList.add("discount");
3 }
```

Create an element based on data

```
1 if (product.discount) {
2   let p = document.createElement("p");
3   p.textContent = "I'm on sale";
4   clone.appendChild(p);
5 }
```

EXERCISE

`productlist.html`, first iteration

1. Follow the process in the video
2. This one will be a little harder than the last time, because our DOM's will be very different
3. But remember, you can `.remove()` things in your clone
4. And use all the other cool methods for navigating the DOM inside your clones