THE

<template>

AND THE DOM

TODAY, PART 2

1. Left overs
2. Understanding the DOM
3. Basic DOM manipulation
4. `<template>`'s
5. Template literals

LEFT OVERS

1. Youtube playlist (live recordings)
2. Friday Bar
3. Information about the top-ups
4. The info channel on teams

UNDERSTANDING THE DOM

VOCABULARY

- DOM
- Element
- Node
- Document
- Document fragment
- Parent
- Child

DOM

The Document Object Model (DOM) is an application programming interface (API) for valid HTML ... documents. It defines the logical structure of documents and the way a document is accessed and manipulated... ...With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML

Element

The HTML elements in the DOM

Node

Common term for "Element Nodes", "Text Nodes" & "Attribute Nodes"

Document

In the HTML DOM (Document Object Model), everything is a node:

- ***The document itself is a document node***
- ***All HTML elements are element nodes***
- ***All HTML attributes are attribute nodes***
- ***Text inside HTML elements are text nodes***
- ***Comments are comment nodes***

Document Fragment

The DocumentFragment interface represents a minimal document object that has no parent. It is used as a light-weight version of Document to store a segment of a document structure comprised of nodes just like a standard document. The key difference is that ... the document fragment isn't part of the actual DOM's structure

MDN

Parent

All "Elements" or "Nodes" higher in the hierachy

Child

All "Elements" or "Nodes" lower in the hierachy

Sibling

All "Elements" or "Nodes" on the same level in the hierachy

BASIC DOM MANIPULATION

Here's what I **hope** you know

```
1  const myEl = document.querySelector("css selector");
2  const myNewEl = document.createElement("some html el");
3
4  myNewEl.textContent = "some text";
5  myEl.innerHTML = "<p>Some HTML</p>";
6
7  myEl.appendChild(myNewEl);
8
9  myEl.classList.add("some css class");
10 myEl.classList.remove("some css class");
11 myEl.classList.toggle("some css class");
12
13 if (myEl.classList.contains("some css class")) {
14   console.log("do something");
15 }
```

How's our status on this?

<template>

The HTML template element `<template>` is a mechanism for holding client-side content that is not to be rendered when a page is loaded but may subsequently be instantiated during runtime using JavaScript.

<template>'s in html

```
<p>regular HTML</p>
...

<template id="myTemplate">
    <article>
        <header>
            <h1>HEADER</h1>
        </header>
        <div class="content">
            MORE CONTENT
        </div>
    </article>
</template>
...
```

LET'S TRY IT

CLONING

PROCEDURE

1. Select the `<template>`'s content
2. Make a "clone / copy"
3. Change the content in the clone
4. Choose the "parent" element
5. Add (Append) the clone to the parent (and thus the DOM)

```
1  //Step 1: Chose the <template>'s content
2  const templateElement = document.querySelector("#myTemplate").c
3
4  //Step 2: Make a "clone"
5  const myClone = templateElement.cloneNode(true);
6
7  //Step 3: Change the content
8  myClone.querySelector("h1").textContent = "Hi Mom";
9  //...
10
11 //Step 4: Chose the new "parent" element
12 const parentElement = document.querySelector("body");
13
14 //Step 5: Add (Append) the clone to the DOM
15 parentElement.appendChild(myClone);
```

LET'S TRY IT

EXERCISE

Fronter: "1. My Little Template"

`.textContent` VS `.innerHTML`

`.textContent` sets the text on an element, and will convert HTML into text

`.innerHTML` sets the text on an element, and allows HTML

`.textContent` is generally preferred

```
1 const el = document.querySelector("p");
2
3 //the browser turns it into text
4 //the user sees the HTML tag
5 el.textContent = "<h1>Hi mom</h1>";
6
7 //the browser turns it in to html
8 //the user sees big text
9 el.innerHTML = "<h1>Hi mom</h1>";
```

INTERPOLATION

interpolation

`/ɪntəːpəˈleɪʃ(ə)n/`

noun
"the insertion of something of a different nature into something else."

Also known as "concatenation", it's basically when we glue together stuff to make new stuff

Two versions exist

The old:

```
1 const name = "Jonas";
2 const age = "none of your business";
3
4 const newString = "Hello, my name is " + name + " and my age is "
```

And the new:

```
1 const name = "Jonas";
2 const age = "none of your business";
3
4 const newString = `Hello, my name is ${name} and my age is ${ag
```

notice the ` backtick character

Both versions are fine, personally, I prefer the new