JavaScript Day - 13

Modifying the Document

Modifying the Document

1. Creating an Element
2. Insertion Methods
3. insertAdjacentHTML
4. Node removal
5. Cloning nodes

Creating an Element

To create DOM nodes, there are two methods:

- **document.createElement(tag)**: Creates a new element node with the given tag
- **document.createTextNode(text):** Creates a new text node with the given text

Creating an Element

Creating an element takes 3 steps

- Create an element
- Set it's attributes if any
- Fill it with content

```
1  // 1. Create <div> element
2  let div = document.createElement('div');
3
4  // 2. Set its class to "alert"
5  div.className = "alert";
6
7  // 3. Fill it with the content
8  div.innerHTML = "<strong>Hi there!</strong> You've read an important message.";
```

We've created the element. But as of now it's only in a variable named div, not in the page yet. So we can't see it.

Insertion Methods

To make the div show up, we need to insert it somewhere into document. For instance, into <body> element, referenced by document.body

There's a special method append for that: document.body.append(div)

```
12    let div = document.createElement('div');
13    div.className = "alert";
14    div.innerHTML = "<strong>Hi there!</strong> You've read an important message.";
15
16    document.body.append(div);
```

Insertion Methods

Here are more insertion methods, they specify different places where to insert:

- **node.append(...nodes or strings)** – append nodes or strings at the end of node,
- **node.prepend(...nodes or strings)** – insert nodes or strings at the beginning of node,
- **node.before(...nodes or strings)** –- insert nodes or strings before node,
- **node.after(...nodes or strings)** –- insert nodes or strings after node,
- **node.replaceWith(...nodes or strings)** –- replaces node with the given nodes or strings.

Arguments of these methods are an arbitrary list of DOM nodes to insert, or text strings (that become text nodes automatically).

Insertion Methods

The methods discussed can only be used to insert DOM nodes or text pieces.

But what if we'd like to insert an HTML string "as html", with all tags and stuff working, in the same manner as elem.innerHTML does it?

insertAdjacentHTML

For that we can use another, pretty versatile method:
elem.insertAdjacentHTML(where, html).

The first parameter is a code word, specifying where to insert relative to elem.
Must be one of the following:

- "beforebegin" – insert html immediately before elem,
- "afterbegin" – insert html into elem, at the beginning,
- "beforeend" – insert html into elem, at the end,
- "afterend" – insert html immediately after elem.

The second parameter is an HTML string, that is inserted “as HTML”.

Node Removal

To remove a node, there's a method **node.remove().**

Cloning Elements

How to insert one more similar message?

The way would be to clone the existing div and modify the text inside it (if needed).

Sometimes when we have a big element, that may be faster and simpler.

- The call **elem.cloneNode(true)** creates a “deep” clone of the element – with all attributes and subelements.
- If we call **elem.cloneNode(false)**, then the clone is made without child elements.

Question

We have an empty DOM element elem and a string text.

Which of these 3 commands will do exactly the same?

- elem.append(document.createTextNode(text))
- elem.innerHTML = text
- elem.textContent = text

Question

Create a function clear(elem) that removes everything from the element.

```
<ol id="elem">
  <li>Hello</li>
  <li>World</li>
</ol>

<script>
  function clear(elem) { /* your code */ }

  clear(elem); // clears the list
</script>
```

Question

Create a function clear(elem) that removes everything from the element.

```
<ol id="elem">
  <li>Hello</li>
  <li>World</li>
</ol>

<script>
  function clear(elem) { /* your code */ }

  clear(elem); // clears the list
</script>
```

Question

Write an interface to create a list from user input.

For every list item:

- Ask a user about its content using prompt.
- Create the <li> with it and add it to <ul>.
- Continue until the user cancels the input (by pressing Esc or via an empty entry).
- All elements should be created dynamically.

If a user types HTML-tags, they should be treated like a text.