

The background is a complex, abstract composition in various shades of teal and blue. It features a collage of elements: a large, dark, circular shape resembling a tunnel or a stylized eye in the upper center; a crescent moon in the upper right; a brick wall on the left; a ladder-like structure on the bottom left; and various geometric and organic shapes throughout, creating a layered, architectural feel.

JavaScript

Working with the DOM

Peter Kassenaar – info@kassenaar.com

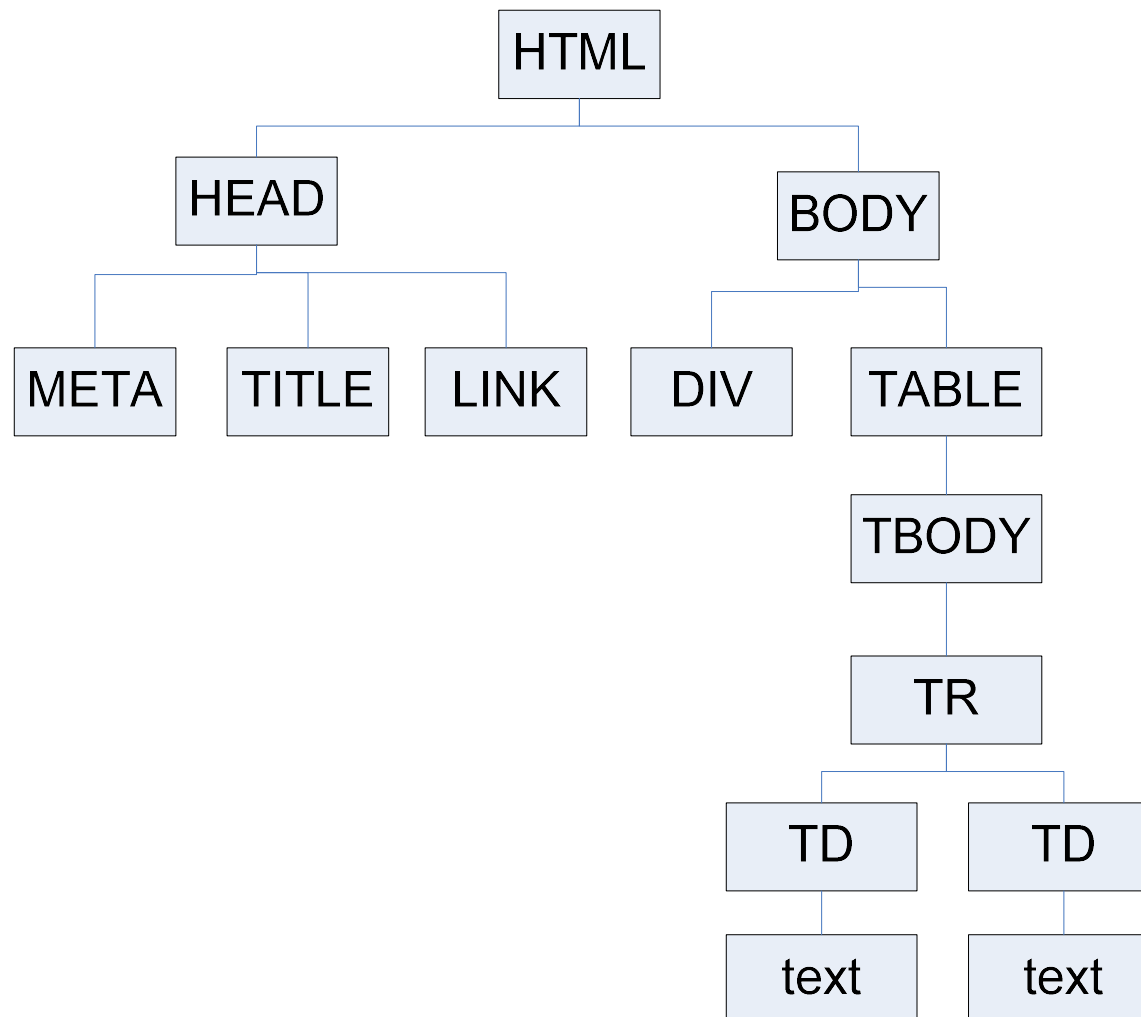
What is HTML DOM?

Built into **every browser** – enabling JavaScript to interact with the web page

*“The HTML DOM (Document Object Model) is a **hierarchical representation** of a webpage's **structure**.*

*This enables JavaScript to **manipulate**, **style**, and **interact** with elements.”*

Document Object Model

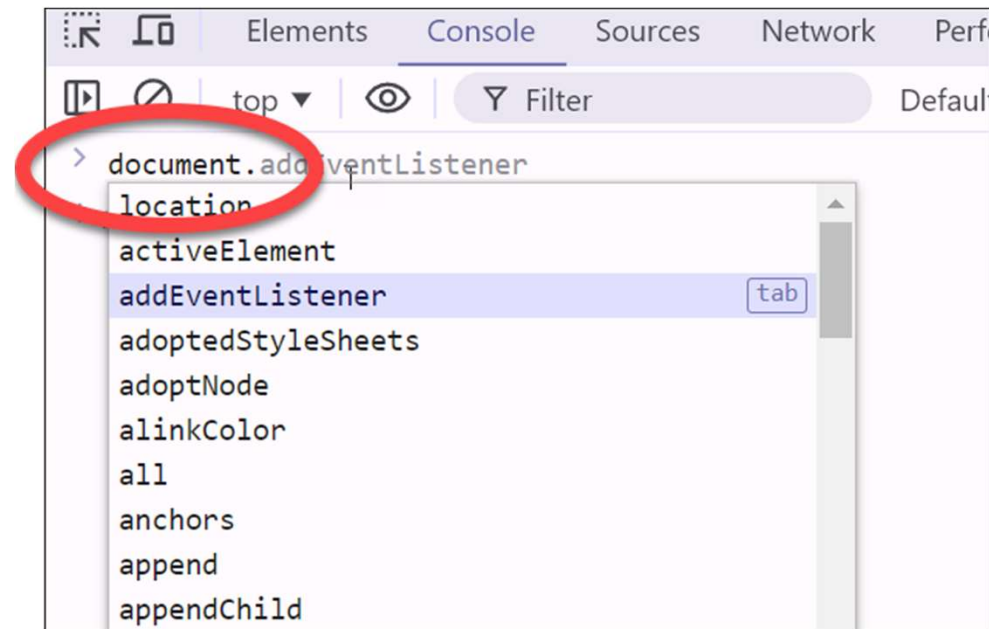
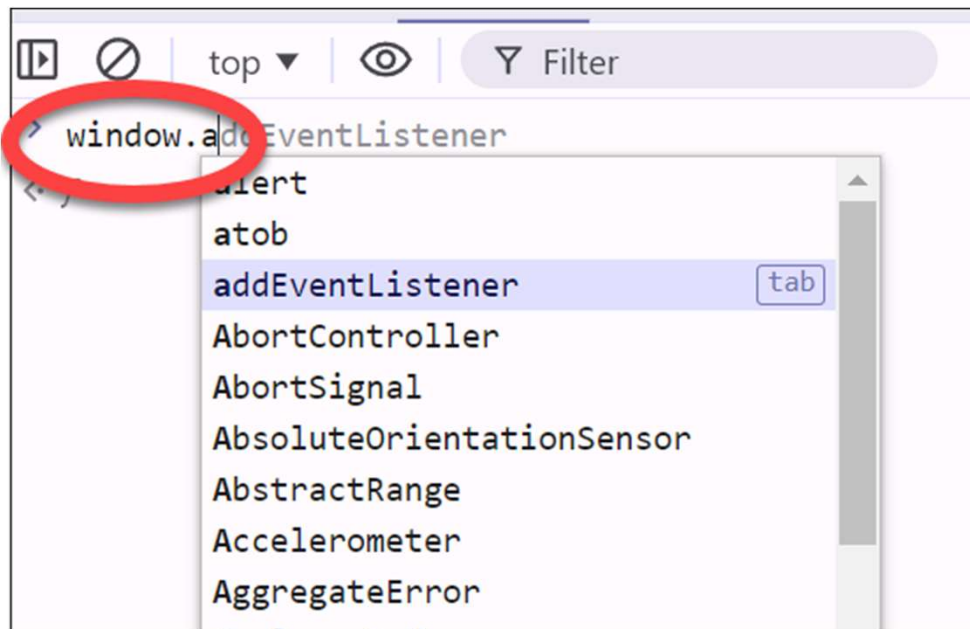


Document Object Model

- Built-in object structure in the browser
- HTML elements are accessible via JavaScript
- Dynamic adaptation of HTML
 - Formerly: 'DHTML'
- Standardised by W3C (DOM level 2)
- Early days: differences per browser
 - Nowadays: more standardised

Browser is accessible via JavaScript:

- Functions `window.*` - access to the browser itself
 - Address bar, history state, browser properties etc.
- Functions `document.*` access to the page inside the browser



Methods on document

- Some important and well-known methods on document:

- `document.getElementById();`
- `document.createElement();`
- `document.createTextNode();`
- `document.getElementsByTagName();`
- `document.getElementsByClassName();`

More document methods

- Also:

- `.getElementsByClassName()`
- `.getElementsByTagName()`

- Important:

- `.querySelector()`
- `.querySelectorAll()`

- Deprecated:

- `document.write(); // Don't use!`

DOM - usage

- **Manipulate** the page
 - Add and remove text, images, links, etc.
 - Change properties of elements (visibility, position, color, etc).
- Create **webapplications** instead of static *websites*
- **External** communication:
 - via **http/AJAX** - communication with a webserver
 - via **events** - interaction with the user

Document Object Model

document
-body -documentElement -implementation
+createElement() +createTextNode() +getElementById() +getElementsByTagName()

Document Object Model

Every element (p, h1, a etc) is of type DOMNode.

It has a – really long – list of properties

DOMNode
<ul style="list-style-type: none">-parentNode-firstChild-lastChild-nextSibling-previousSibling-innerHTML-data-id-nodeName-nodeValue-NodeType-specified-tagName-title-className-currentStyle-style
<ul style="list-style-type: none">+getElementsByTagName()+appendChild()+cloneNode()+insertBefore()+removeChild()+replaceChild()+hasChildNodes()+getAttribute()+removeAttribute()+setAttribute()+addEventListener()+addEvent()+removeEvent()+removeEventListener()

Example – methods on window object

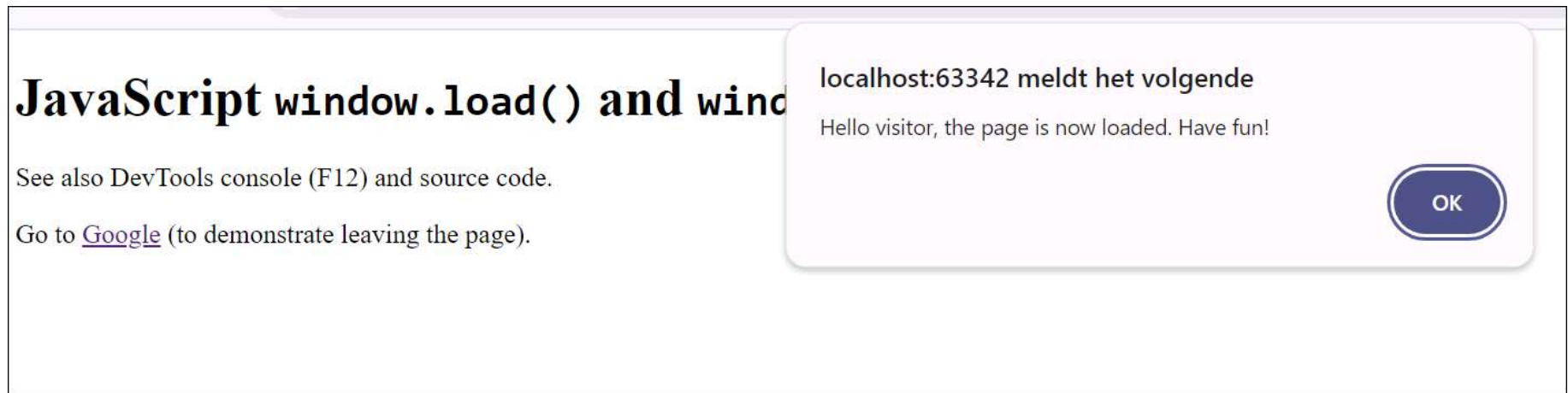
// 1. When entering the page.

```
window.addEventListener('load', function () {  
    alert('Hello visitor, the page is now loaded. Have fun!');  
});
```

// 2. When Leaving the page.

```
window.addEventListener('beforeunload', function (event) {  
    // Cancel the event, so a popup is shown  
    // Most of the times, we CAN NOT set a message ourselves.  
    event.preventDefault();  
    // Chrome requires returnValue to be set  
    event.returnValue = '';  
});
```

Sample output



50_windowEvent.html

Example – methods on the document object

Script

```
// Note - the usage of event listeners inside eventlisteners.
window.addEventListener('load', function () {
    document.getElementById('btnShowName')
        .addEventListener('click', function () {
            const name = document.getElementById('txtName').value;
            const message = 'Welcome ' + name;
            alert(message);
        });
});
```

HTML

```
<h2>Show name in an alert()</h2>
<input type="text" id="txtName" placeholder="Please type your name"/>
<button id="btnShowName">Show Name</button>
```

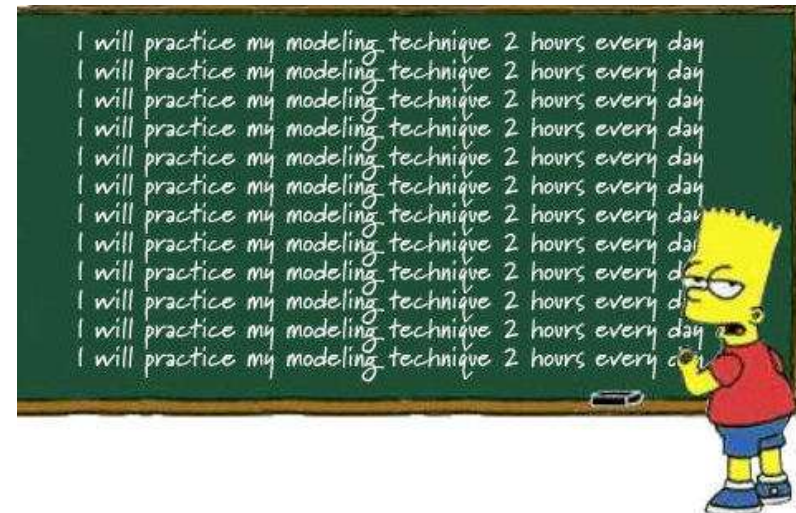
Sample output



51_documentEvent.html

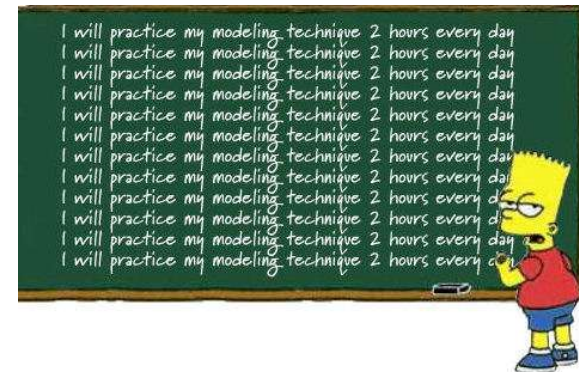
Mini workshop

- Update the script from the previous page so that:
 - A name MUST be entered, otherwise an error message is shown
 - The name must be at least 2 characters in length
 - The name is shown inside the page, instead of an `alert()` window



Mini workshop

- Create a script with an 'unclickable' link
 - How irritating...
- If the mouse hovers over the link, show an alert 'this is to close!'
- Tip: listen for the `mouseover` event on the link or surrounding `<div>`.
- Variant: let the link disappear when the mouse hovers over it.



Further reading

HTML

CSS

JAVASCRIPT

SQL

PYTHON

JAVA

PHP

HOW TO

W3.CSS

C

C++

C#

BOOTSTRAP

REACT

MYSQL

JQUERY

Function Closures

JS Classes

Class Intro

Class Inheritance

Class Static

JS Async

JS Callbacks

JS Asynchronous

JS Promises

JS Async/Await

JS HTML DOM

DOM Intro

DOM Methods

DOM Document

DOM Elements

DOM HTML

DOM Forms

DOM CSS

DOM Animations

DOM Events

DOM Event Listener

DOM Navigation

DOM Nodes

DOM Collections

DOM Node Lists

JavaScript HTML DOM

< Previous

Next >

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **Document Object Model** of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

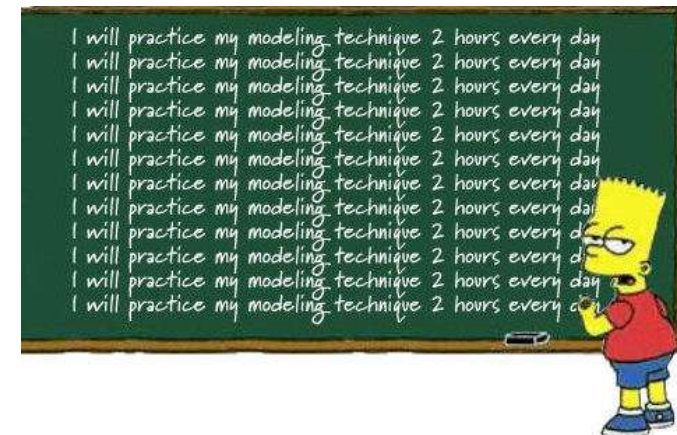
The HTML DOM Tree of Objects

```
graph TD; Document[Document] --> Root["Root element: <html>"]; Root --> Head["Element: <head>"]; Root --> Body["Element: <body>"]; Head --> Title["Element: <title>"]; Title --> TitleText["Text:"]; Body --> Href["Attribute: "href""]; Body --> A["Element: <a>"]; Body --> H1["Element: <h1>"]; A --> AText["Text:"]; H1 --> H1Text["Text:"];
```

https://www.w3schools.com/js/js_htmlDOM.asp

Workshop #20

- Write a script that **adds a paragraph** to a page greeting the visitor.
- The greeting depends on **time of day**: good morning, good afternoon, good evening, good night.
- Use:
 - `new Date().getHours()`
 - `document.createElement()`
 - `.textContent` or `.innerHTML`
 - `document.getElementById('xxx')`
`.appendChild();`

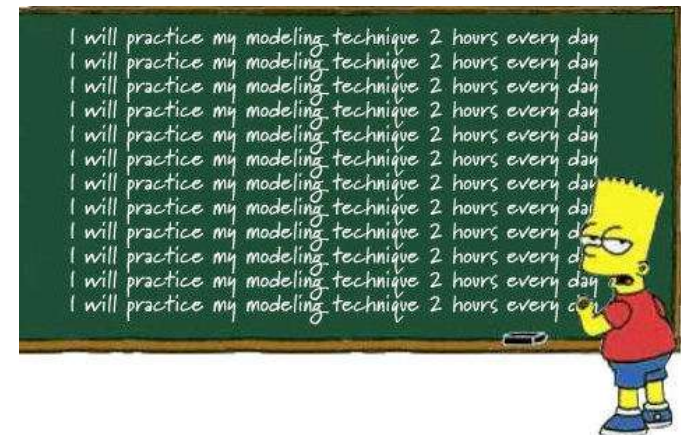


Sample output

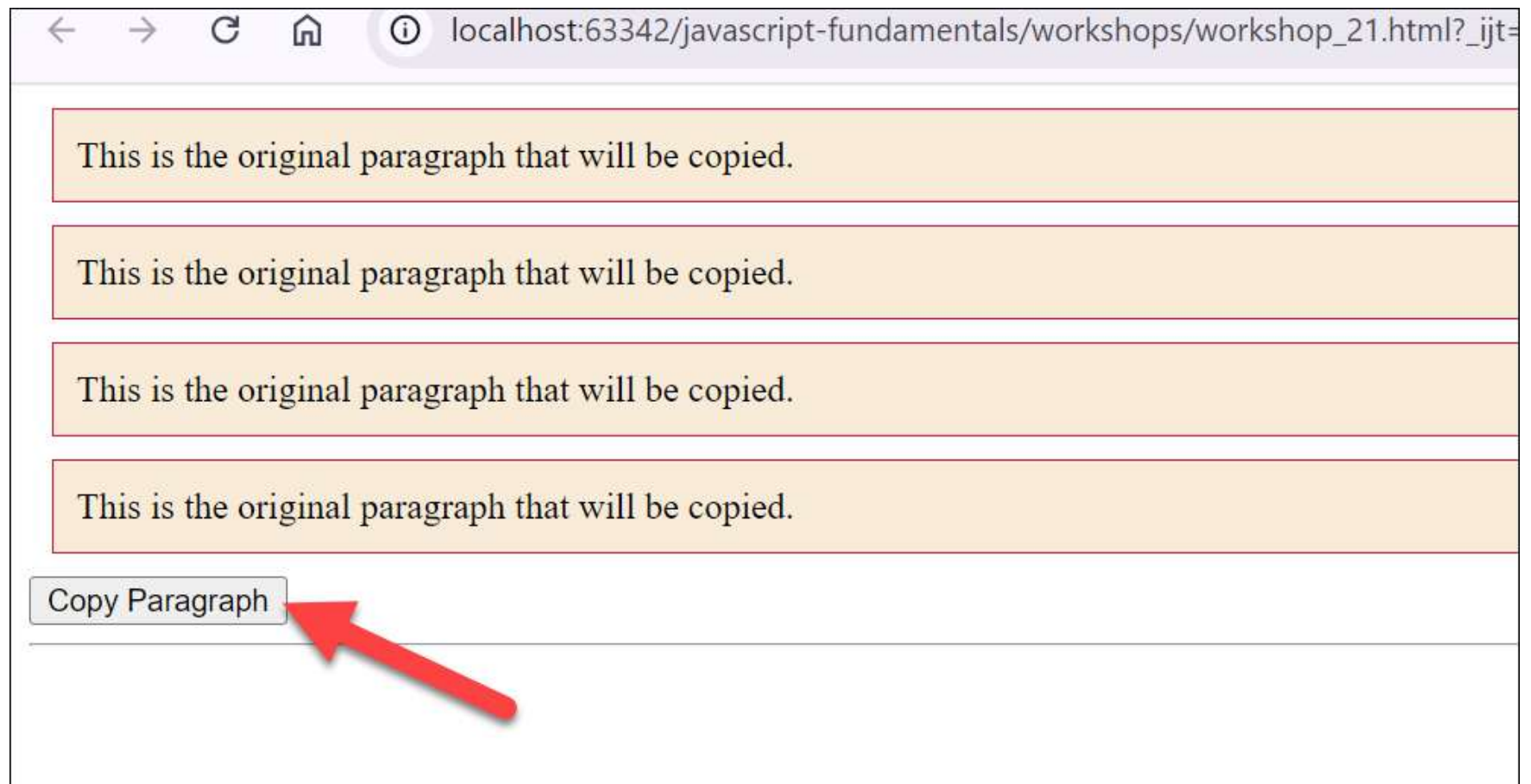


Workshop #21

- Write a script that copies a paragraph and pastes the copy below the original
- use:
 - `document.getElementById('xxx').cloneNode();`
 - Property `nextSibling`
 - Also look at : `insertBefore();`
- Look up documentation yourself!



Sample output



insertAfter?

There is no `insertAfter` method. It can be emulated by combining the `insertBefore` method with `nextSibling`.

In the previous example, `sp1` could be inserted after `sp2` using:

```
1 parentDiv.insertBefore(sp1, sp2.nextSibling);
```

<https://developer.mozilla.org/en-US/docs/Web/API/Node.insertBefore>