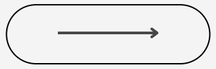
APPLICATION'S STYLES ENCAPSULATION



PRESENTED BY
Priatko Oleh

STRUCTURE

1 MOTIVATION

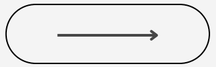
2 IFRAME

3 SHADOW DOM

4 PROS AND CONS

5 SUMMARY

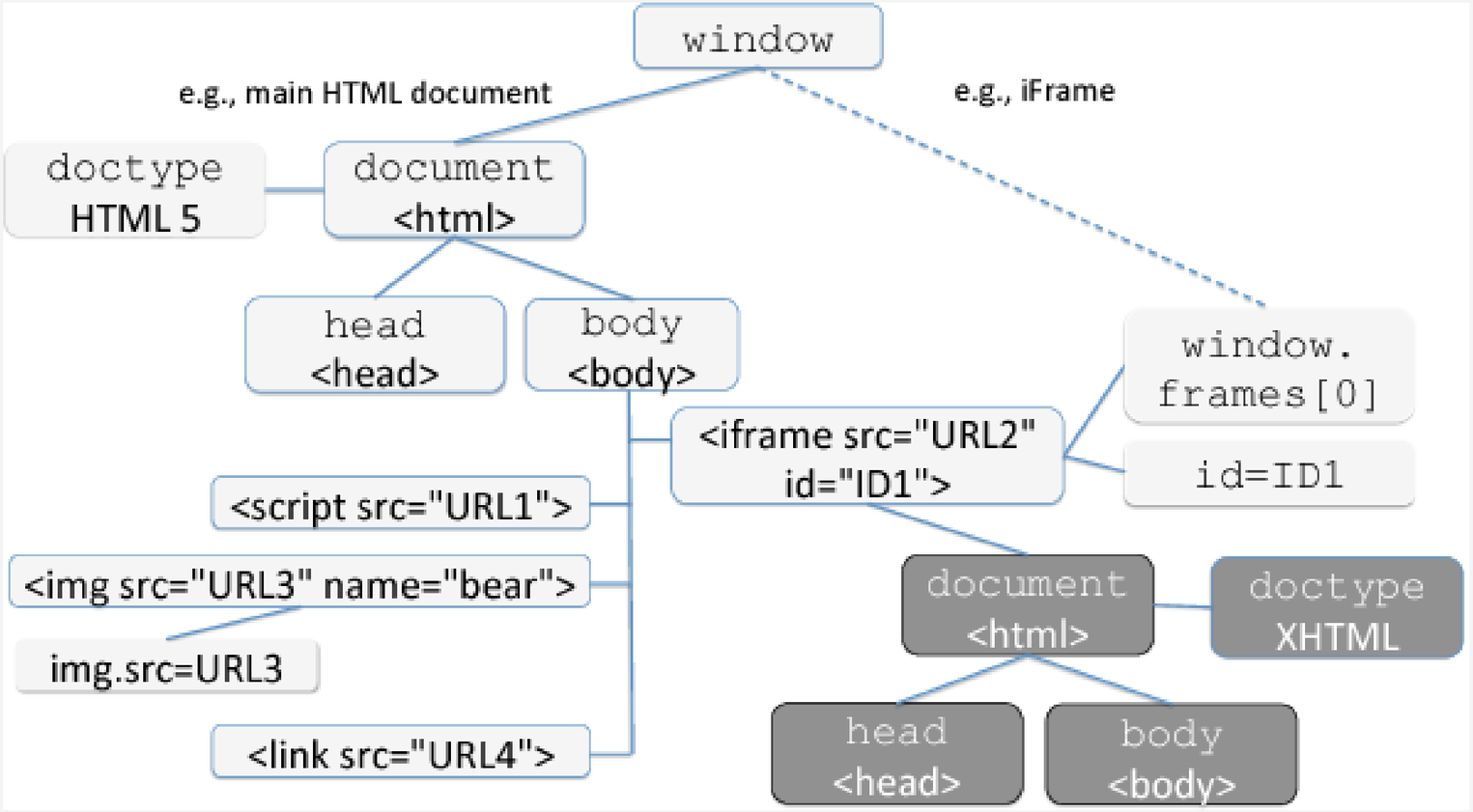
6 LINKS



1 MOTIVATION

- **Style Conflicts.** Style Isolation prevents external styles from affecting the component's internals.
- **Security.** Isolation prevents external scripts from accessing or modifying the internals of a component.
- **Reusability & Modularity.**
- **Libraries.** When an application or a component is going to be used public.

2 IFRAME



What is Iframe ?

An inline frame (iFrame) is an element that loads another HTML element inside of a web page. They are commonly used to embed specific content like external ads, videos, tags, or other interactive elements into the page.

2 IFRAME

When & Why?

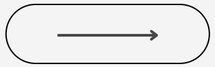
EMBEDDING EXTERNAL CONTENT STANDALONE APPS: WHEN YOU WANT TO EMBED CONTENT FROM ANOTHER WEBSITE INTO YOUR OWN, SUCH AS A VIDEO FROM YOUTUBE, A MAP FROM GOOGLE MAPS, OR A SOCIAL MEDIA POST.

ISOLATING THIRD-PARTY CONTENT: IF YOU'RE INCORPORATING CONTENT FROM A THIRD-PARTY SOURCE, LIKE A WIDGET OR A FORM, USING AN IFRAME CAN ISOLATE ITS CODE FROM YOUR MAIN PAGE'S CODE, REDUCING THE RISK OF CONFLICTS OR SECURITY ISSUES.

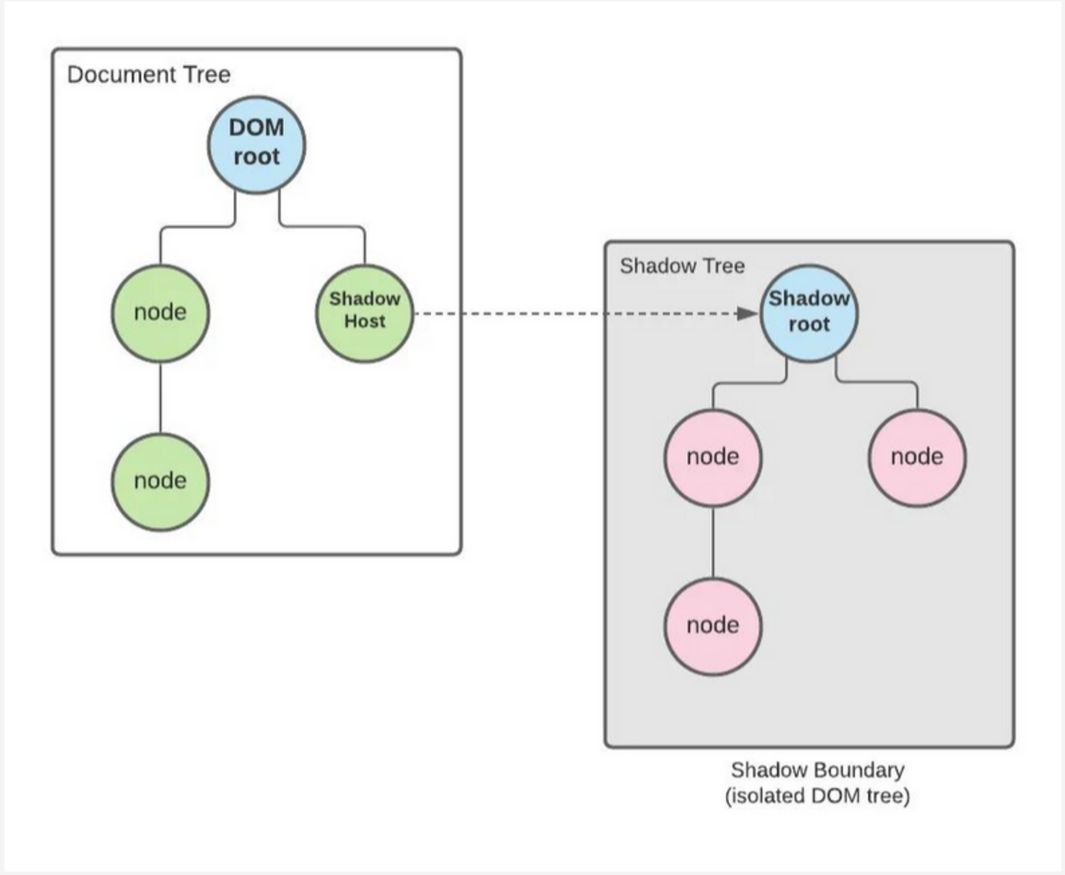
MAINTAINING CONSISTENCY: IF YOU HAVE CONTENT THAT'S STATIC OR SLOW TO UPDATE, EMBEDDING IT VIA AN IFRAME CAN ENSURE THAT IT REMAINS CONSISTENT ACROSS MULTIPLE PAGES OR SITES WITHOUT NEEDING TO UPDATE EACH ONE INDIVIDUALLY.

CROSS-DOMAIN COMMUNICATION: IN SOME CASES, IF YOU NEED TO COMMUNICATE BETWEEN DIFFERENT DOMAINS WHILE ADHERING TO SECURITY RESTRICTIONS (LIKE WITH OAUTH FLOWS FOR AUTHENTICATION), IFRAMES CAN BE USED TO ACHIEVE THIS.

DYNAMIC LOADING: IF YOU HAVE PARTS OF A WEBPAGE THAT YOU WANT TO LOAD DYNAMICALLY WITHOUT RELOADING THE ENTIRE PAGE, IFRAMES CAN BE USED TO LOAD CONTENT ASYNCHRONOUSLY.



3 SHADOW DOM



What is Shadow DOM ?

Shadow DOM serves for encapsulation. It allows a component to have its very own "shadow" DOM tree, that can't be accidentally accessed from the main document, may have local style rules, and more.

There are some bits of shadow DOM terminology to be aware of:

Shadow host: The regular DOM node that the shadow DOM is attached to.
Shadow tree: The DOM tree inside the shadow DOM.
Shadow boundary: the place where the shadow DOM ends, and the regular DOM begins.
Shadow root: The root node of the shadow tree.

3 SHADOW DOM

HTML

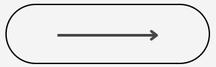
```html
<div id="host">
  <template shadowrootmode="open">
    <span>I'm in the shadow DOM</span>
  </template>
</div>
```

JS

```js
const host = document.querySelector("#host");
const shadow = host.attachShadow({ mode: "open" });
const span = document.createElement("span");
span.textContent = "I'm in the shadow DOM";
shadow.appendChild(span);
```

Result

```
▼ <div id="host">
  ▼ #shadow-root (open)
      <span>I'm in the shadow DOM</span>
  </div>
```



3 SHADOW DOM

```html
<div id="host">
    <template shadowrootmode="open">
        <span>I'm in the shadow DOM</span>
    </template>
</div>
<script>
  console.log(document.getElementById('host').shadowRoot.querySelector('span'));
</script>
```

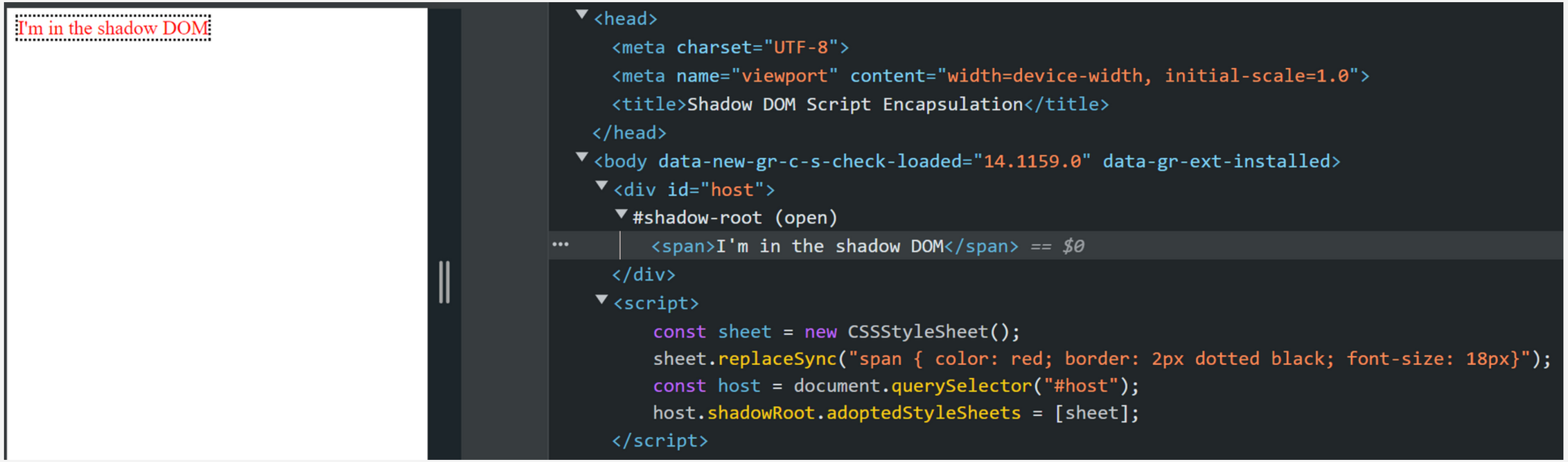
```
<span>I'm in the shadow DOM</span>
```

Console

```html
<div id="host">
    <template shadowrootmode="closed">
        <span>I'm in the shadow DOM</span>
    </template>
</div>
<script>
  console.log(document.getElementById('host').shadowRoot.querySelector('span'));
</script>
```

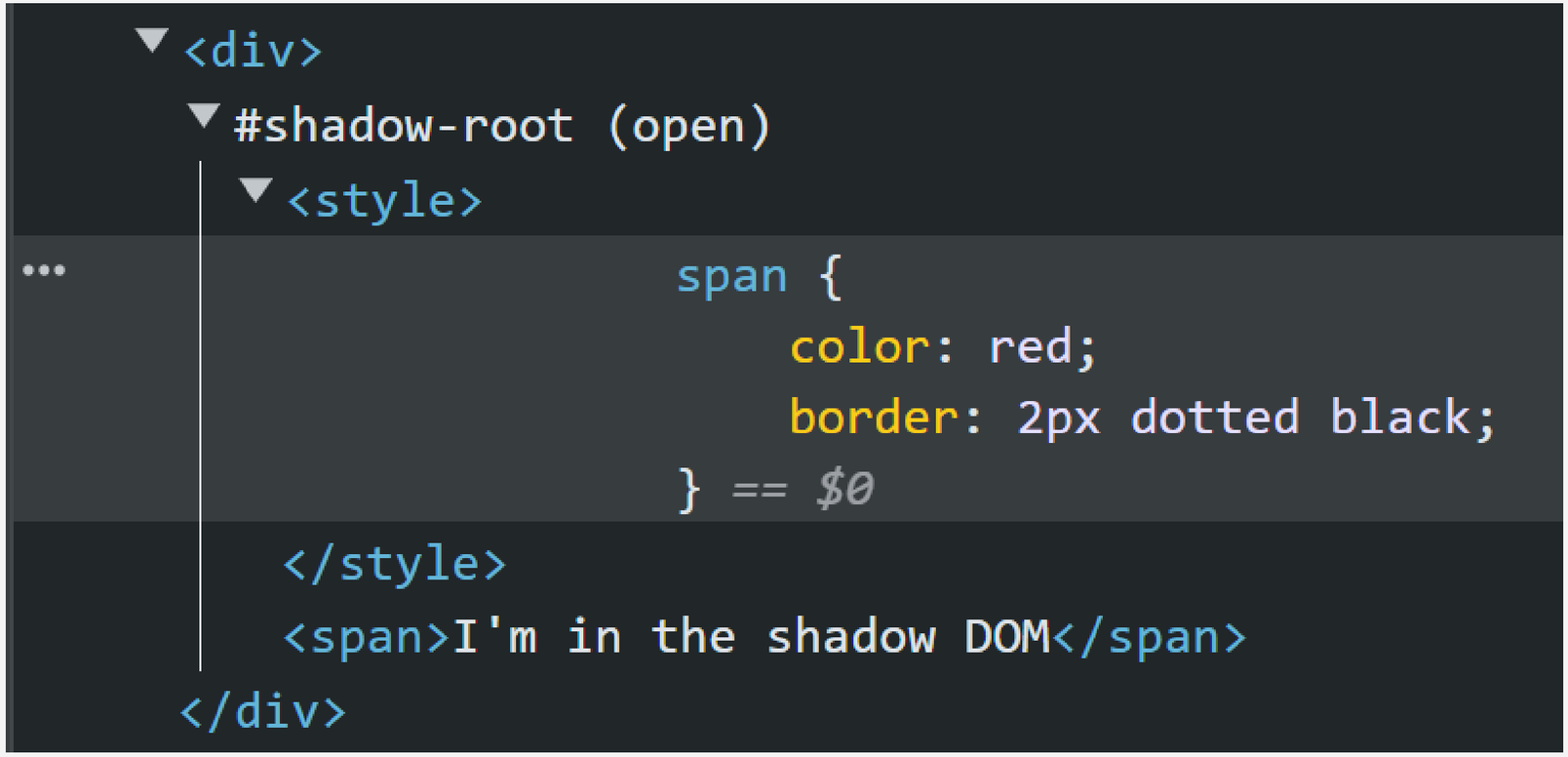
```
▶ Uncaught TypeError: Cannot read properties of null (reading 'querySelector')
    at index.html?_ijt=80dk…ELOAD_ON_SAVE:16:57
```

3 SHADOW DOM



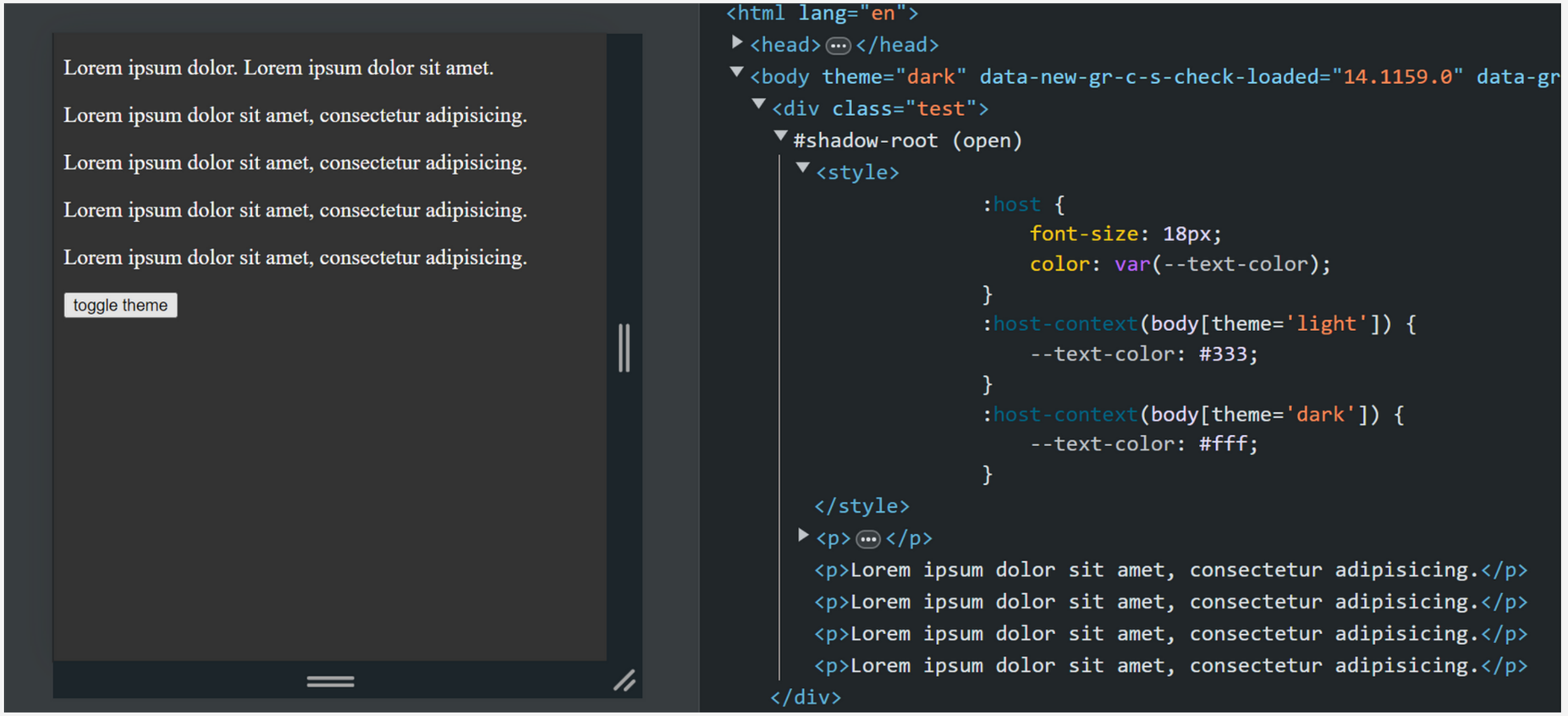
CREATING A CSSSTYLESHEET AND ASSIGNING IT TO THE SHADOW ROOT USING ADOPTEDSTYLESHEETS ALLOWS YOU TO CREATE A SINGLE STYLESHEET AND SHARE IT AMONG MANY DOM TREES. FOR EXAMPLE, A COMPONENT LIBRARY COULD CREATE A SINGLE STYLESHEET AND THEN SHARE IT AMONG ALL THE CUSTOM ELEMENTS BELONGING TO THAT LIBRARY. THE BROWSER WILL PARSE THAT STYLESHEET ONCE. ALSO, YOU CAN MAKE DYNAMIC CHANGES TO THE STYLESHEET AND HAVE THEM PROPAGATE TO ALL COMPONENTS THAT USE THE SHEET.

3 SHADOW DOM



THE APPROACH OF ATTACHING A <STYLE> ELEMENT IS GREAT IF YOU WANT TO BE DECLARATIVE, HAVE FEW STYLES, AND DON'T NEED TO SHARE STYLES ACROSS DIFFERENT COMPONENTS.

3 SHADOW DOM


Lorem ipsum dolor. Lorem ipsum dolor sit amet.
Lorem ipsum dolor sit amet, consectetur adipisicing.
Lorem ipsum dolor sit amet, consectetur adipisicing.
Lorem ipsum dolor sit amet, consectetur adipisicing.
Lorem ipsum dolor sit amet, consectetur adipisicing.
toggle theme
<html lang="en">
<head> </head>
<body theme="dark" data-new-gr-c-s-check-loaded="14.1159.0" data-gr

#shadow-root (open)
<style>
:host {
font-size: 18px;
color: var(--text-color);
}
:host-context(body[theme='light']) {
--text-color: #333;
}
:host-context(body[theme='dark']) {
--text-color: #fff;
}
</style>

Lorem ipsum dolor sit amet, consectetur adipisicing.
Lorem ipsum dolor sit amet, consectetur adipisicing.
Lorem ipsum dolor sit amet, consectetur adipisicing.
Lorem ipsum dolor sit amet, consectetur adipisicing.

3 SHADOW DOM

Shadow Root !== HTMLElement

~~insertAdjacentElement()~~

~~insertAdjacentHTML()~~

~~insertAdjacentText()~~

~~get dataset()~~

~~get id()~~

~~get classlist()~~

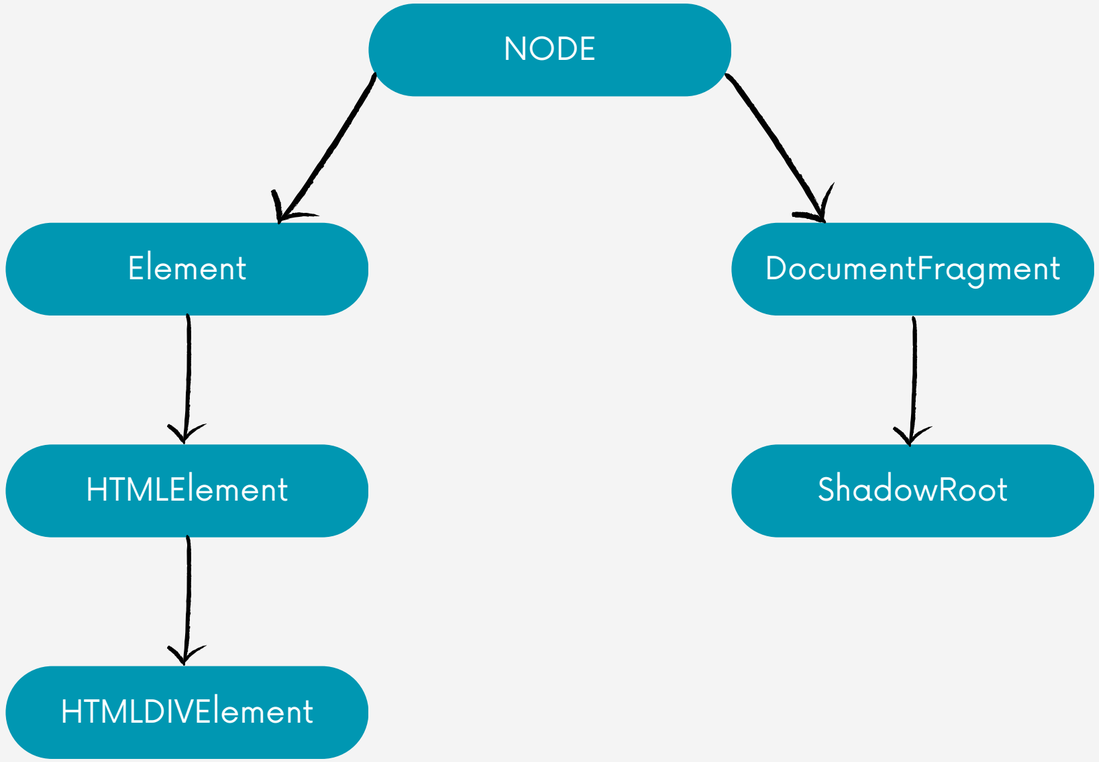
~~getBoundingClientRect()~~

~~remove()~~

~~cloneNode()~~

~~onCallbacks()~~

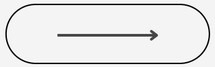
....



3 SHADOW DOM

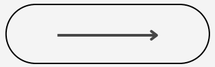
```
<div class="test">
    <template shadowrootmode="open">
        <style>
            @font-face {
                font-family: "Avenir Next LT Pro";
                font-style: normal;
                font-weight: 400;
                src: url(https://assets.leasehawk.com/fonts/avenir/35BCC0_4_0.woff2) format('woff2');
                src: url(https://assets.leasehawk.com/fonts/avenir/35BCC0_4_0.woff) format('woff');
            }
            :host {
                font-size: 18px;
                font-family: "Avenir Next LT Pro", serif;
            }
        </style>
```

https://issues.chromium.org/issues/41085401



3 SHADOW DOM

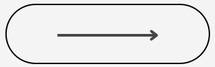
```
<div class="test">
    <style>
        @font-face {
            font-family: "Avenir Next LT Pro";
            font-style: normal;
            font-weight: 400;
            src: url(https://assets.leasehawk.com/fonts/avenir/35BCC0_4_0.woff2) format('woff2');
            src: url(https://assets.leasehawk.com/fonts/avenir/35BCC0_4_0.woff) format('woff');
        }
    </style>
    <template shadowrootmode="open">
        <style>
            :host {
                font-size: 18px;
                font-family: "Avenir Next LT Pro", serif;
            }
        </style>
```



3 SHADOW DOM

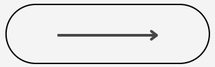
Incorporate styles from styled-components to shadow dom

```
<StyleSheetManager target={getAppRootElement().shadowRoot as ShadowRoot}>
  <>
    <GlobalStyles />
    <ShadowRootElementStyles isChatOpened={isChatOpened} isChatVisible={isChatVisible} />
    {isChatOpened ? chatElement : <ChatGreeting />}
  </>
</StyleSheetManager>
```



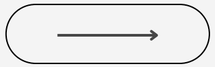
5 PROS & CONS IFRAME

PROS	CONS
Isolation: iframes provide strong isolation between the parent document and the content within the iframe. This means CSS styles and JavaScript from the parent page won't affect the content of the iframe and vice versa.	**Performance Overhead**: Each iframe comes with its own overhead in terms of memory and processing. This can affect the performance of the overall application, especially if many iframes are used.
Security: Due to the isolation, iframes can be used to embed content from untrusted sources without risking the security of the main page.	**Complexity**: Managing communication between the parent document and the content within iframes can be complex, especially in large applications.
Cross-Domain Communication: iframes can facilitate communication between different domains using techniques like postMessage.	**Styling Limitations**: CSS styles applied within an iframe do not affect the parent document, and vice versa. This can lead to inconsistencies in styling unless carefully managed.



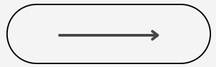
5 PROS & CONS SHADOW DOM

PROS	CONS
Scoped Styling: Shadow DOM allows encapsulation of CSS styles within a component, preventing styles from leaking out and affecting other parts of the document. This helps in building modular and maintainable components.	**Browser Support**: Support for Shadow DOM across different browsers might not be consistent, although it has been improving over time.
DOM Encapsulation: Similar to CSS, Shadow DOM encapsulates the DOM structure of a component, preventing outside code from accessing or modifying the internal structure. This enhances security and reduces the risk of unintended side effects.	**Complexity**: Implementing and managing Shadow DOM can be more complex compared to iframes, especially for developers who are not familiar with web components.
Performance: Shadow DOM typically has less performance overhead compared to iframes since it doesn't create separate browsing contexts.	**Limited Isolation**: While Shadow DOM provides encapsulation, it doesn't offer the same level of isolation as iframes. JavaScript code in the parent document can still access the Shadow DOM of a component if not properly secured.



5 SUMMARY

IN SUMMARY, THE CHOICE BETWEEN IFRAMES AND SHADOW DOM DEPENDS ON FACTORS SUCH AS THE **LEVEL OF ISOLATION** REQUIRED, **PERFORMANCE** CONSIDERATIONS, AND **THE COMPLEXITY OF IMPLEMENTATION**. IF **STRONG ISOLATION** AND **CROSS-DOMAIN COMMUNICATION** ARE NEEDED, IFRAMES MIGHT BE PREFERABLE. ON THE OTHER HAND, IF YOU'RE BUILDING **MODULAR COMPONENTS** WITH **SCOPED STYLING**, SHADOW DOM COULD BE A BETTER CHOICE.



5 USEFUL LINKS

https://issues.chromium.org/issues/41085401

https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_shadow_DOM

https://github.com/dofamine/shadow-dom/blob/main/index.html

https://javascript.info/shadow-dom

Thank you!