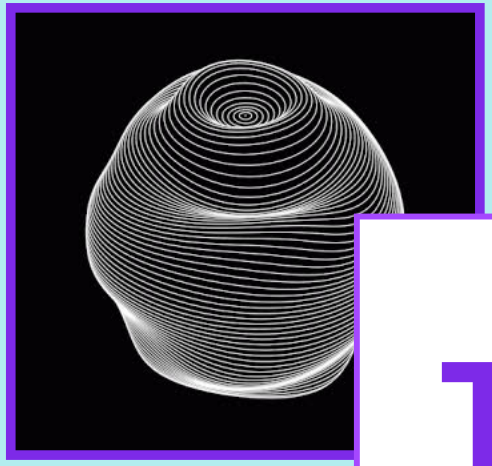
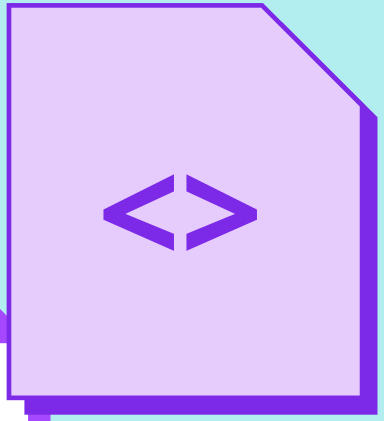
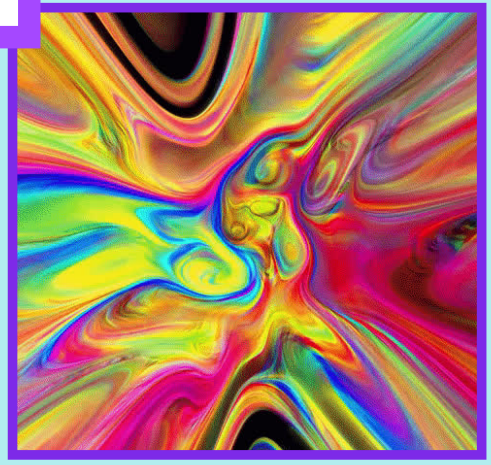


The DOM



Presented by
Anna Azzam

Canva



Overview

- What is the DOM?
- Data types of DOM Elements
- Reading the DOM in JS
- Modifying the DOM in JS
- Reading and modifying live example
- Scrolling live example

HTML

Markup language creating web documents

CSS

Style sheet language for applying styles to a document

JavaScript

Scripting language to make your web page dynamic

What is the DOM?

The DOM (Document Object Model) is an interface that allows JavaScript to interact with HTML through the browser.

HTML

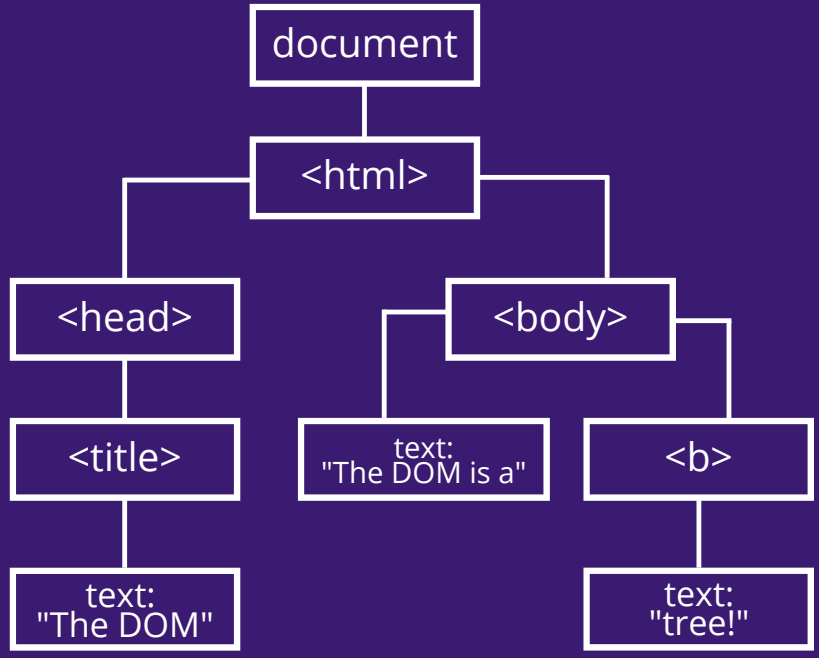
CSS

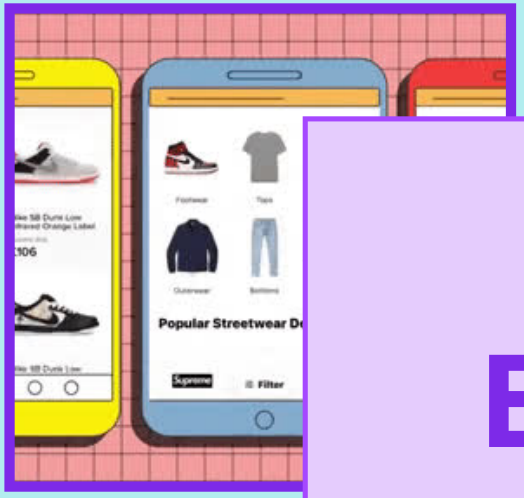
JavaScript

The DOM interface allows JavaScript to access and update the content, structure, and style of a document

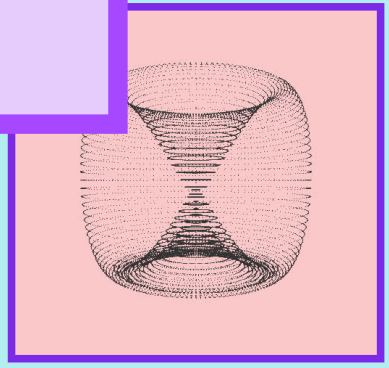
Tree-like structure of the DOM

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>The DOM</title>
  </head>
  <body>
    The DOM is a <b>tree!</b>
  </body>
</html>
```





DOM Elements and Data Types



DOM Data Types

To understand how the DOM is represented, we will introduce some new data types

Document

The type of the `document` object. Represents the root of the entire DOM.

Element

A node in the DOM tree. Objects of this type implement an interface that allows interacting with the document.

NodeList

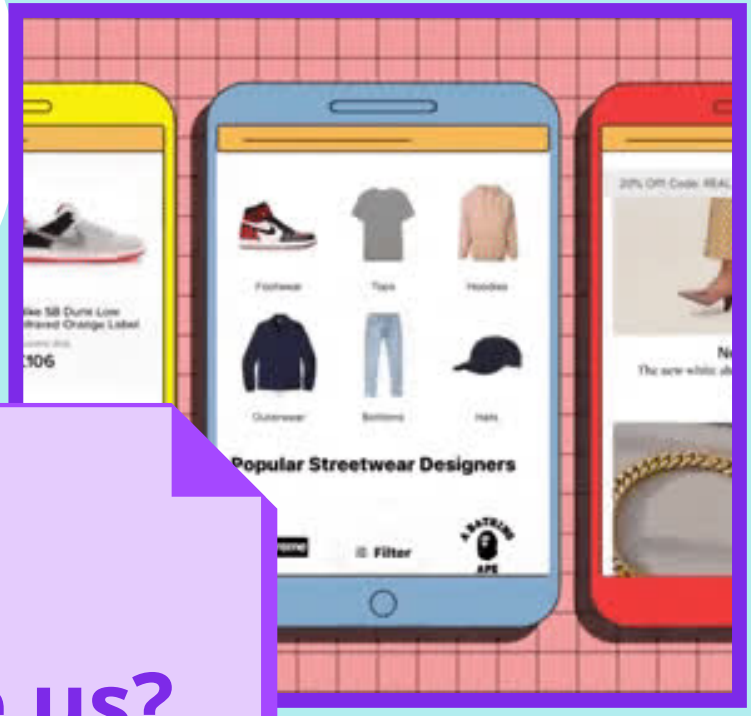
An array of elements, like the kind that is returned by the method.

Understanding DOM Elements

- Element is the base class for all types of objects in the Document
- Different HTML tags/elements correspond to different Element types in JS
- Different Element types include:

```
HTMLInputElement, HTMLSpanElement, HTMLDivElement,
HTMLScriptElement, HTMLHeadingElement,HTMLImageElement....
```

Q



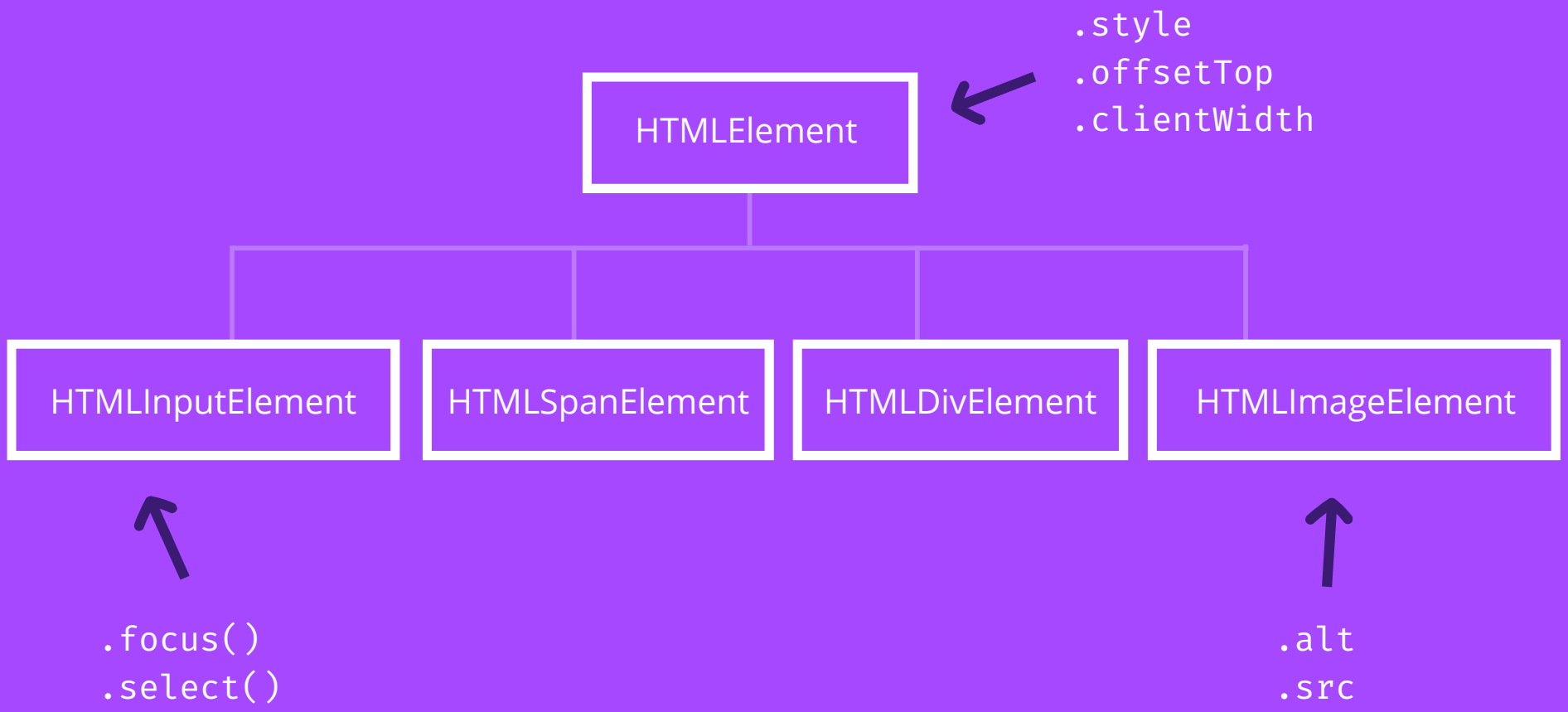
What does the Element interface give us?

Ways of *reading* things

e.g. Get the size, position, color or text of an element

Ways of *writing* things

e.g. Set an attribute, change the styling of an element


HTMLElement
.style
.offsetTop
.clientWidth
HTMLInputElement
HTMLSpanElement
HTMLDivElement
HTMLImageElement
.focus()
.select()
.alt
.src

HTMLElement

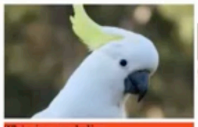
The HTMLElement interface represents any HTML element. Some elements directly implement this interface, while others implement it via an interface that inherits it.

MDN Web Docs

Reading the DOM

```
// Returns an html element with the given id
document.getElementById(id);

// Returns a DOM HTMLCollection of all matches
document.getElementsByTagName(name); eg: div
document.getElementsByClassName(classname);

// Returns the first Element that matches the selector
document.querySeletor(query);
```

This is an example webpage



This is a red div

This is a blue div

VIDEOS

Elements | Console | Lighthouse | Sources | Network | Performance

top | Filter | All levels

```
> document.getElementById('puppy')
<- <img style="width: 200px" id="puppy" src="https://hips.hearstapps.com/countryliving.cdnds.net/17/47/1511194376-cavachon-puppy-christmas.jpg">
> document.getElementsByClassName('box')
<- HTMLCollection(2) [div.box, div.box]
    0: div.box
    1: div.box
    length: 2
    __proto__: HTMLCollection
> document.getElementsByTagName('img')
<- HTMLCollection(3) [img, img, img#puppy, puppy: img#puppy]
    0: img
    1: img
    2: img#puppy
    length: 3
    puppy: img#puppy
    __proto__: HTMLCollection
> document.querySelector('#puppy')
<- <img style="width: 200px" id="puppy" src="https://hips.hearstapps.com/countryliving.cdnds.net/17/47/1511194376-cavachon-puppy-christmas.jpg">
> document.querySelector('.box')
<- <div class="box" style="background-color: red; width: 200px; height: 200px;">
        This is a red div
    </div>
> document.querySelectorAll('.box')
<- NodeList(2) [div.box, div.box]
    0: div.box
    1: div.box
    length: 2
    __proto__: NodeList
>
```

Reading the DOM example

Writing to the DOM

```
// Create a new div element
let element = document.createElement("div");
// Create a new text node
let textNode = document.createTextNode("Some text");

// Adding and removing elements
element.appendChild(textNode);
element.removeChild(textNode);

// Making changes to attributes
button.setAttribute("disabled", "");
```

Changing the style of an element

An element has a "style" property which corresponds to the "style" attribute of the HTML element.

This can be modified in the JavaScript.

```
// Changing element.style
element.style.left = "50px"; // Note: don't forget units!

// Adding 5px to the left value
let newLeft = parseInt(element.style.left, 10) + 5 + "px";
element.style.left = newLeft;

element.style.backgroundColor = "red"; // Note: camelCase
```

Getting the style of an element

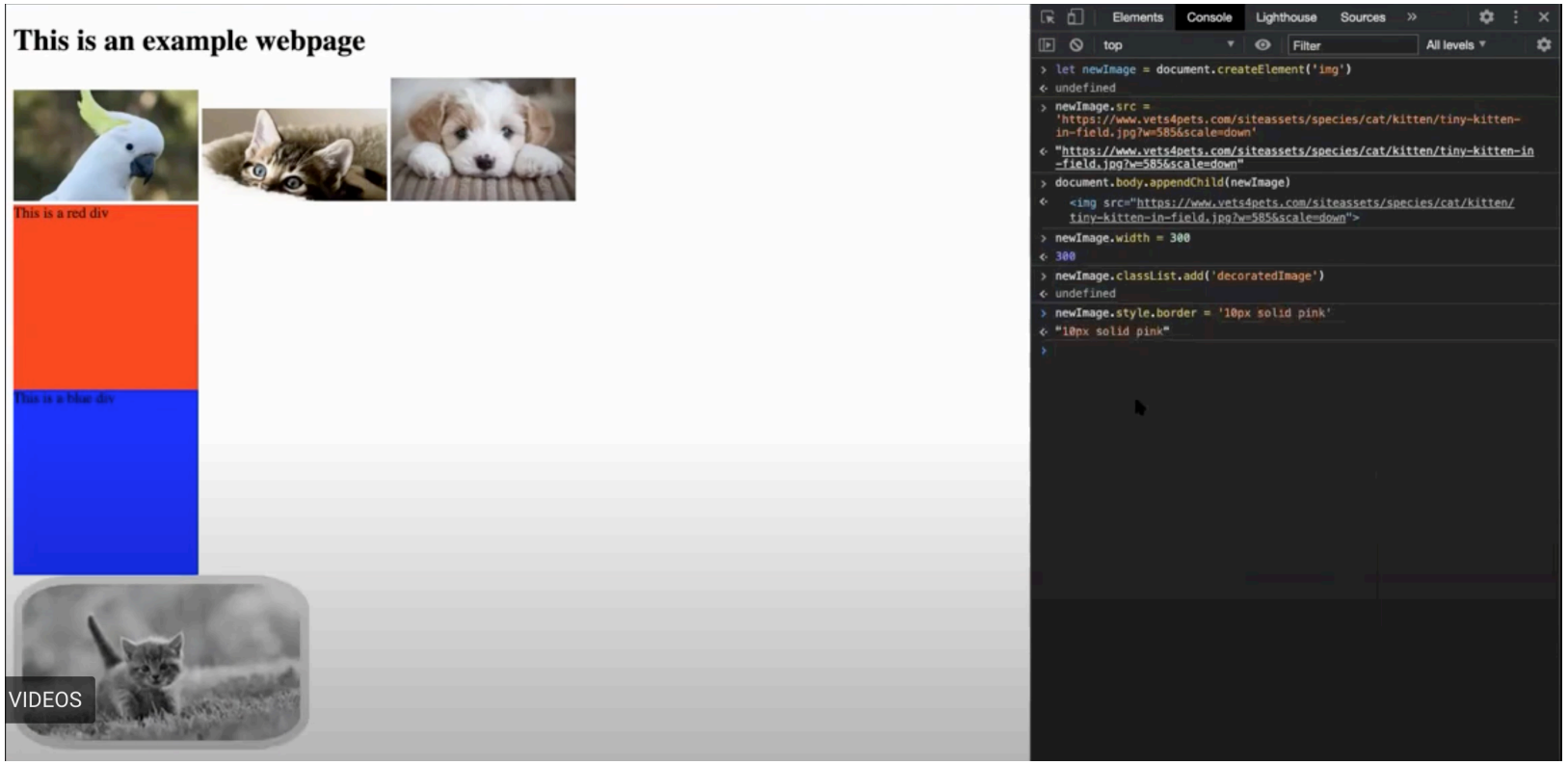
Note: `element.style.left` will only be present on an element if the `left` property was set in inline styles, or by scripting (not if it was set in CSS).

```
// Getting computed style
let computedStyle = window.getComputedStyle(element, null)
let bgColor = computedStyle.getPropertyValue("background-
color")
```

Changing the classes of an element

Another way of modifying the style of the element is to change the classnames which exist on the element. This can be done using the "classList" property.

```
// Changing element.classList
element.classList.add("class");
element.classList.remove("class");
element.classList.toggle("class");
element.classList.contains("class"); // returns true if
class exists on element
```

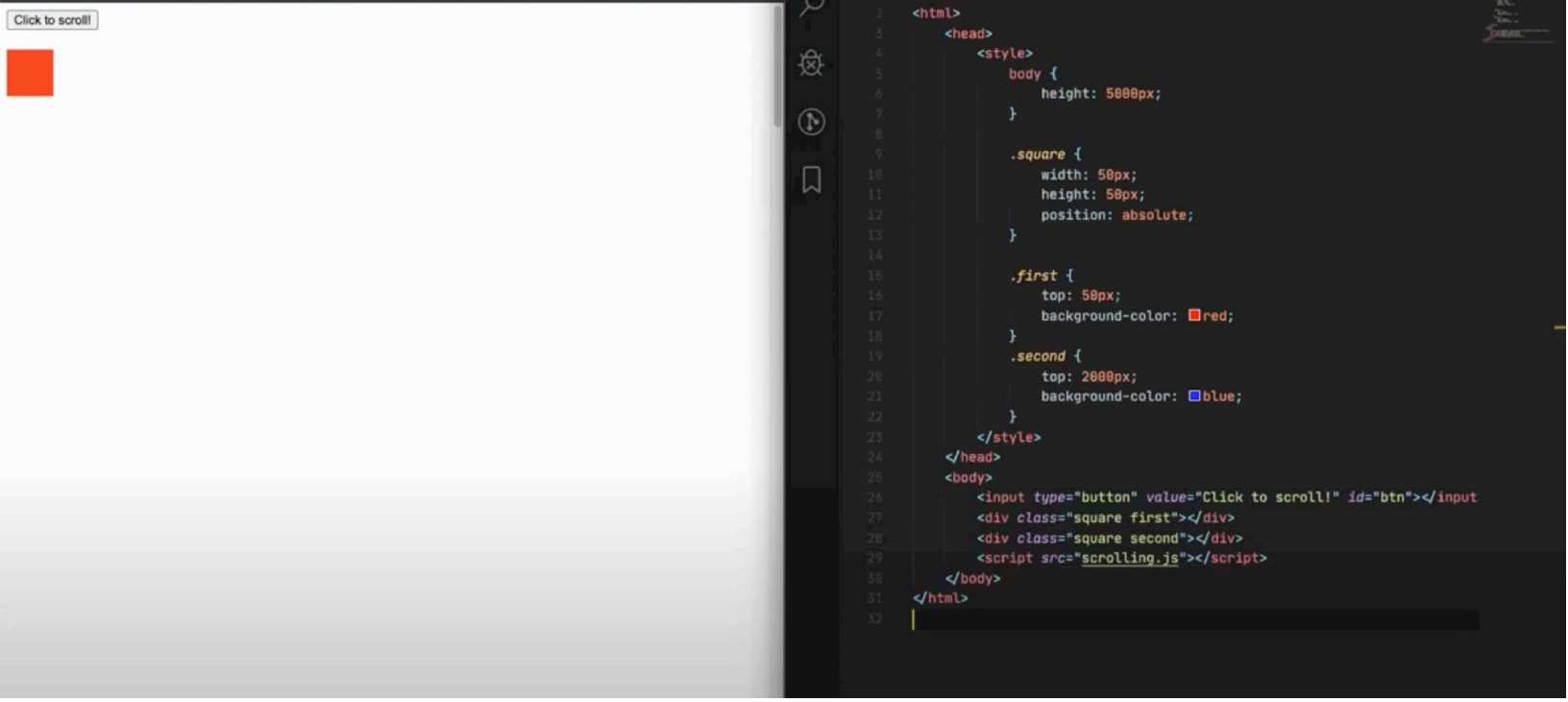

This is an example webpage
This is a red div
This is a blue div
VIDEOS
Elements
Console
Lighthouse
Sources
top
Filter
All levels
let newImage = document.createElement('img')
undefined
newImage.src = 'https://www.vets4pets.com/siteassets/species/cat/kitten/tiny-kitten-in-field.jpg?w=585&scale=down'
"https://www.vets4pets.com/siteassets/species/cat/kitten/tiny-kitten-in-field.jpg?w=585&scale=down"
document.body.appendChild(newImage)
<img src="https://www.vets4pets.com/siteassets/species/cat/kitten/tiny-kitten-in-field.jpg?w=585&scale=down">
newImage.width = 300
300
newImage.classList.add('decoratedImage')
undefined
newImage.style.border = '10px solid pink'
"10px solid pink"

Writing to the DOM example

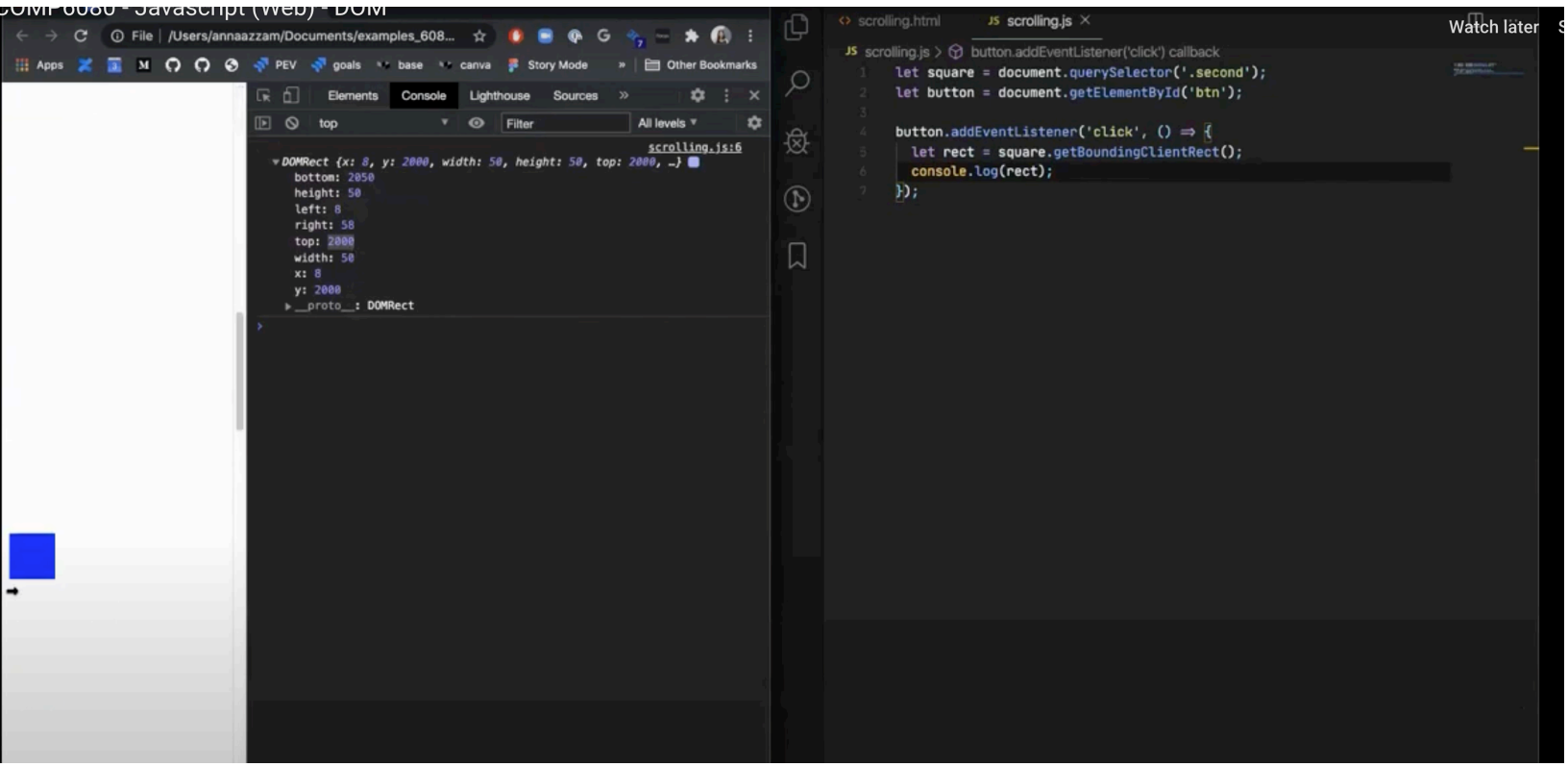
Scrolling

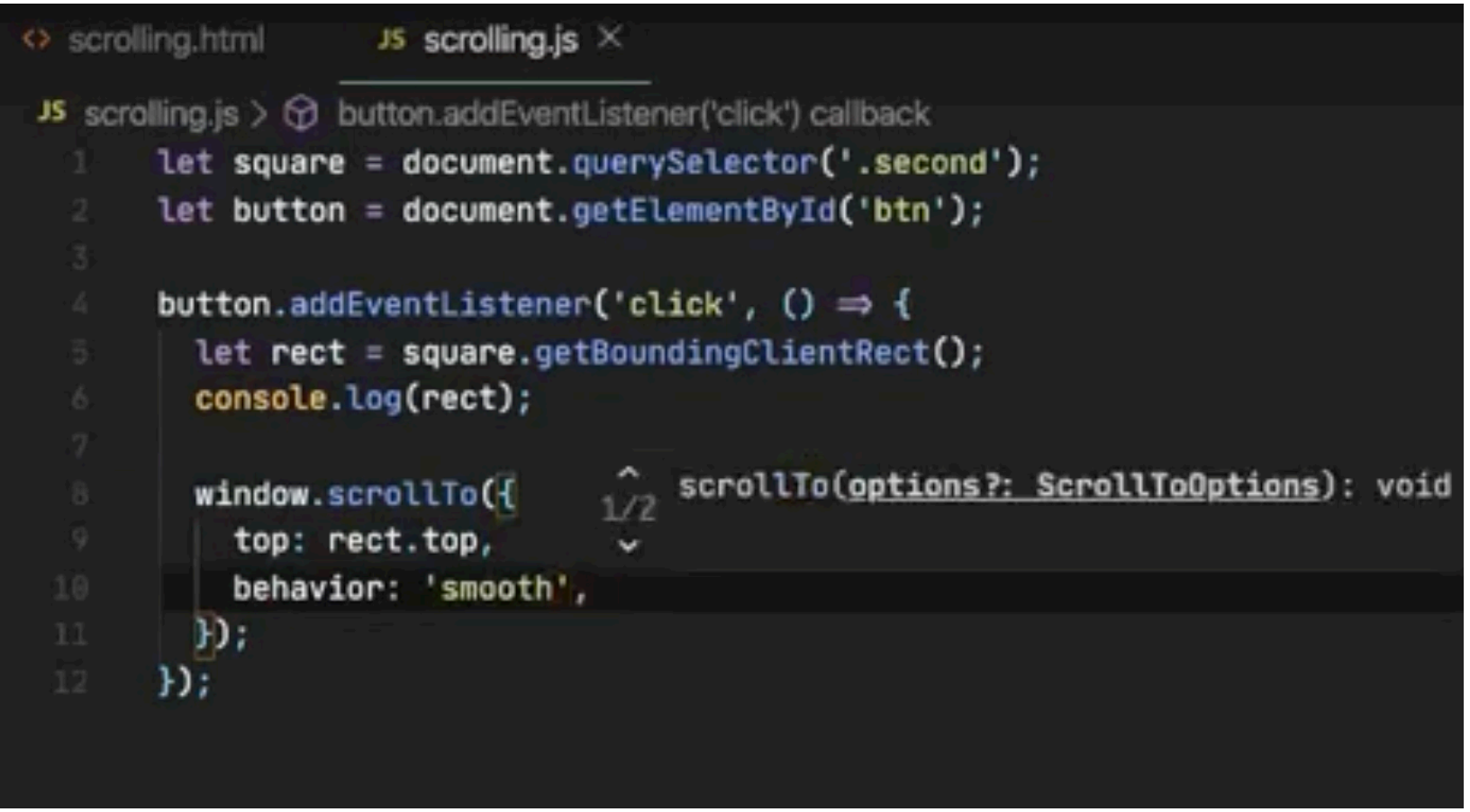
```
// Get the current scroll position of the page
console.log(window.scrollX);
console.log(window.scrollY);

// Scroll to a position on the page:
window.scrollTo({
  top: 100,
  left: 0,
  behavior: "smooth",
});
```

Scrolling to an element example


Click to scroll!
<html>
<head>
<style>
body {
height: 5000px;
}
.square {
width: 50px;
height: 50px;
position: absolute;
}
.first {
top: 50px;
background-color: red;
}
.second {
top: 2000px;
background-color: blue;
}
</style>
</head>
<body>
<input type="button" value="Click to scroll!" id="btn"></input

<script src="scrolling.js"></script>
</body>
</html>


DOM 6080 - Javascript (Web) - DOM
Watch later
File | /Users/annaazzam/Documents/examples_608...
Apps PEV goals base canva Story Mode Other Bookmarks
Elements Console Lighthouse Sources
top Filter All levels
scrolling.js:6
DOMRect {x: 8, y: 2000, width: 50, height: 50, top: 2000, …}
bottom: 2050
height: 50
left: 8
right: 58
top: 2000
width: 50
x: 8
y: 2000
__proto__: DOMRect
scrolling.html
scrolling.js
scrolling.js > button.addEventListener('click') callback
let square = document.querySelector('.second');
let button = document.getElementById('btn');
button.addEventListener('click', () => {
let rect = square.getBoundingClientRect();
console.log(rect);
});


scrolling.html
scrolling.js
scrolling.js > button.addEventListener('click') callback
let square = document.querySelector('.second');
let button = document.getElementById('btn');
button.addEventListener('click', () => {
let rect = square.getBoundingClientRect();
console.log(rect);
window.scrollTo({
top: rect.top,
behavior: 'smooth',
});
});
1/2
scrollTo(options?: ScrollToOptions): void