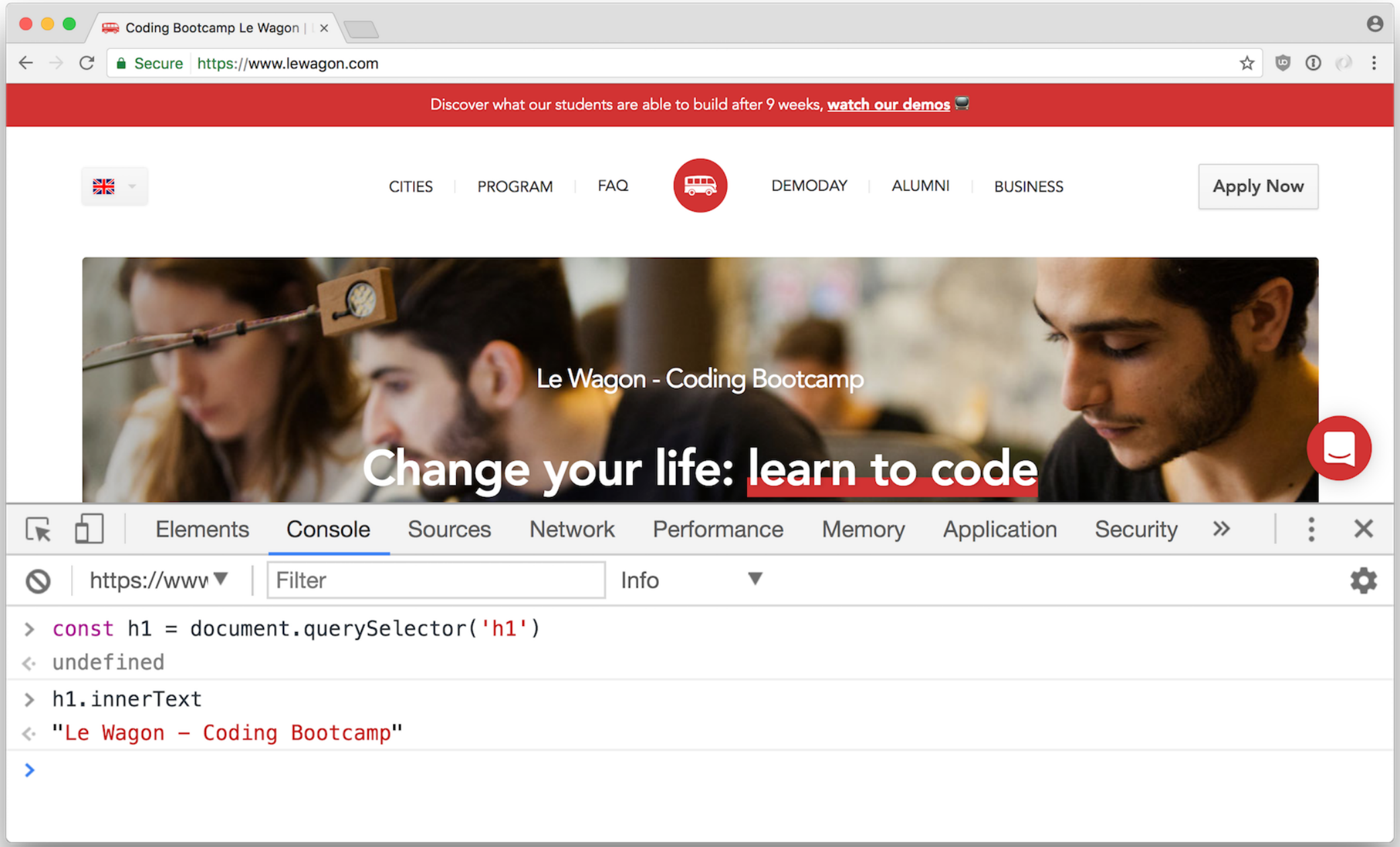
Let's open up the browser

Your browser is not just a browser, it's an IDE


Coding Bootcamp Le Wagon
Secure https://www.lewagon.com
Discover what our students are able to build after 9 weeks, watch our demos
CITIES
PROGRAM
FAQ
DEMODAY
ALUMNI
BUSINESS
Apply Now
Le Wagon - Coding Bootcamp
Change your life: learn to code
Elements
Console
Sources
Network
Performance
Memory
Application
Security
https://www
Filter
Info
> const h1 = document.querySelector('h1')
<- undefined
> h1.innerText
<- "Le Wagon – Coding Bootcamp"
>

Using JavaScript with HTML

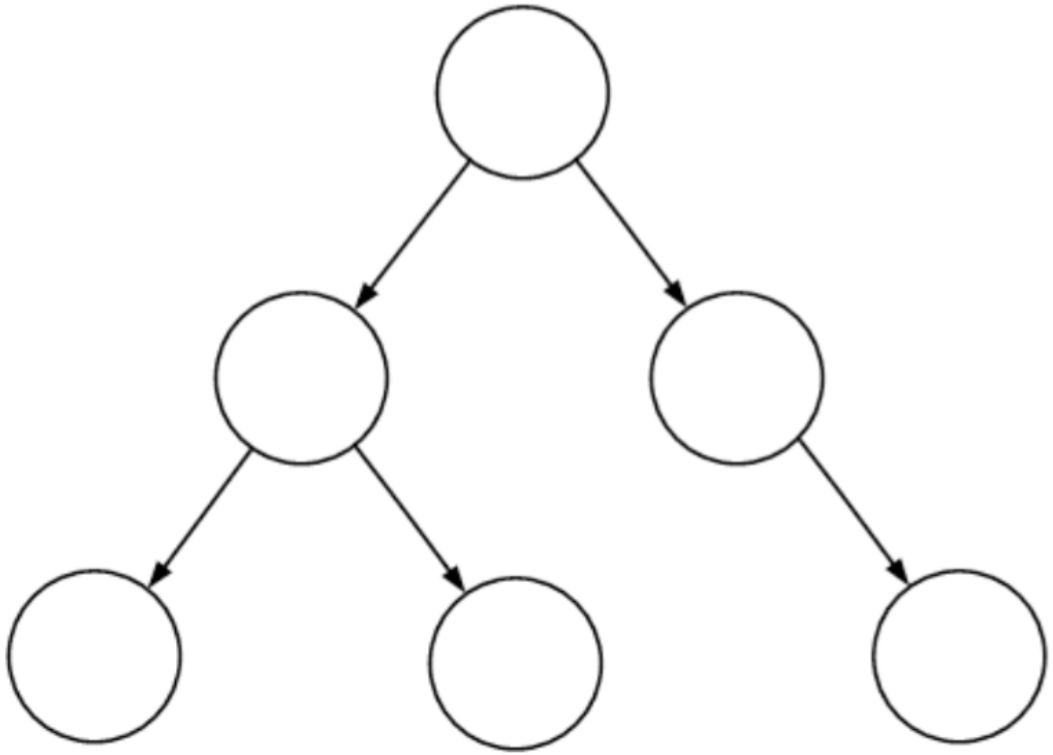
```html
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>
    <!-- Your content -->

    <!-- Your script tags here -->
    <!-- <script src="first_file.js"></script> -->
    <!-- <script src="other_file.js"></script> -->
  </body>
</html>
```

Loading a JS file is **blocking** the page rendering => Put it at the end.

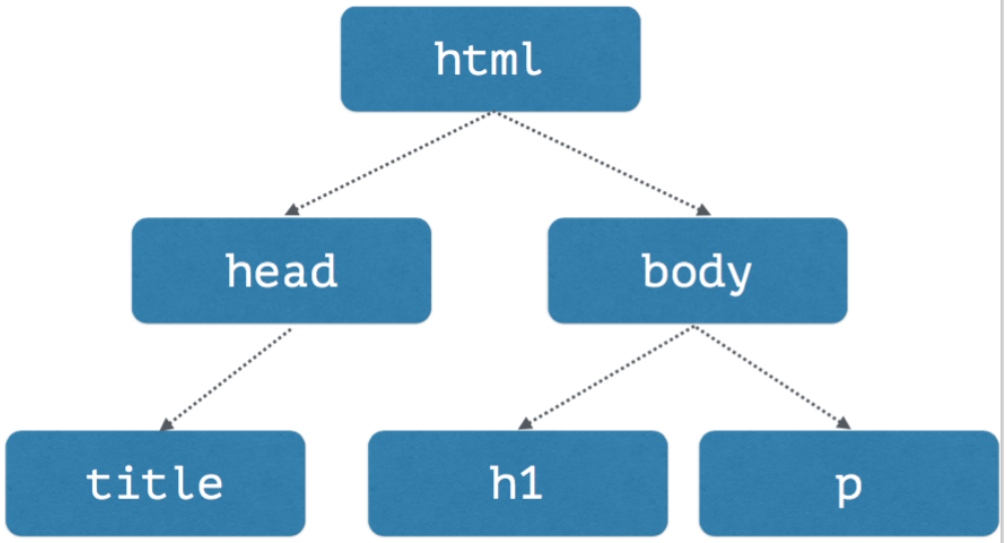
DOM

Document Object Model

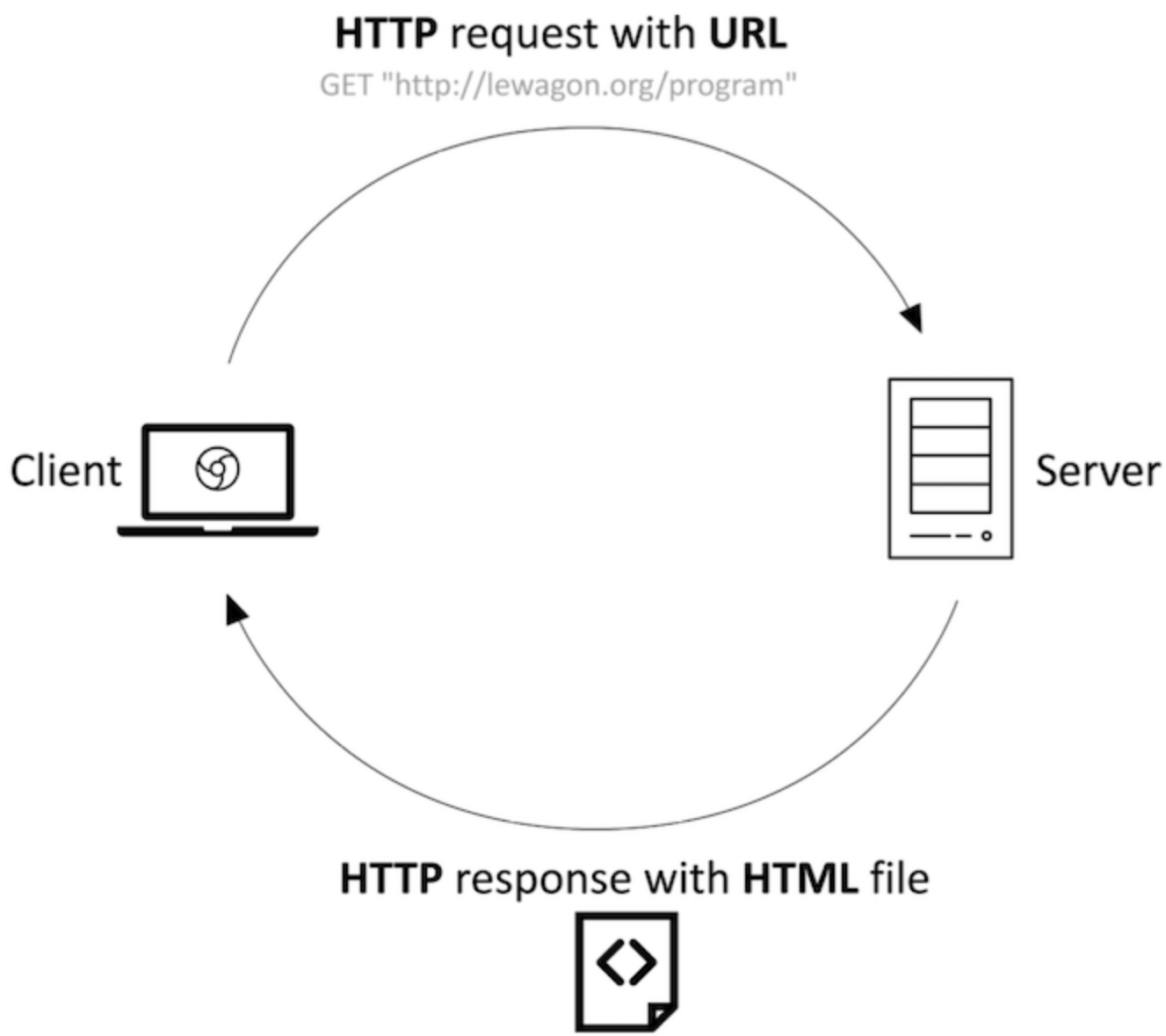
wikipedia.org/Tree*(data*structure)



```html
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <h1>
      Hello
    </h1>
    <p>
      Lorem Ipsum..
    </p>
  </body>
</html>
```




HTTP request with URL
GET "http://lewagon.org/program"
Client
Server


HTTP request with URL
GET "http://lewagon.org/program"
Client
Server
HTTP response with HTML file

The browser **parses** the HTML response and **creates** the DOM from it.

You can visualize the DOM in the Chrome Inspector, in **Elements** tab.

Hello

file:///Users/cb/Desktop/exemple.html

Hello

Lorem Ipsum...

Elements Network Sources Timeline Profiles Resources Audits Console

```
<!DOCTYPE html>
▼<html>
  ▼<head>
     <title>Hello</title>
   </head>
  ▼<body>
     <h1>
           Hello
         </h1>
     <p>
           Lorem Ipsum...
         </p>
   </body>
 </html>
```

html body h1

Styles Computed Event Listeners »

```
element.style {
}
```

```
h1 {                                  user agent stylesheet
   display: block;
   font-size: 2em;
   -webkit-margin-before: 0.67em;
   -webkit-margin-after: 0.67em;
   -webkit-margin-start: 0px;
   -webkit-margin-end: 0px;
   font-weight: bold;
}
```

margin 21.440

border -

Find in Styles

Reference

Please keep a tab open with the DOM Documentation

Interacting with the DOM

The most important method

```
document.querySelector(CSS_SELECTOR);
```

Selecting an element with an `id`

```
<ul id="players"></ul>
```

```
const list = document.querySelector("#players"); // CSS id selector
```

```
// or
const list = document.getElementById("players"); // just the ID name
```

🤔 What about elements with no `id`?

Basic CSS selectors

Reminder

```
p                /* Type selector  */
.red             /* Class selector */
#players         /* ID selector    */
```

Advanced CSS selectors

Reminder

```
ul.active       /* Combined selectors */
ul .active      /* Descending combinator */
ul > .active    /* Child combinator */
```

Combine them to get **specific** CSS selectors:

```
let list = document.querySelector('ul#players > .active');`
```

We've just selected an element! 💪

What can we do now? 🤔

Append content

We are using the Element#insertAdjacentHTML method.

```
// element.insertAdjacentHTML("position", "HTML String")
list.insertAdjacentHTML("beforeend", "<li>Luke</li>");
list.insertAdjacentHTML("beforeend", "<li>Anakin</li>");
```

You can also have a look at ParentNode#append.

Selecting from a subset of the DOM

🎩 You can call `querySelector` on any element!

```
<p class="red">A red paragraph</p>

<ul id="players">
  <li class="green">Luke</li>
  <li class="red">Anakin</li>
</ul>
```

```
const list = document.querySelector("#players");
const element = list.querySelector(".red");
console.log(element.innerText);
// => ?
```

```
Anakin
```

Selecting several elements

We want to select **all** winners

```
<ol id="fifa-wins">
  <li>Brazil (5 wins)</li>
  <li>Germany (4 wins)</li>
  <li>Italy (4 wins)</li>
  <li>Argentina (2 wins)</li>
  <li>Uruguay (2 wins)</li>
</ol>
```

We can with `Element.querySelectorAll`!

```
const countries = document.querySelectorAll("#fifa-wins li");
countries.forEach((item) => {
  console.log(item.innerText);
});
```

`countries` is a `NodeList` variable.

Use the right method

```
const countries = document.querySelector("#fifa-wins li");
// => <li>Brazil (5 wins)</li>
```

`querySelector` returns **the first** element that matches selector!

```
const countries = document.querySelectorAll("#fifa-wins li");
// => NodeList(5) [li, li, li, li, li]
```

`querySelectorAll` returns them all in a list!

Your turn! How would you append `"France (2 wins)"` to the list? 🤔

```
const list = document.querySelector('#fifa-wins');
list.insertAdjacentHTML('beforeend', '<li>France (2 wins)</li>');
```

Advanced DOM Manipulations

Show / Hide

Use HTMLElement.style to change any CSS property

```
const element = document.querySelector(CSS_SELECTOR);

// Hide
element.style.display = "none";

// Show
element.style.display = "";

// Change font color
element.style.color = "red";
```

Add / Remove a class

Use `classList`

```
element.classList.add("red");
element.classList.remove("red");
element.classList.toggle("red");
```

Read / Write inputs

```html
<!-- Some HTML -->
<input name="email" id="email" value="paul@gmail.com" />
```

```javascript
const emailInput = document.querySelector("#email");

// Read
console.log(emailInput.value);

// Write
emailInput.value = "john@gmail.com";
```

Extract text / HTML

```
<a href="https://www.lewagon.com" id="home">Le Wagon <em>rocks</em></a>
```

```
const home = document.querySelector("#home");
console.log(home.innerText);
console.log(home.innerHTML);
console.log(home.attributes.href.value);

home.innerHTML = "Le Wagon <strong>rocks</strong>!"; // Update HTML
```

Dataset

Use HTMLElement.dataset

```
<div id="user" data-uid="2471555" data-github-nickname="Papillard">
  Boris Paillard
</div>
```

```
const boris = document.querySelector('#user');
console.log(boris.dataset.uid);
console.log(boris.dataset.githubNickname); // note the lowerCamelCase
```

Events

Full Reference

HTML DOM Events

```
DOMContentLoaded
blur
click
change
focus
keyup
mouseover
resize
scroll
submit
touchstart
```

Events occur on specific objects

```
DOMContentLoaded  # document
blur              # input / textarea
click             # any visible element
change            # selectors and checkboxes
input             # input / textarea
focus             # any visible element
keyup             # window / any focused element
mouseover         # any visible element
resize            # window
scroll            # window / any scrollable element
submit            # form
touchstart        # for mobile devices
```

Event Listener 🎙️

There are two ways of adding an event listener to an `element`

1. Use `addEventListener` on a DOM Element

```
// index.js
let element = document.querySelector('CSS SELECTOR')
element.addEventListener(eventType, (event) => {
  // Do_something (callback)
});
```

2. Use inline event listeners

```html
<!-- index.html -->
<button onclick="Do_something(event)">Click me!</button>
```

```js
const Do_something = (event) => // callback
```

We will mostly use... **inline event listeners**! 🎉

1. Working with a JS framework - Vue, React, Angular - you will encounter these
2. They are also more similar to the mini program framework syntax

What's a callback?

Listening to a click

```
<img src="https://kitt.lewagon.com/placeholder/users/monsieurpaillard"
     id="romain" height="200" alt="Romain Paillard"
     onclick="logEvent(event)"
      />
```

```
const logEvent = (event) => {
  console.log(event); // see the event details
  console.log(event.target); // see the element causing the event
}
});
```

You can read more about Event.target

> UX tip: change the default cursor if the image is clickable.

Live-code

Toggle the `img-circle` CSS class when clicking on these images.

```
.img-circle {
  border-radius: 50%;
}
```

You might want to add an `onclick` on an image and then...

```
const roundifyImage = (event) => {
  let element = event.target
  element.classList.toggle = "img-circle"
}
```

BUT you can instead pass the element itself using `this` keyword!

```
<img src="https://kitt.lewagon.com/placeholder/users/monsieurpaillard"
     id="romain" height="200" alt="Romain Paillard"
     onclick="roundifyImage(this)"
      />
```

```
const roundifyImage = (element) => {
  element.classList.toggle = "img-circle"
}
```

What if we have many elements that need to be interactive?

```
<img src="https://kitt.lewagon.com/placeholder/users/monsieurpaillard"
  id="romain" height="200" alt="Romain Paillard" />

<img src="https://kitt.lewagon.com/placeholder/users/Papillard"
  id="boris" height="200" alt="Boris Paillard" />

<img src="https://kitt.lewagon.com/placeholder/users/tgenaitay"
  id="boris" height="200" alt="Thibault Genaitay" />

<img src="https://kitt.lewagon.com/placeholder/users/smithavt14"
  id="boris" height="200" alt="Alex Smith" />

<!-- and hundreds more... --->
```

In this case we should add the event handler to all the images using `querySelectorAll`

```
document.querySelectorAll("img").forEach((img) => {
  img.setAttribute("onclick", "roundifyImage(this)");
});
```

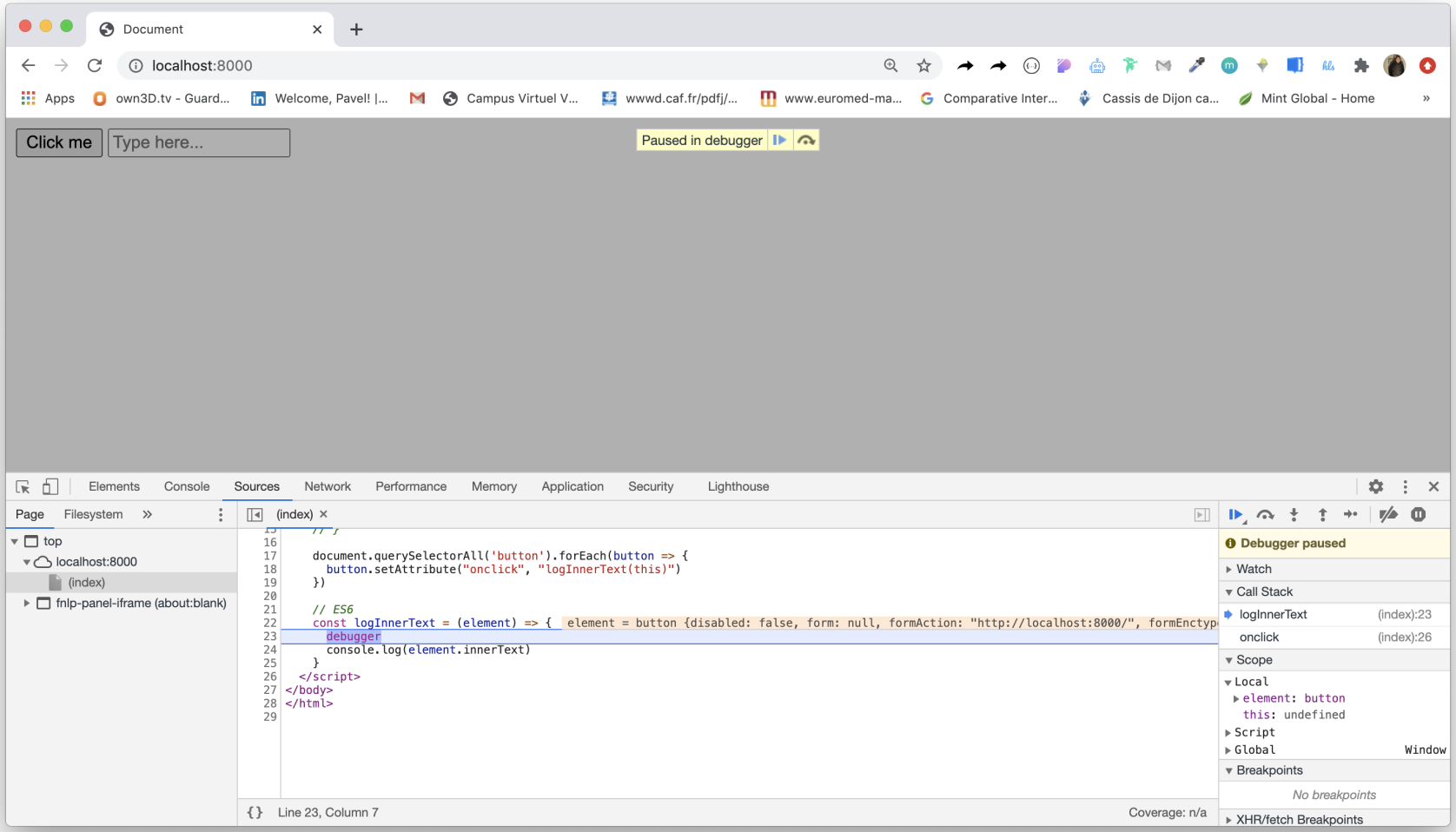
> `attributes` are *any* parameters that go inside HTML tags, like `id`, `src` or `onclick`

Debugging

Add this to your JavaScript file and open your browser's **inspector**.

```
debugger
```

The `debugger` keyword will pause the runtime, giving you the chance to explore your code



Happy JavaScripting! 🚀