


Let's open up the browser

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


Coding Bootcamp Le Wagon | x

Secure https://www.lewagon.com

Discover what our students are able to build after 9 weeks, [watch our demos](#)




CITIES | PROGRAM | FAQ




DEMODOY | ALUMNI | BUSINESS

Apply Now



Le Wagon - Coding Bootcamp

Change your life: learn to code



Elements Console Sources Network Performance Memory Application Security >> X

https://www Filter Info ▼ ⚙

```
> const h1 = document.querySelector('h1')
< undefined
> h1.innerText
< "Le Wagon - Coding Bootcamp"
>
```

Using JavaScript with HTML

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

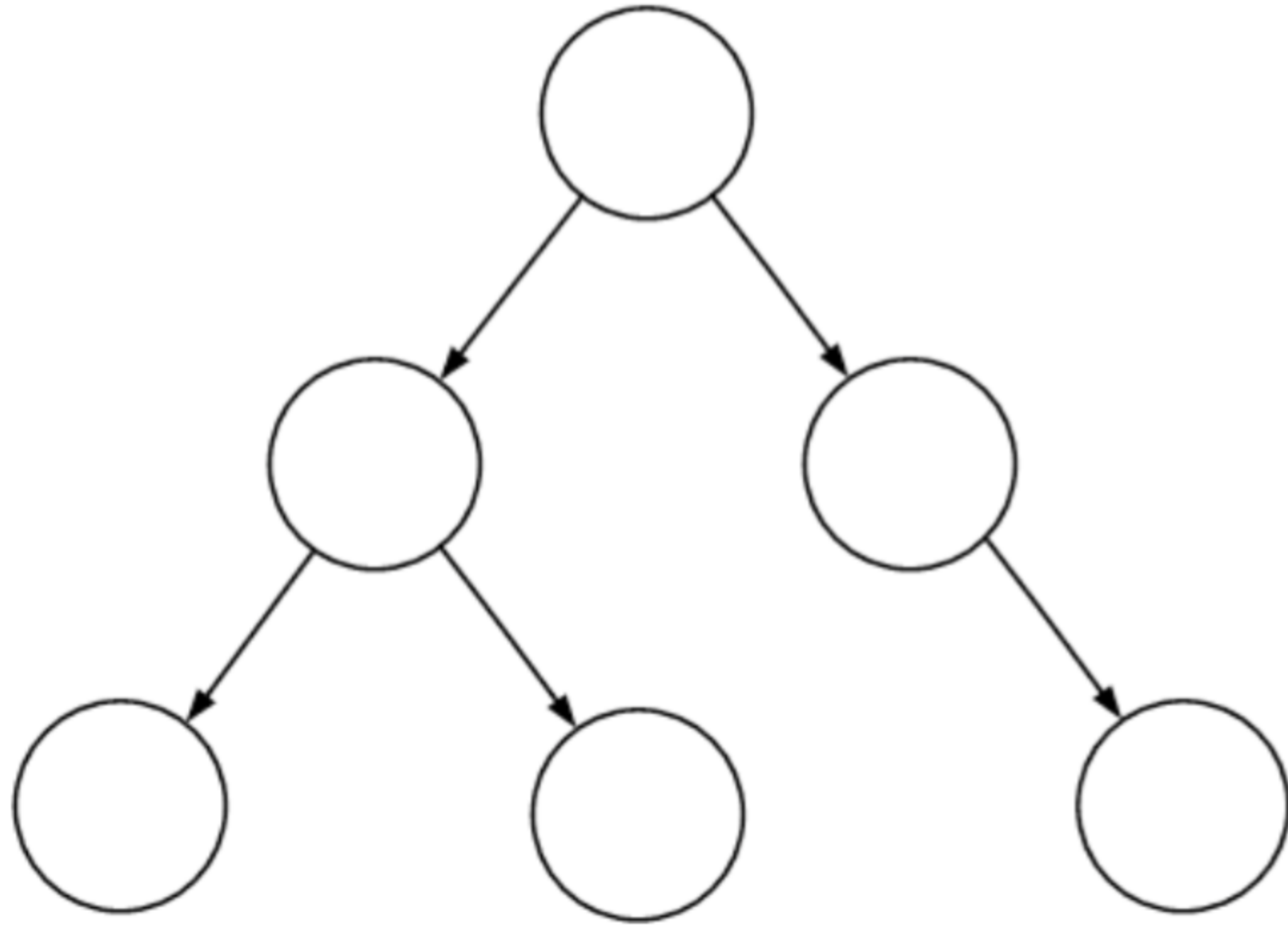
  </head>
  <body>
    <!-- Your content -->

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

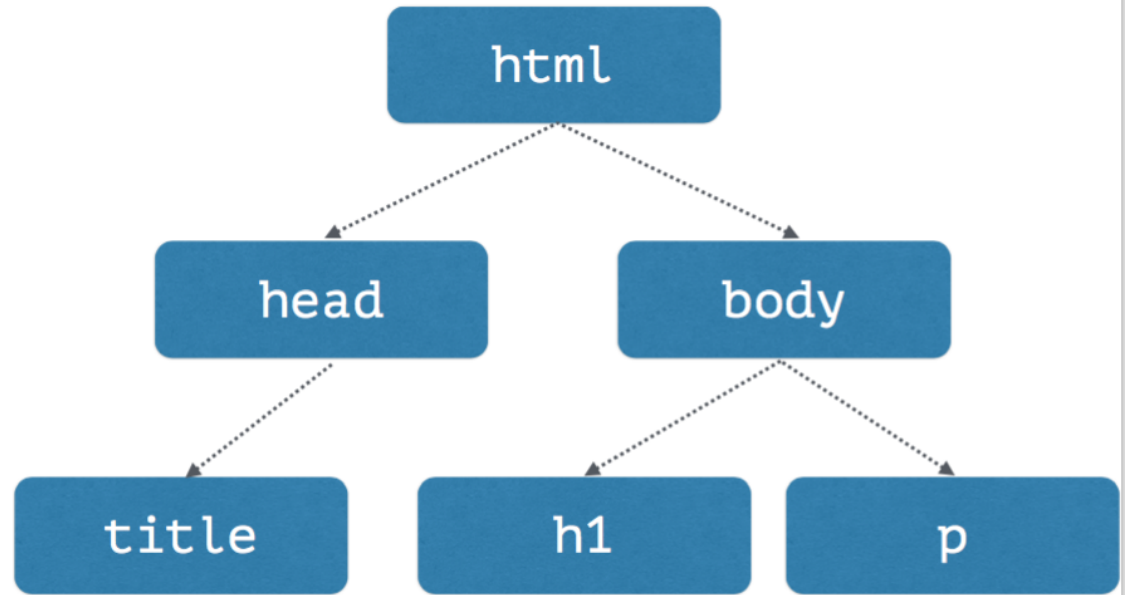
Loading a JS file is **blocking** the page rendering => Put it at the end.

DOM

Document Object Model

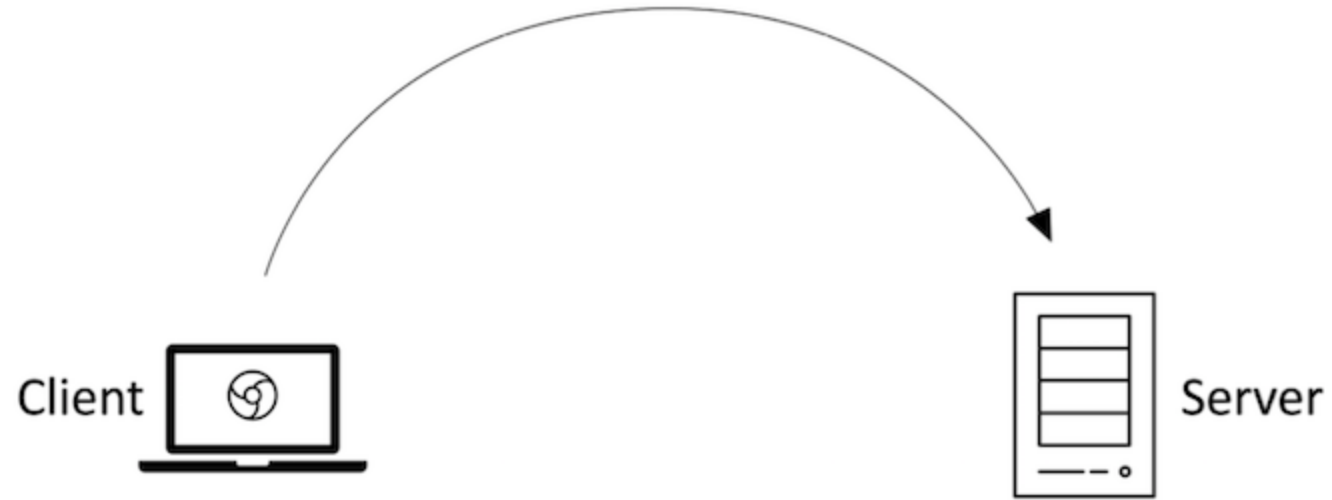


```
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <h1>
      Hello
    </h1>
    <p>
      Lorem Ipsum..
    </p>
  </body>
</html>
```



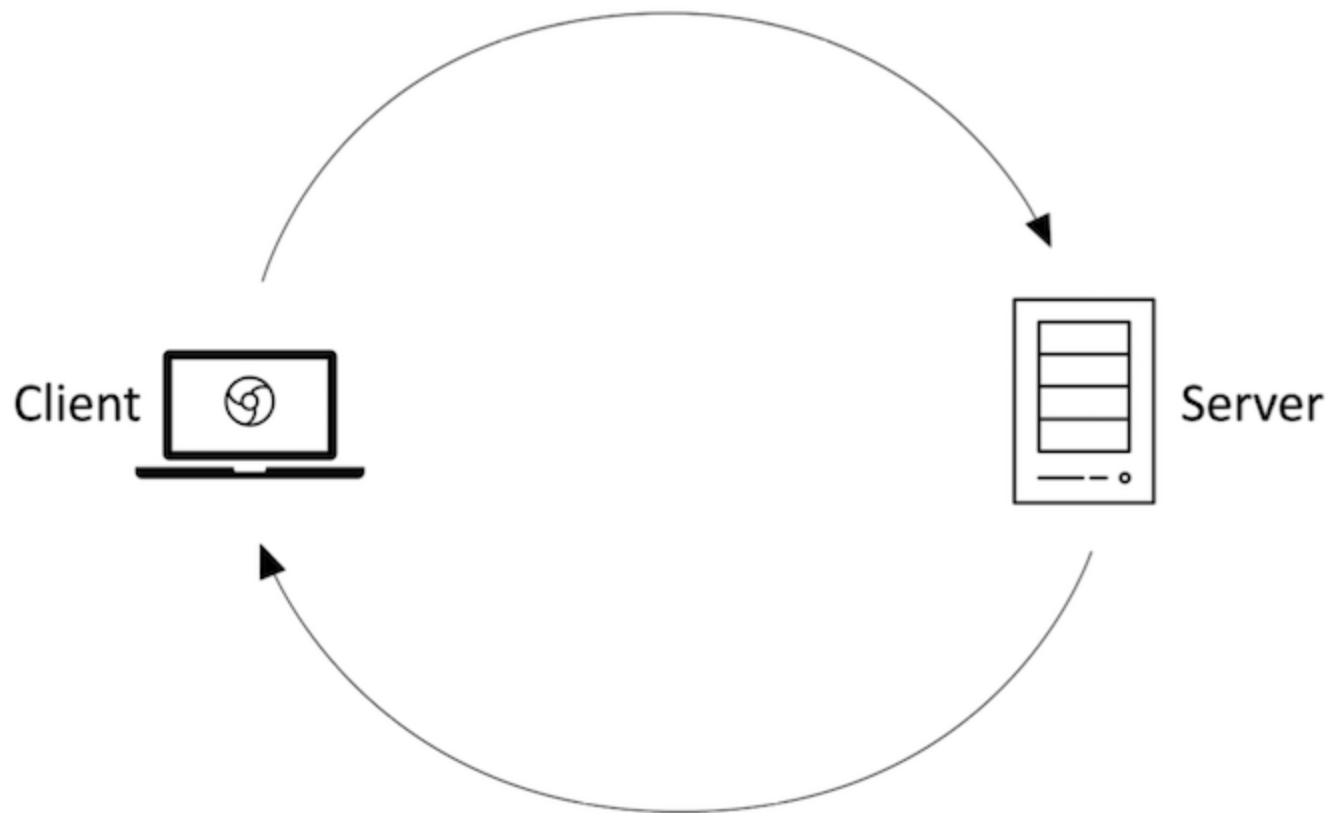
HTTP request with URL

GET "http://lewagon.org/program"



HTTP request with **URL**

GET "http://lewagon.org/program"

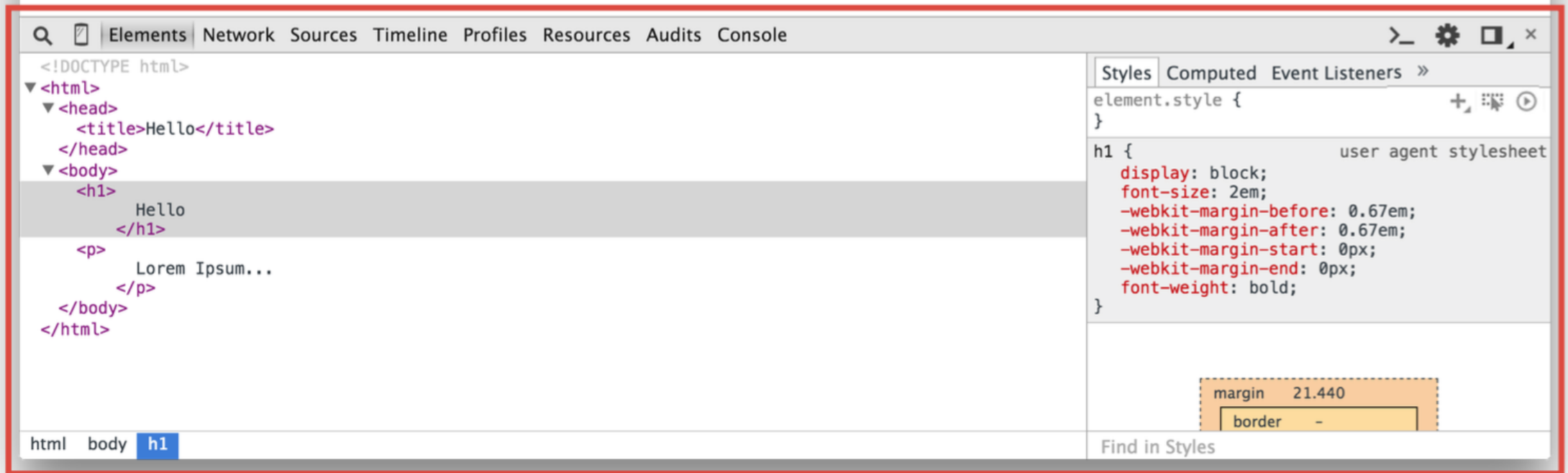
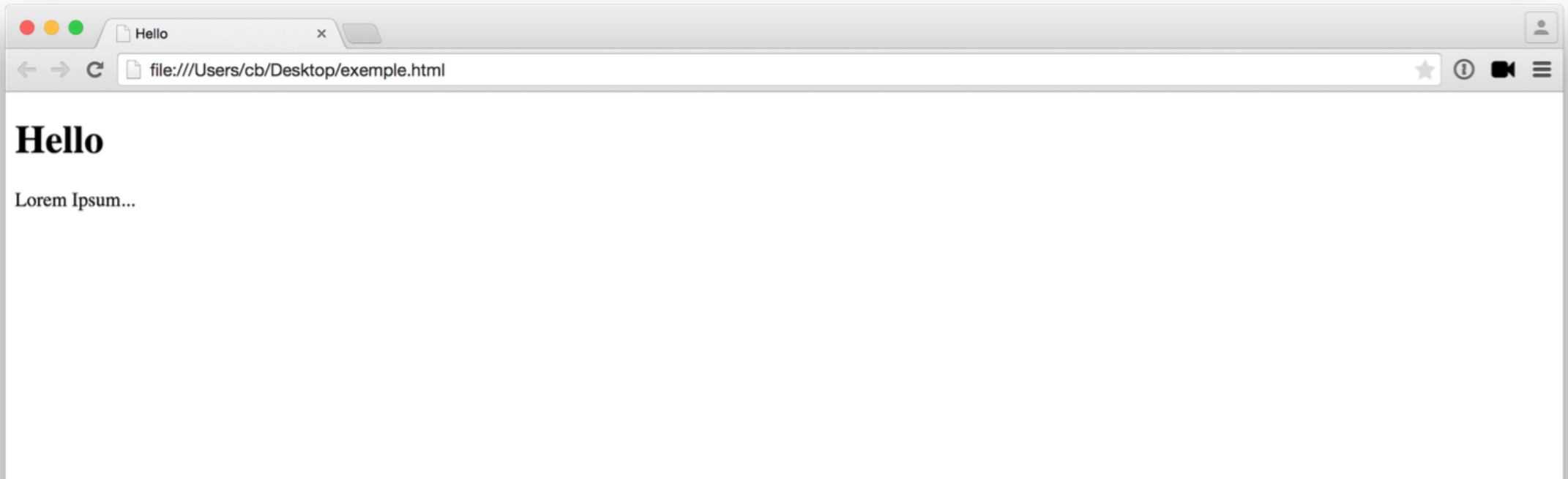


HTTP response with **HTML** file



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

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



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");
```

🤔 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    /* Descending combinator */  
ul > .active  /* Child combinator */
```

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

```
jsdocument.querySelector('ul#players > .active a.btn');
```

We've just selected an element! 💪

What can we do now? 🤔

Append content

We are using the [Element#insertAdjacentHTML](#) method.

```
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 it finds!

```
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`

```
const element = document.querySelector(CSS_SELECTOR);  
  
// Hide  
element.style.display = "none";  
  
// Show  
element.style.display = "";
```

Add / Remove a class

Use `classList`

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

Read / Write inputs

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

```
const emailInput = document.getElementById("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.getElementById("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.getElementById('user');  
console.log(boris.dataset.uid);  
console.log(boris.dataset.githubNickname);
```


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	# select
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

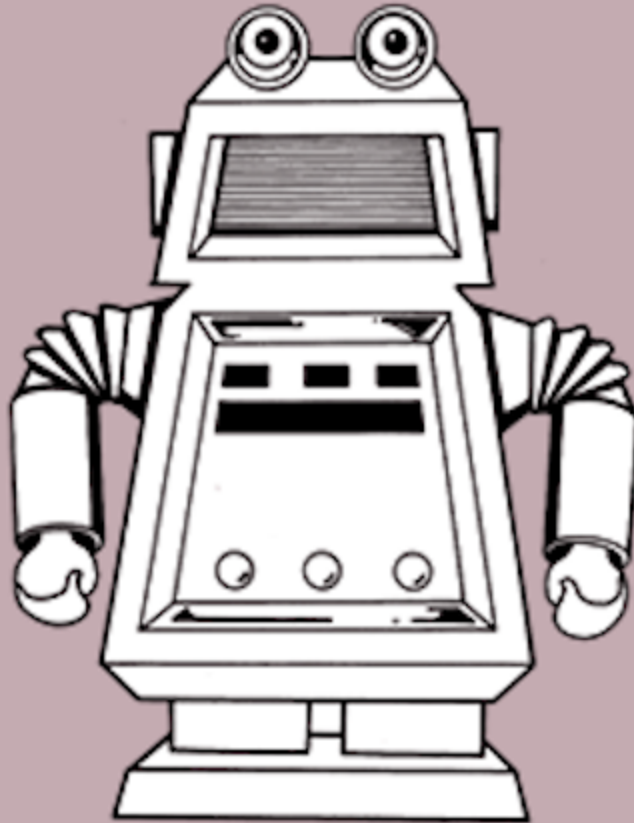
Use `addEventListener` to react to an event.

```
element.addEventListener(eventType, (event) => {  
  // Do something (callback)  
});
```

What's a callback?

Don't call us, we'll
call you

- The Hollywood
Principle



som^{ee}cards
user card

Listening to a click

```

```

```
const romain = document.getElementById("romain");  
romain.addEventListener("click", (event) => {  
  console.log(event);  
  console.log(event.currentTarget);  
});
```

You can read more about [Event.currentTarget](#)

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%;  
}
```

What if we have several elements?

```


```

```
document.querySelectorAll("img").forEach((img) => {
  img.addEventListener("click", (event) => {
    event.currentTarget.classList.toggle("img-circle");
  });
});
```


Debugging

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

```
debugger
```

Happy JavaScripting!