Today's schedule

- DOM: How to interact with your web page

Events

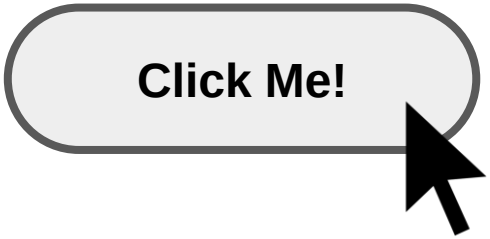
Event-driven programming

Most JavaScript written in the browser is **event-driven**:
The code doesn't run right away, but it executes after some event fires.

Example:
Here is a UI element that the user can interact with.

Event-driven programming

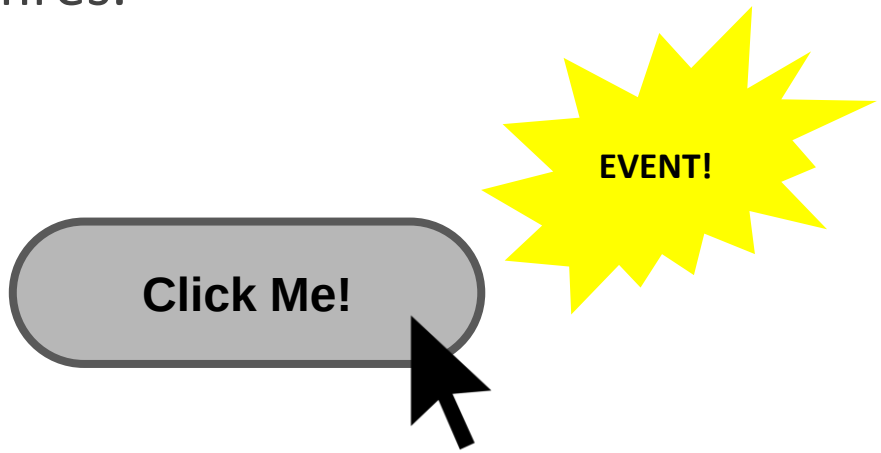
Most JavaScript written in the browser is **event-driven**:
The code doesn't run right away, but it executes after some event fires.



When the user clicks the button...

Event-driven programming

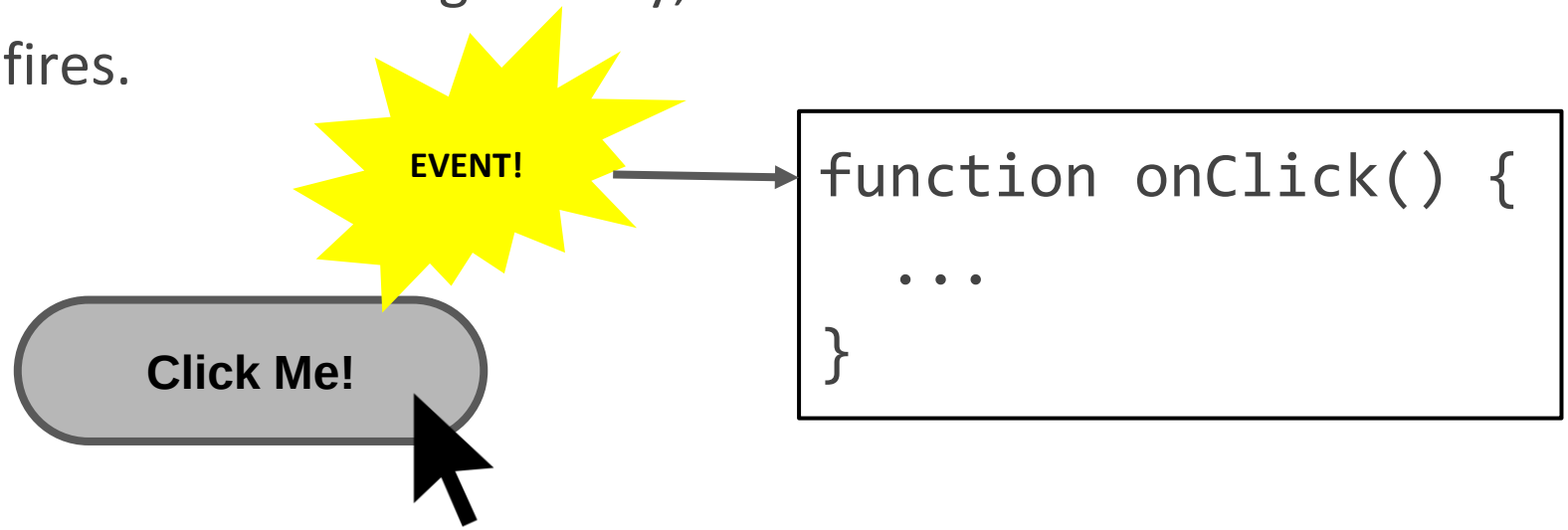
Most JavaScript written in the browser is **event-driven**: The code doesn't run right away, but it executes after some event fires.



...the button emits an "**event**," which is like an announcement that some interesting thing has occurred.

Event-driven programming

Most JavaScript written in the browser is **event-driven**:
The code doesn't run right away, but it executes after some event fires.



Any function listening to that event now executes. This function is called an "**event handler**."

A few more HTML elements

Buttons:

```
<button>Click me</button>
```

Click me

Single-line text input:

```
<input type="text" />
```

hello

Multi-line text input:

```
<textarea></textarea>
```

I can add
multiple lines of text!

Using event listeners

Let's print "Clicked" to the Web Console when the user clicks the given button:

```
<button>Click me</button>
```

Click me

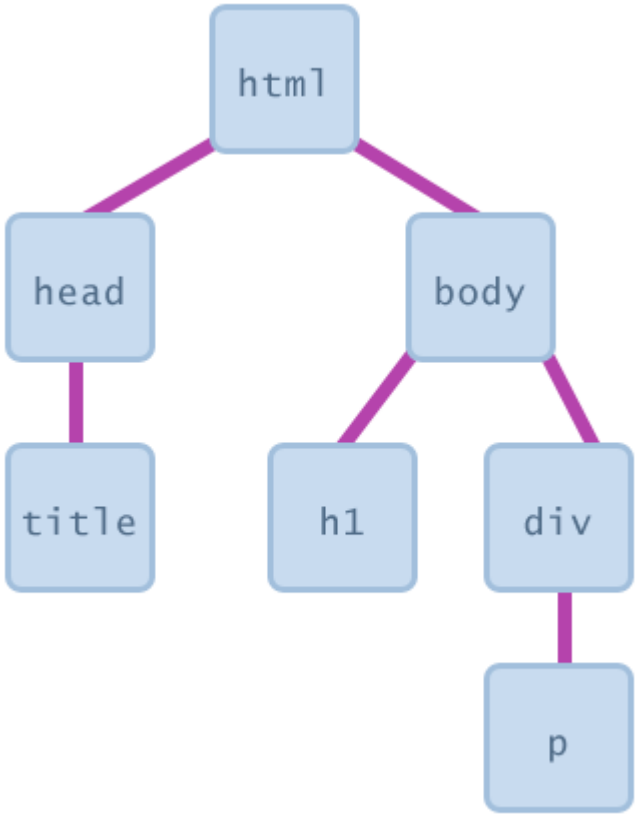
We need to add an event listener to the button...

How do we talk to an element in HTML from JavaScript?

The DOM

Every element on a page is accessible in JavaScript through the **DOM**: **Document Object Model**

- The DOM is the tree of nodes corresponding to HTML elements on a page.
- Can modify, add and remove nodes on the DOM, which will modify, add, or remove the corresponding element on the page.



Getting DOM objects

We can access an HTML element's corresponding DOM node in JavaScript via the `querySelector` function:

```
document.querySelector('css selector');
```

- Returns the **first** element that matches the given CSS selector.

And via the `querySelectorAll` function:

```
document.querySelectorAll('css selector');
```

- Returns **all** elements that match the given CSS selector.

Getting DOM objects

```
// Returns the DOM object for the HTML element
// with id="button", or null if none exists.
let element = document.querySelector('#button');


// Returns a list of DOM objects containing all
// elements that have a "quote" class AND all
// elements that have a "comment" class.
let elementList =
    document.querySelectorAll('.quote, .comment');
```

Adding event listeners

Each DOM object has the following function:

`addEventListener(`***event name, function name***`);`

- ***event name*** is the string name of the JavaScript event you want to listen to
  - Common ones: click, focus, blur, etc
- ***function name*** is the name of the JavaScript function you want to execute when the event fires

Removing event listeners

To stop listening to an event, use removeEventListener:

`removeEventListener(`***event name, function name***`);`

- ***event name*** is the string name of the JavaScript event to stop listening to
- ***function name*** is the name of the JavaScript function you no longer want to execute when the event fires

```html
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```

```javascript
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

script.js

```
1 function onClick() {
2   console.log('clicked');
3 }
4
5 const button = document.querySelector('button');
6 button.addEventListener('click', onClick);
7
```

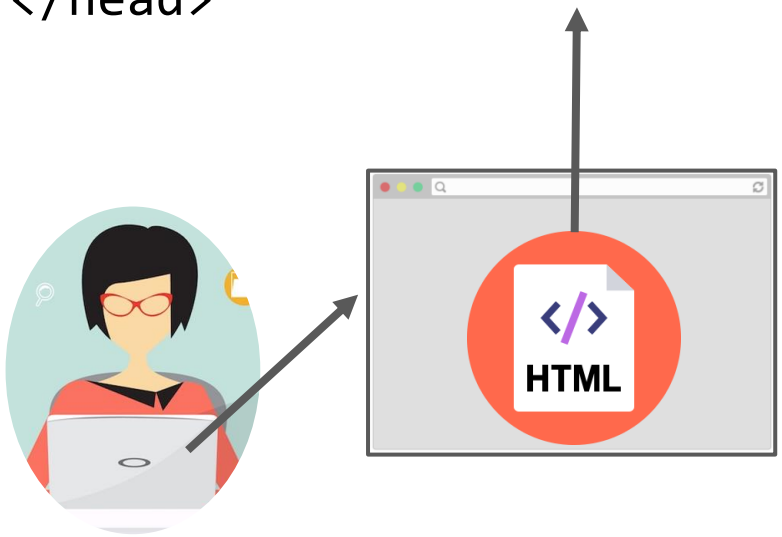
Elements Console Sources Network Timeline Profiles

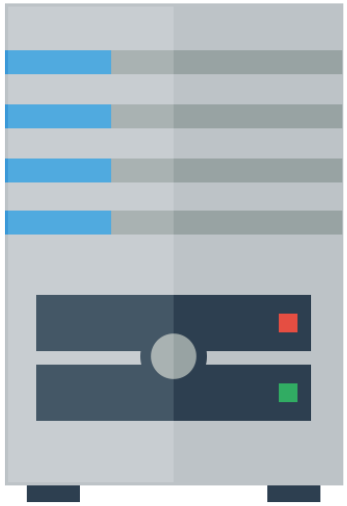
top Preserve log

```
Uncaught TypeError: Cannot read property 'addEventListener' of null
    at script.js:6
>
```

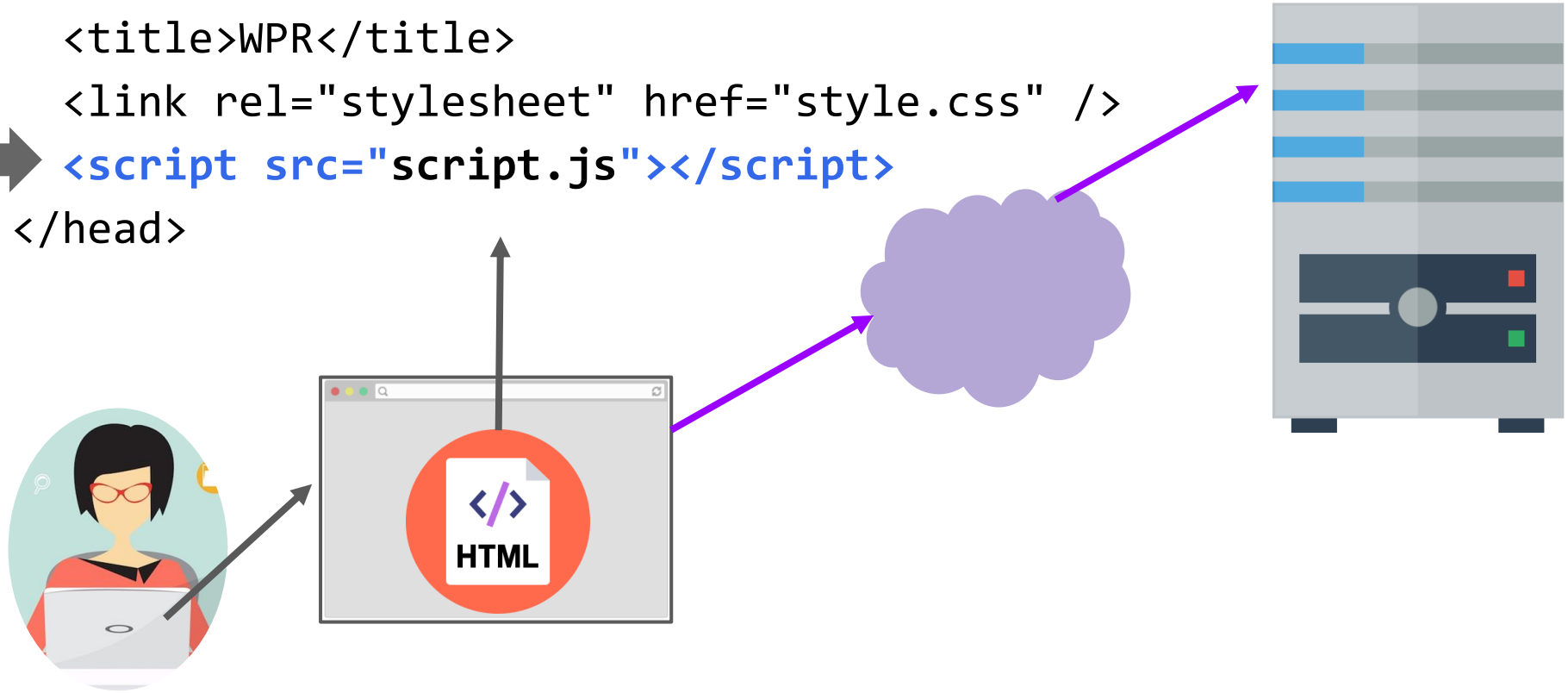
Error! Why?

```
<head>
  <title>WPR</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```

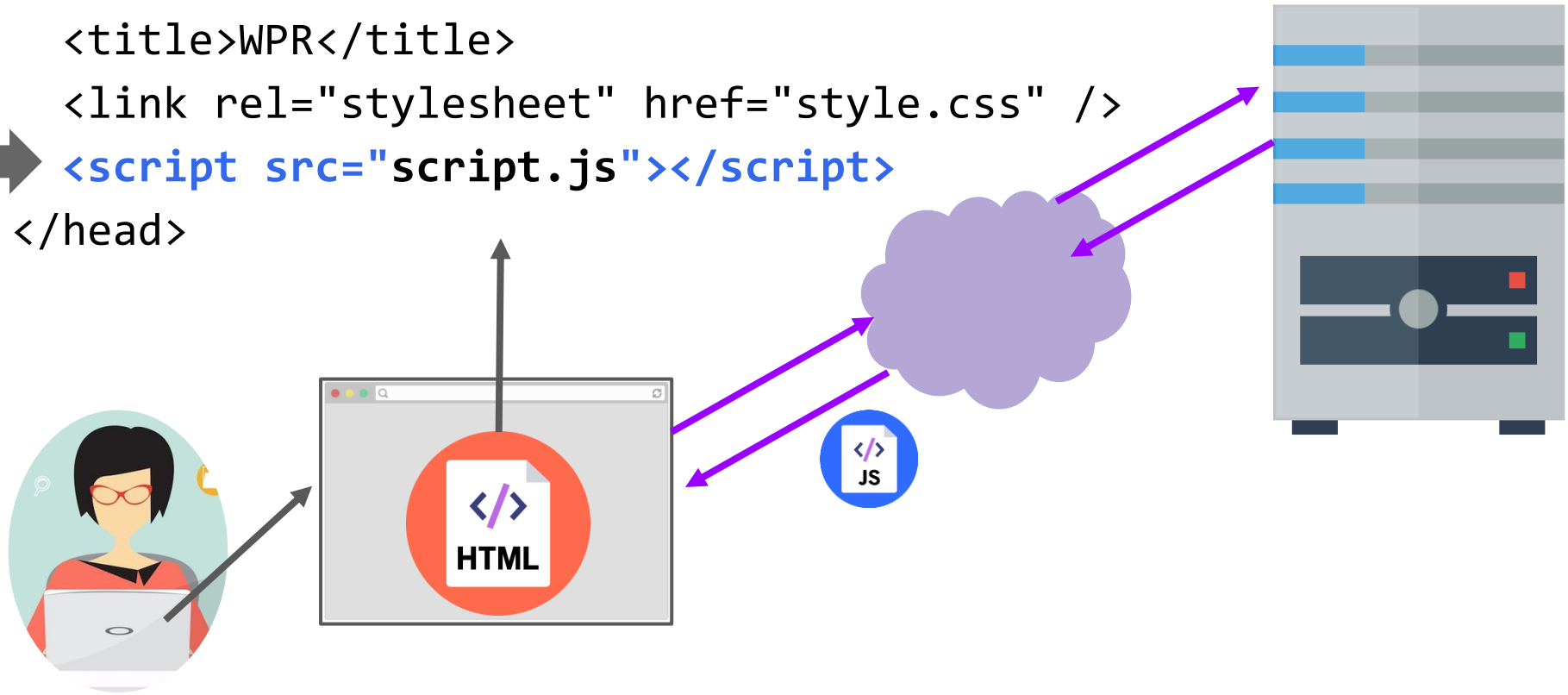




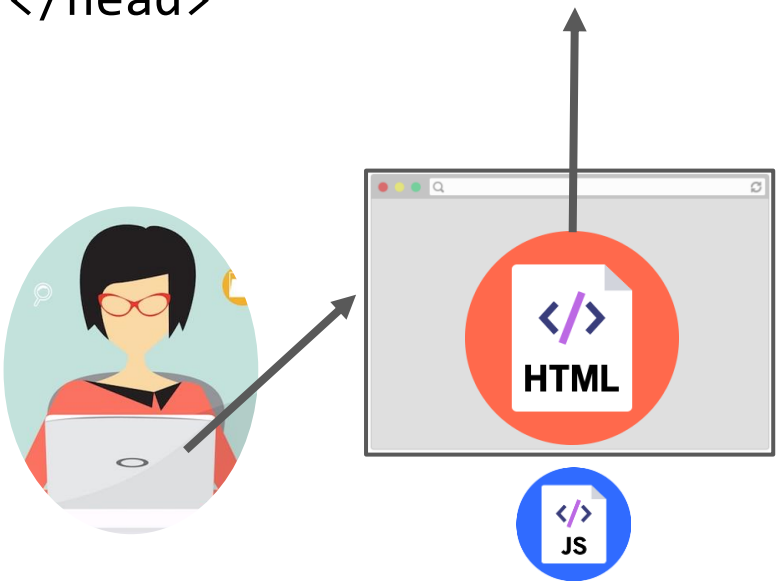
```
<head>
  <title>WPR</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



```
<head>
  <title>WPR</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



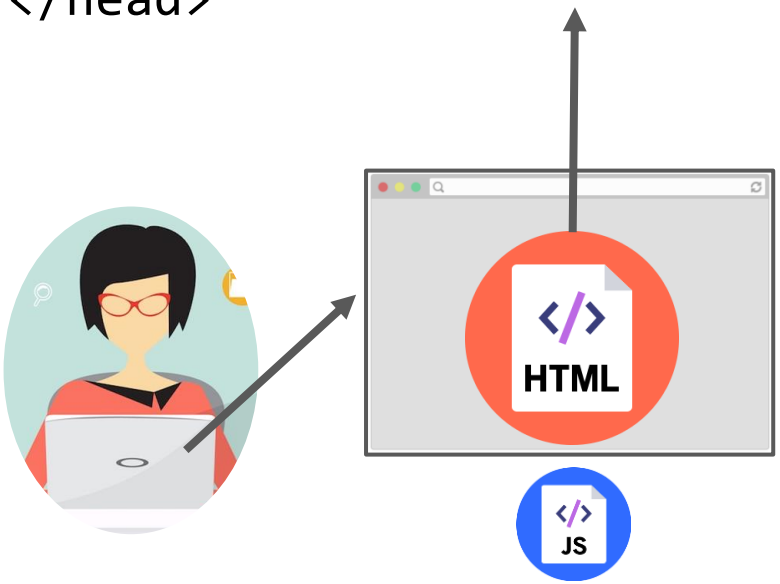
```html
<head>
  <title>WPR</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```




```js
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
```

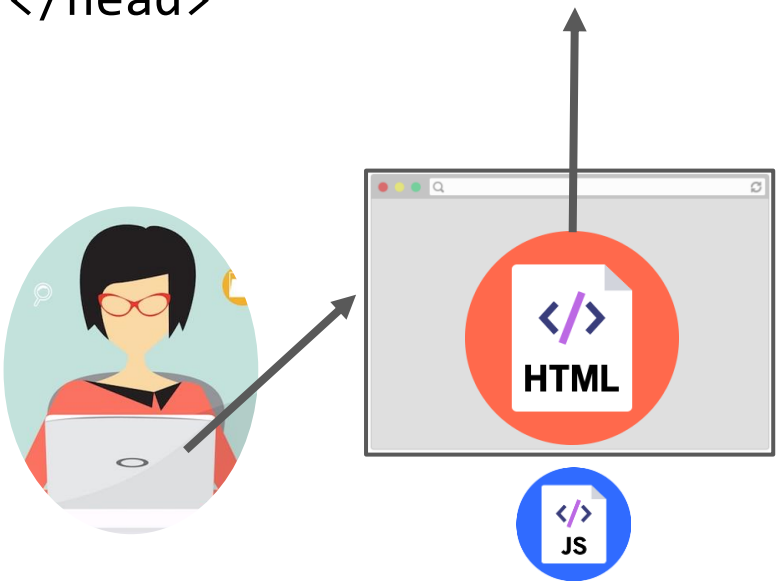
```
<head>
  <title>WPR</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```




```
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
```

```html
<head>
  <title>WPR</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```

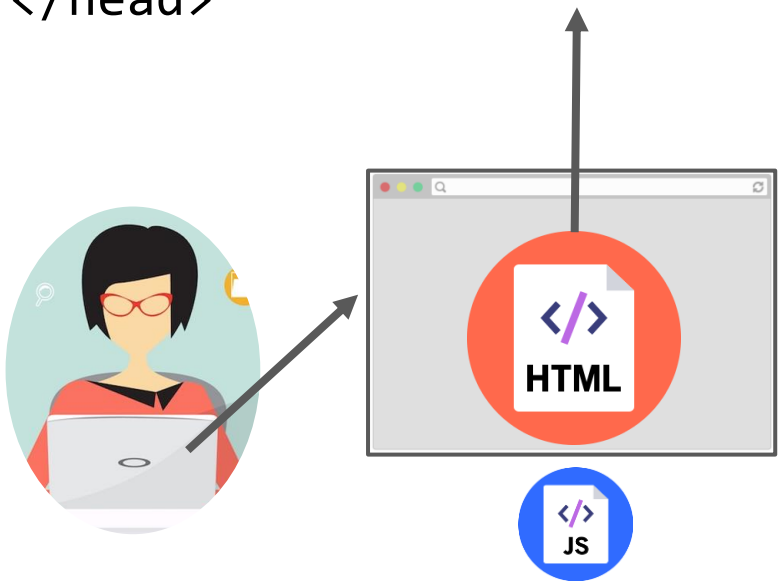



```
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
```

We are only at the `<script>` tag, which is at the top of the document... so the `<button>` isn't available yet.

```
<head>
  <title>WPR</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```




```
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
```

Therefore `querySelector` returns `null`, and we can't call `addEventListener` on `null`.

Use `defer`

You can add the `defer` attribute onto the script tag so that the JavaScript doesn't execute until after the DOM is loaded (mdn):

```
<script src="script.js" defer></script>
```

Use `defer`

You can add the `defer` attribute onto the script tag so that the JavaScript doesn't execute until after the DOM is loaded (mdn):

```
<script src="script.js" defer></script>
```

Other old-school ways of doing this (**don't do these**):

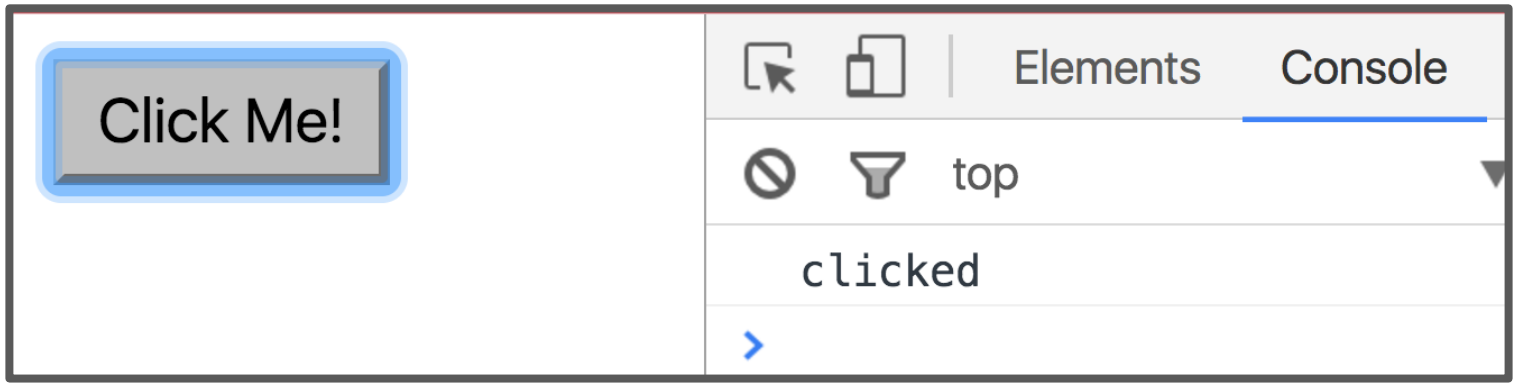
- Put the `<script>` tag at the bottom of the page
- Listen for the "`load`" event on the window object

You will see tons of examples on the internet that do this. They are out of date. defer is widely supported and better.

```html
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js" defer></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```

```javascript
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```



How do we interact with the page?

A few technical details

The DOM objects that we retrieve from `querySelector` and `querySelectorAll` have types:

- Every DOM node is of general type `Node` (an interface)
- `Element` implements the `Node` interface
  (FYI: This has nothing to do with NodeJS, if you've heard of that)
- Each HTML element has a specific `Element` derived class, like `HTMLImageElement`

Attributes and DOM properties

Roughly every **attribute** on an HTML element is a **property** on its respective DOM object...

HTML

```
<img src="puppy.png" />
```

JavaScript

```
const element = document.querySelector('img');
element.src = 'bear.png';
```

(But you should always check the JavaScript spec to be sure.
In this case, check the `HTMLImageElement`.)

Some properties of `Element` objects

Property	Description
`id`	The value of the id attribute of the element, as a string
`innerHTML`	The raw HTML between the starting and ending tags of an element, as a string
`textContent`	The text content of a node and its descendants. (This property is inherited from Node)
`classList`	An object containing the classes applied to the element

Maybe we can adjust
the **`textContent`**!
CodePen

More next time!