

HEIG-VD / INGÉNIERIE DES MÉDIAS / PROGRAMMATION WEB

INTRODUCTION AU DOM

```
<div class="card">
  
  <p>Lorem ipsum dolor sit amet consectetur.</p>
</div>
```

Qu'est-ce que le DOM?

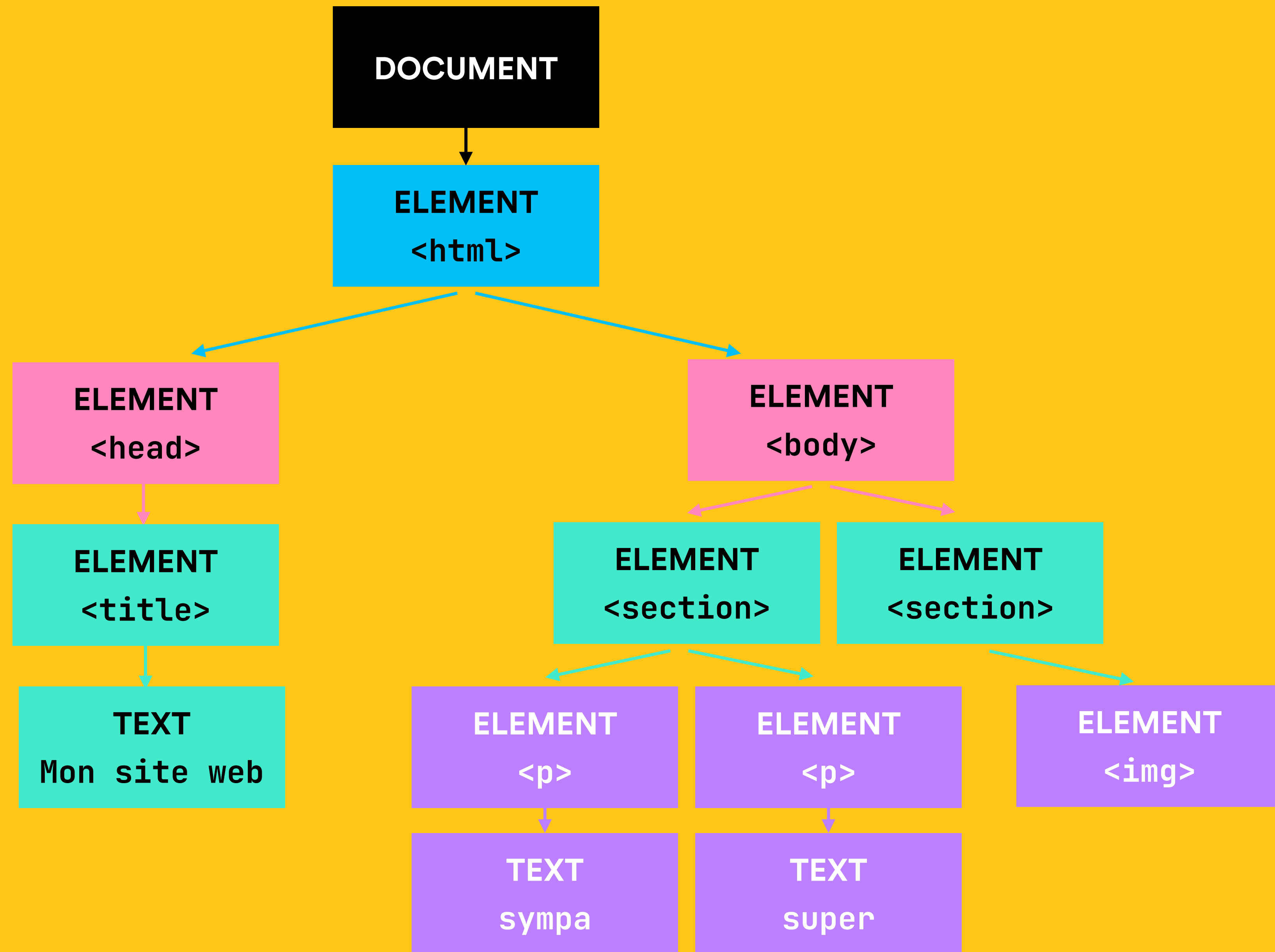
Document Object Model: Une représentation structurée de documents HTML. Il permet au JavaScript d'accéder aux différents éléments HTML et de les manipuler.

Qu'est-ce que le DOM?

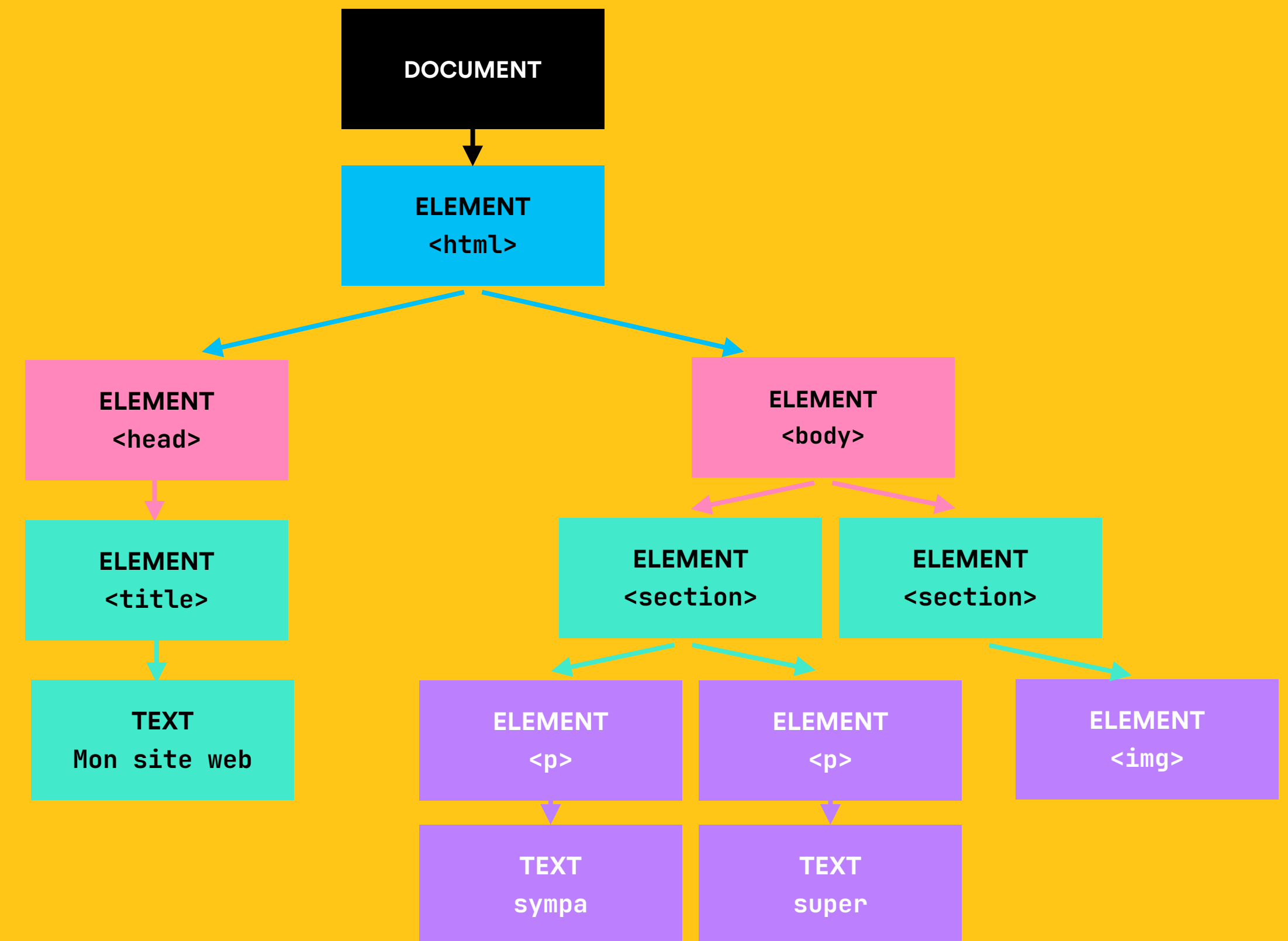
Document Object Model: Une représentation structurée de documents HTML. Il permet au JavaScript d'accéder aux différents éléments HTML et de les manipuler.

Changer le texte, les attributs HTML et même les styles CSS.





```
<html>
  <head>
    <title>Mon site web</title>
  </head>
  <body>
    <section>
      <p>sympa</p>
      <p>super</p>
    </section>
    <section>
      
    </section>
  </body>
</html>
```



DOCUMENT

A diagram illustrating a hierarchy. A black rectangular box at the top contains the word "DOCUMENT" in white, bold, uppercase letters. A thin black arrow points vertically downwards from the bottom center of this box to the top center of a light blue rectangular box below it. The light blue box contains the word "ELEMENT" in black, bold, uppercase letters.

ELEMENT

On objet spécial qui est le point
d'entrée vers le DOM.



DOCUMENT



ELEMENT


```
graph TD; ELEMENT[ELEMENT] --> TITLE[<title>]; ELEMENT --> TEXT[TEXT]; TEXT --> MON[Mon site web];
```

ELEMENT

<title>

TEXT

Mon site web

Tout ce qui est dans un document HTML se retrouve automatiquement dans le DOM.

Ce dernier est une représentation complète du document HTML.

ELEMENT

<title>

TEXT

Mon site web

DOM !== JS

JS

Web APIs



Web APIs

DOM

BACKGROUND TASKS

FILE API

UI EVENTS

CANVAS

FETCH API

GEOLOCATION

STORAGE

TOUCH EVENTS

WEB ANIMATION

WEB SPEECH

WEB SOCKETS

WEB WORKERS

WEBRTC

...

```
document.querySelector("sel");
```

Le méthode `querySelector()`
nous permet d'aller piocher
un élément dans le DOM.

```
document.querySelector("element");  
document.querySelector(".class");  
document.querySelector("#id");
```

On utilise la même syntaxe que pour les sélecteurs CSS.

querySelector() retourne le premier élément répondant aux critères du sélecteur passé en paramètre.

L'élément est représenté sous forme d'objet, avec des méthodes et des propriétés qui lui sont propres.

INDEX.HTML

```
<body>
  <div class="card"></div>
  <div class="card"></div>
</body>
```

SCRIPT.JS

```
document.querySelector(".card");
```


INDEX.HTML

```
<body>
  <div class="card"></div>
  <div class="card"></div>
</body>
```

SCRIPT.JS

```
document.querySelector(".card");
```

```
ariaLabel: null
ariaLevel: null
ariaLive: null
ariaModal: null
ariaMultiLine: null
ariaMultiSelectable: null
ariaOrientation: null
ariaPlaceholder: null
ariaPosInSet: null
ariaPressed: null
ariaReadOnly: null
ariaRelevant: null
ariaRequired: null
ariaRoleDescription: null
ariaRowCount: null
ariaRowIndex: null
ariaRowSpan: null
ariaSelected: null
ariaSetSize: null
ariaSort: null
ariaValueMax: null
ariaValueMin: null
ariaValueNow: null
ariaValueText: null
assignedSlot: null
▶ attributeStyleMap: StylePropertyMap {size: 0}
▶ attributes: NamedNodeMap {0: class, class: class, length: 1}
autocapitalize: ""
autofocus: false
baseURI: "http://127.0.0.1:5500/"
childElementCount: 0
▶ childNodes: NodeList []
▶ children: HTMLCollection []
▶ classList: DOMTokenList ['card', value: 'card']
className: "card"
clientHeight: 0
clientLeft: 0
clientTop: 0
clientWidth: 935
contentEditable: "inherit"
▶ dataset: DOMStringMap {}
dir: ""
draggable: false
elementTiming: ""
enterKeyHint: ""
firstChild: null
firstElementChild: null
hidden: false
id: ""
inert: false
innerHTML: ""
innerText: ""
inputMode: ""
isConnected: true
isContentEditable: false
lang: ""
lastChild: null
```

INDEX.HTML

```
<body>  
  <div id="card">Super! </div>  
</body>
```

OUTPUT

Super!

SCRIPT.JS

```
document.querySelector("#card").textContent
```

INDEX.HTML

```
<body>  
  <input type="number" />  
</body>
```

OUTPUT

i.e. 23

SCRIPT.JS

```
document.querySelector("input").value
```

```
<body>
  <div class="card"></div>
  <div class="card"></div>
</body>
```

```
document.querySelectorAll(".card");
// NodeList(2) [div.card, div.card]
```

Le méthode `querySelectorAll()` nous permet d'aller piocher plusieurs éléments dans le DOM.

Cette méthode retourne un objet `NodeList`, qui a la particularité d'être itérable. Ce n'est techniquement pas un tableau, même si ça en a tout l'air.

On peut utiliser `for...of` ou `forEach()` pour traverser une `NodeList`.

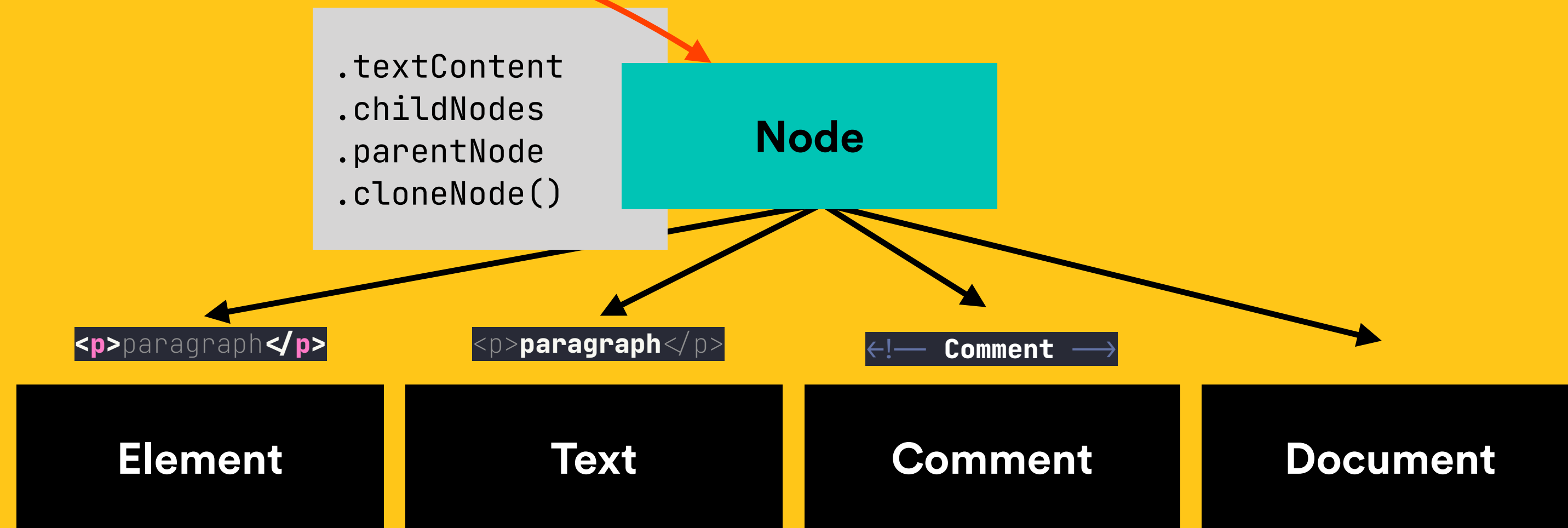
Method	Searches by...	Can call on an element?	Live?
querySelector	CSS-selector	✓	-
querySelectorAll	CSS-selector	✓	-
getElementById	id	-	-
getElementsByName	name	-	✓
getElementsByTagName	tag or '*'	✓	✓
getElementsByClassName	class	✓	✓

Un objet JavaScript

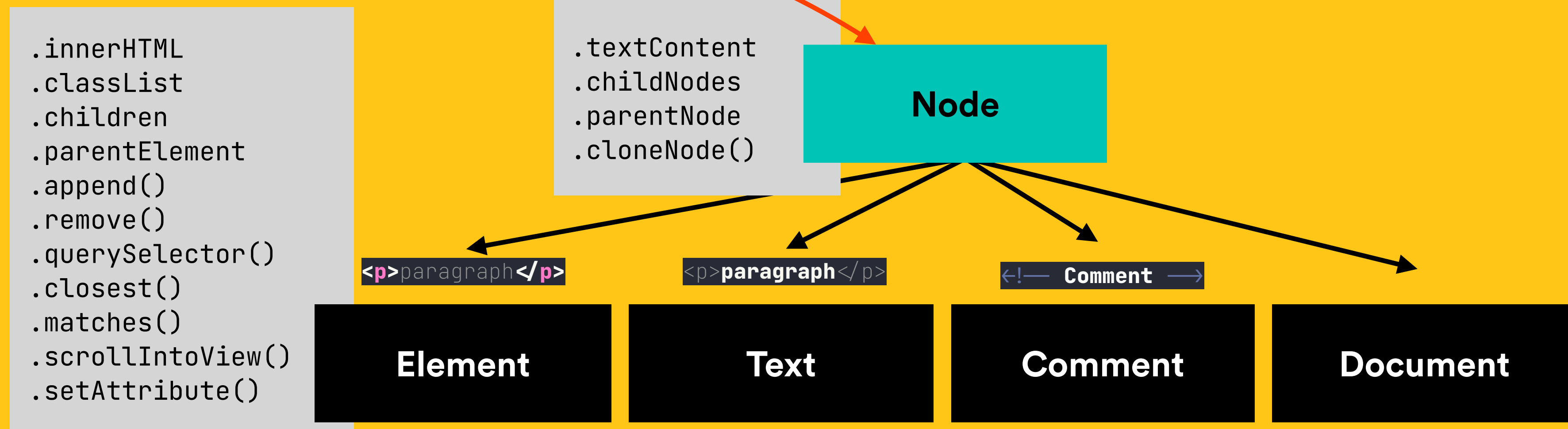
```
.textContent  
.childNodes  
.parentNode  
.cloneNode()
```

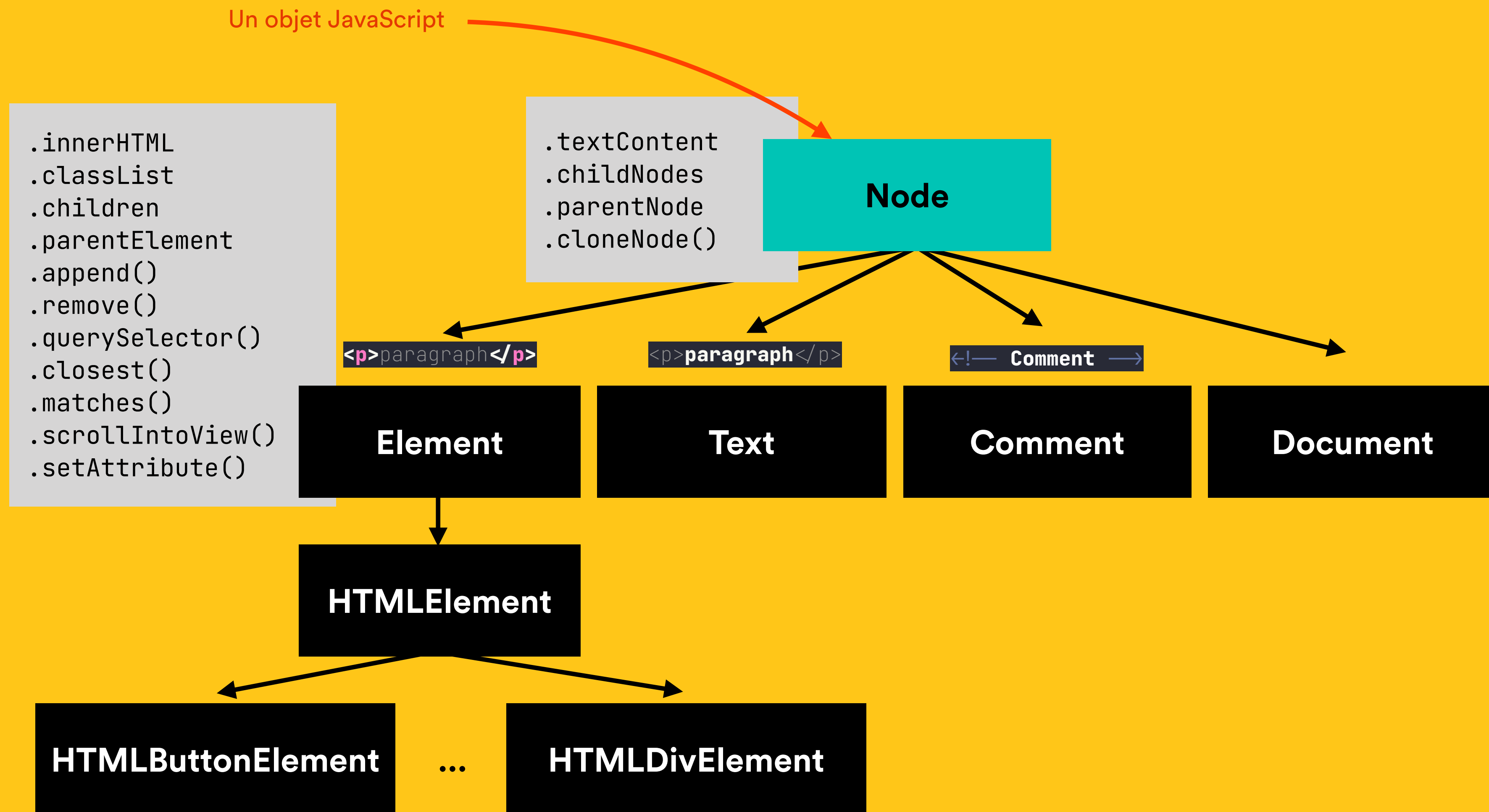
Node

Un objet JavaScript



Un objet JavaScript





Un objet JavaScript

```
.innerHTML  
.classList  
.children  
.parentElement  
.append()  
.remove()  
.querySelector()  
.closest()  
.matches()  
.scrollIntoView()  
.setAttribute()
```

```
.textContent  
.childNodes  
.parentNode  
.cloneNode()
```

Node

`<p>paragraph</p>`

Element

`<p>paragraph</p>`

Text

`<!-- Comment -->`

Comment

Document

HTMLElement

`.disabled`

HTMLButtonElement

...

HTMLDivElement

Un objet JavaScript

Héritage des propriétés et des méthodes!

Par exemple:
HTMLElement à accès aux méthodes
cloneNode() ou
closest()

```
.innerHTML  
.classList  
.children  
.parentElement  
.append()  
.remove()  
.querySelector()  
.closest()  
.matches()  
.scrollIntoView()  
.setAttribute()
```

```
.textContent  
.childNodes  
.parentNode  
.cloneNode()
```

Node

`<p>paragraph</p>`

Element

`<p>paragraph</p>`

Text

`<!-- Comment -->`

Comment

Document

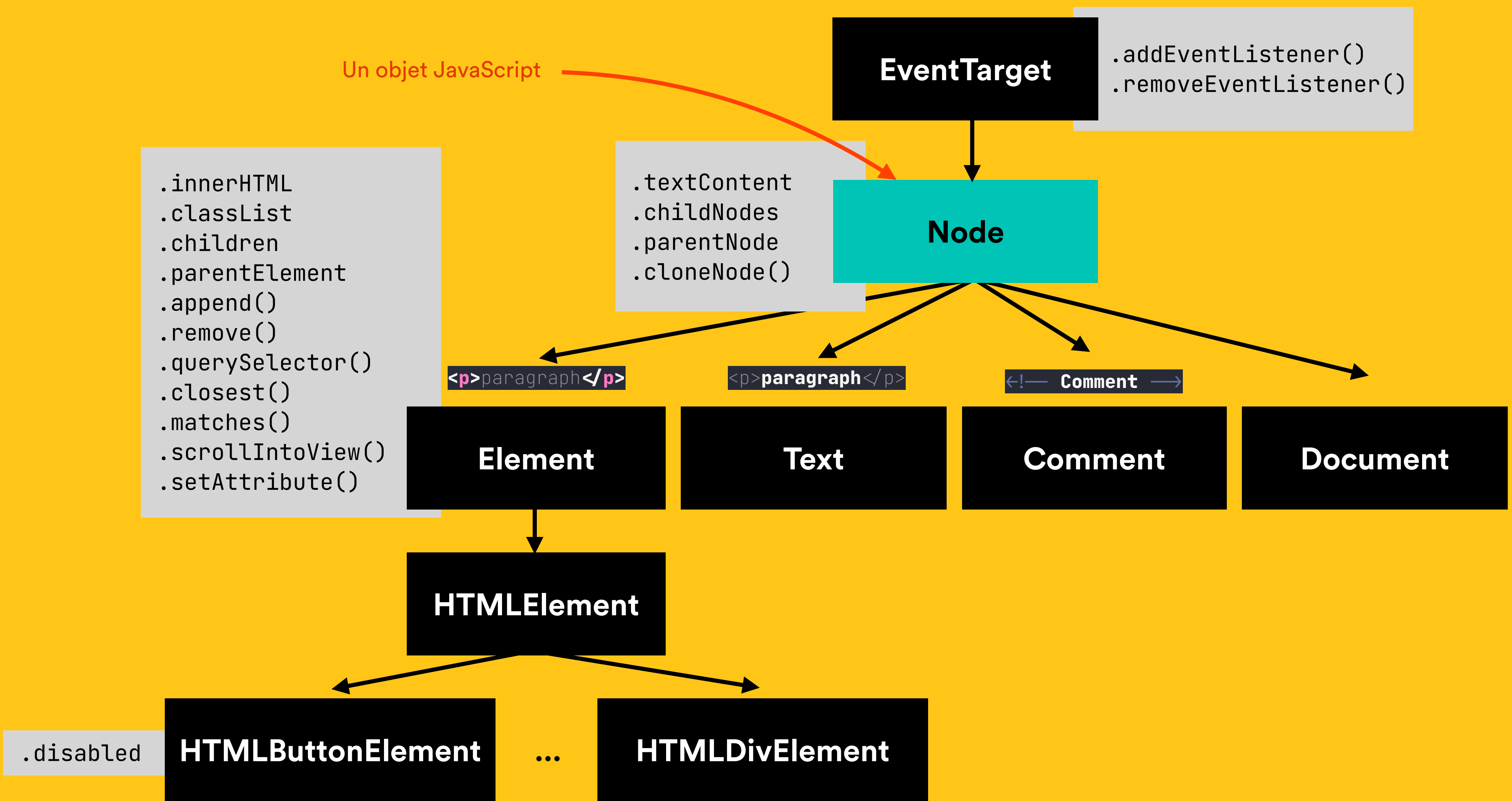
HTMLElement

`.disabled`

HTMLButtonElement

...

HTMLDivElement



Héritage des propriétés et des méthodes!

Par exemple:
HTMLElement à accès aux méthodes `cloneNode()` ou `closest()`