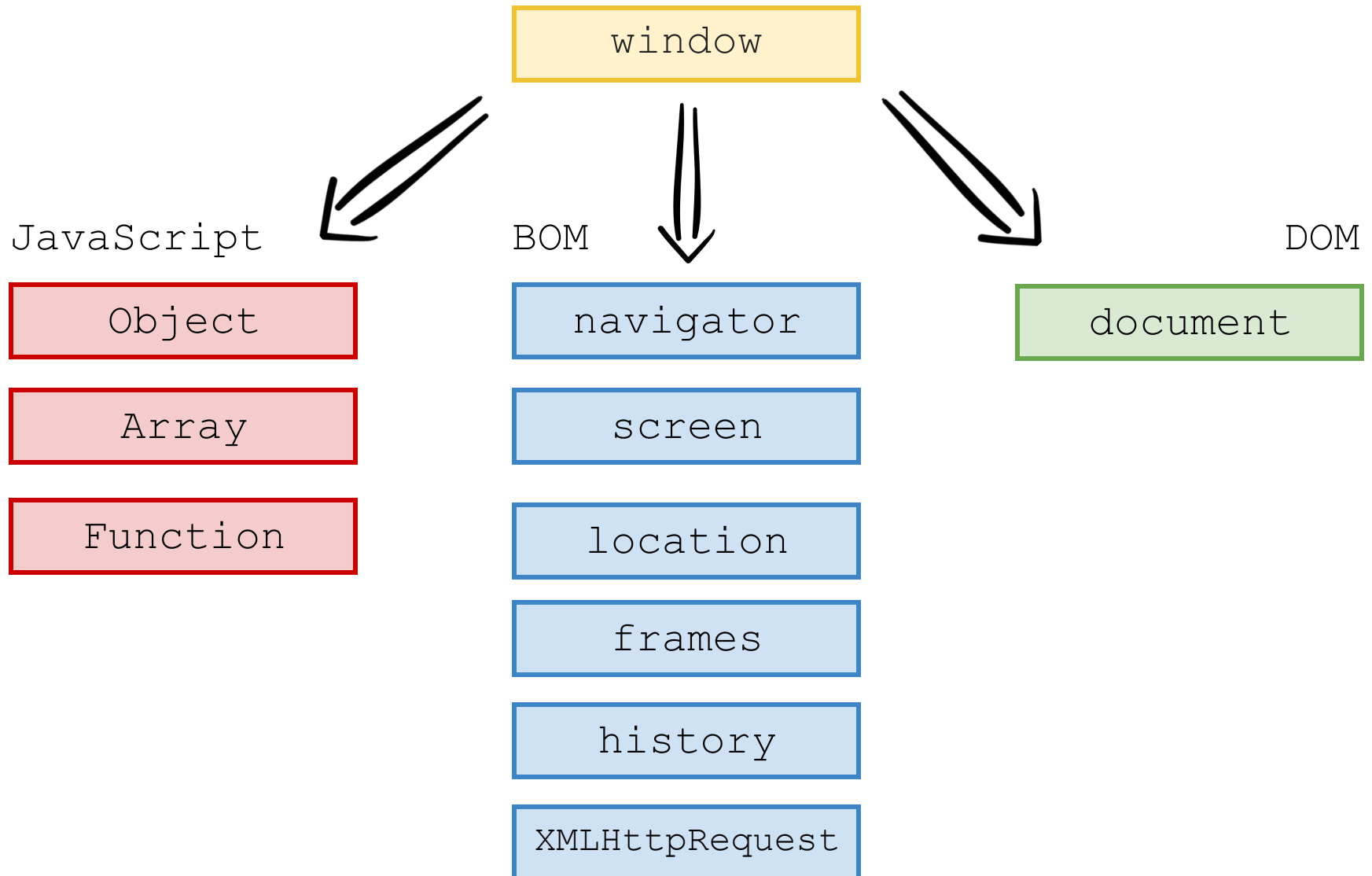


Programmation web - Client riche

Le Document Object Model (DOM)

L'environnement du navigateur



JavaScript

Object

Array

Function

...

C'est ce qu'on a vu au cours précédent :
les objets et fonctions qui sont le coeur
du langage JavaScript (communs à tous
les environnements)

BOM

navigator

screen

location


frames

history

XMLHttpRequest

Le Browser Object Model regroupe des API propres au navigateur. Elles nous permettent d'obtenir des détails sur l'OS, le navigateur et l'écran de l'utilisateur. Mais aussi de manipuler l'historique du navigateur ou de faire des requêtes sur le réseau.

DOM



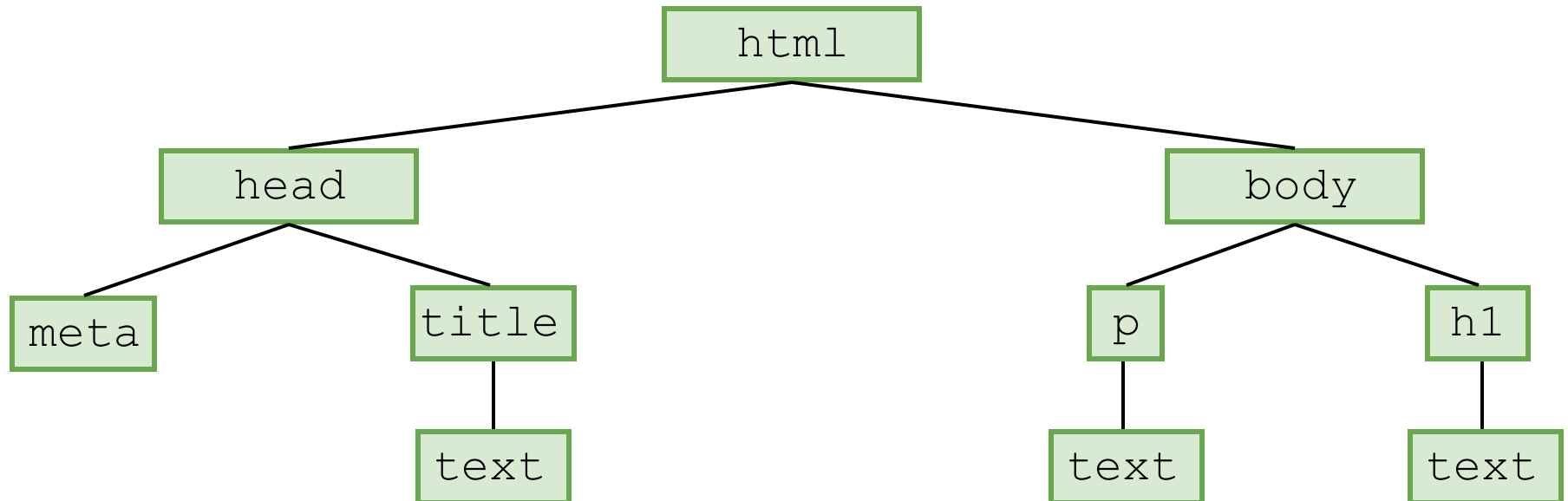
document

Le Document Object Model est une représentation de la structure HTML de la page sous forme d'arbre d'objets représentant les différents noeuds du document

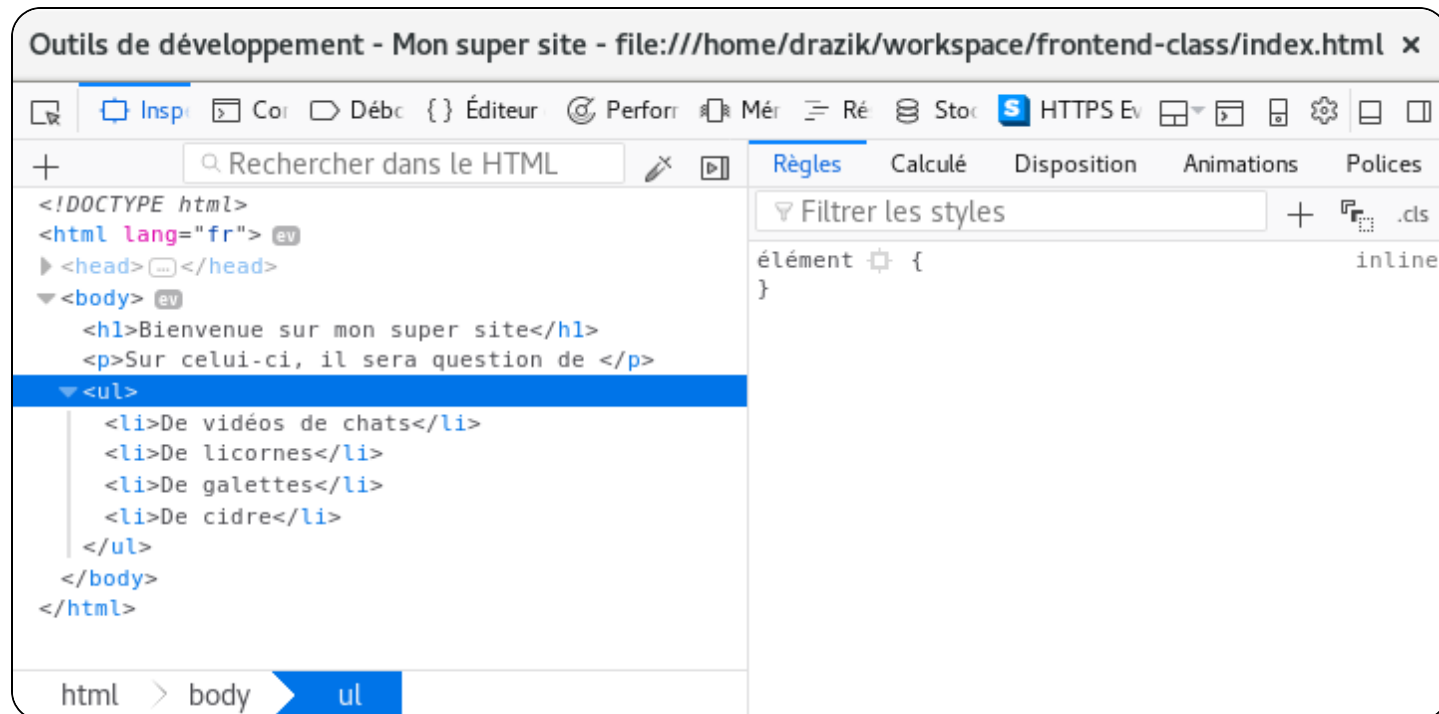
Le DOM : définition

*“ Le Document Object Model ou DOM (pour modèle objet de document) est une **interface de programmation pour les documents HTML, XML et SVG**. Il fournit une **représentation structurée** du document sous forme d'un **arbre** et définit **la façon dont la structure peut être manipulée** par les programmes, en termes de **style** et de **contenu**. Le DOM représente le document comme un ensemble de **nœuds** et d'**objets** possédant des propriétés et des méthodes. Les nœuds peuvent également avoir des gestionnaires d'événements qui se déclenchent lorsqu'un événement se produit. Cela permet de **manipuler des pages web grâce à des scripts et/ou des langages de programmation**.*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mon super site</title>
  </head>
  <body>
    <h1>Bienvenue sur mon super site</h1>
    <p>Ceci est un paragraphe</p>
  </body>
</html>
```



Il y a un inspecteur de DOM dans les outils de développement de votre navigateur (**F12** ou **CTRL+SHIFT+I**, pour rappel)

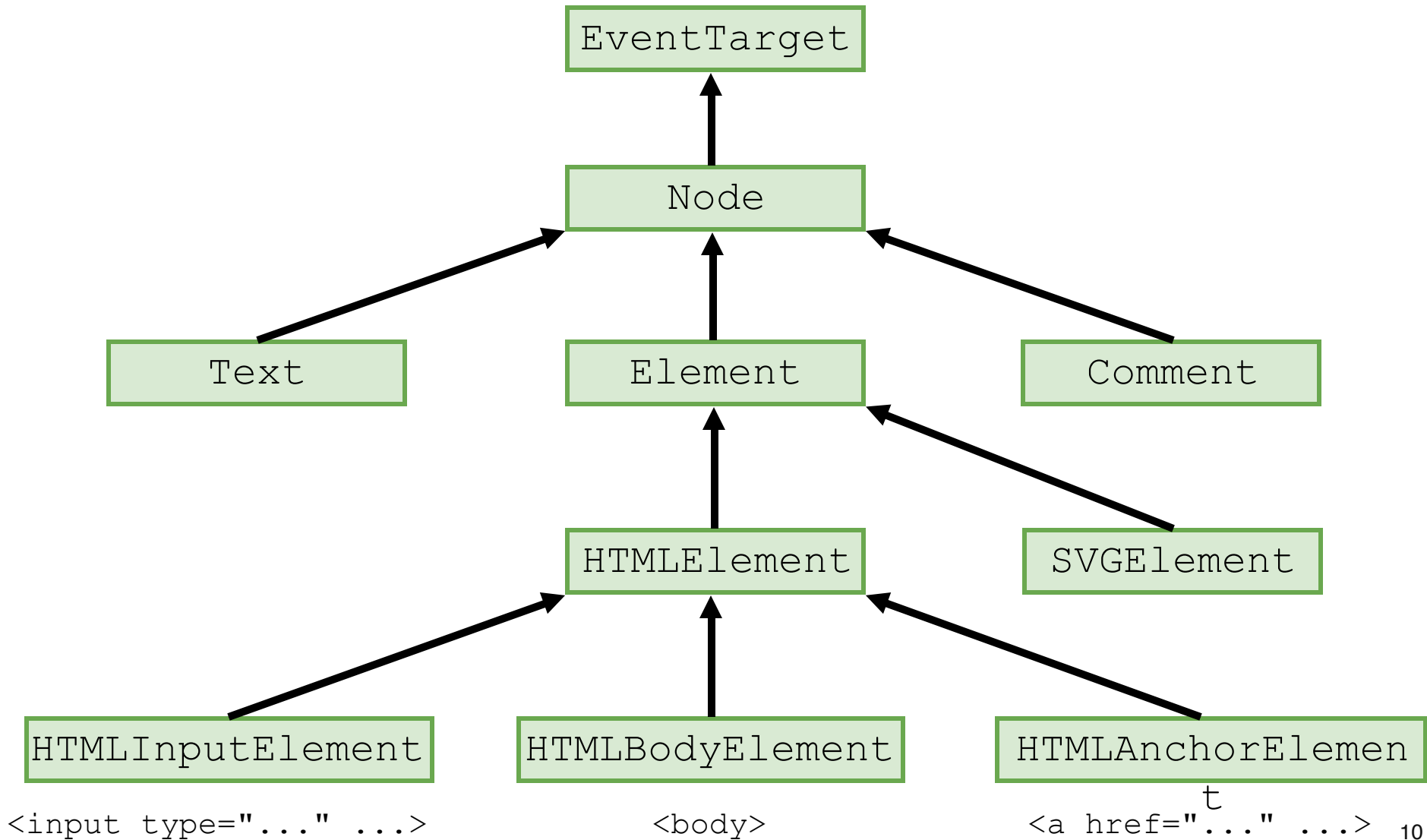



```
document.documentElement  
// => <html> (HTMLHtmlElement)
```

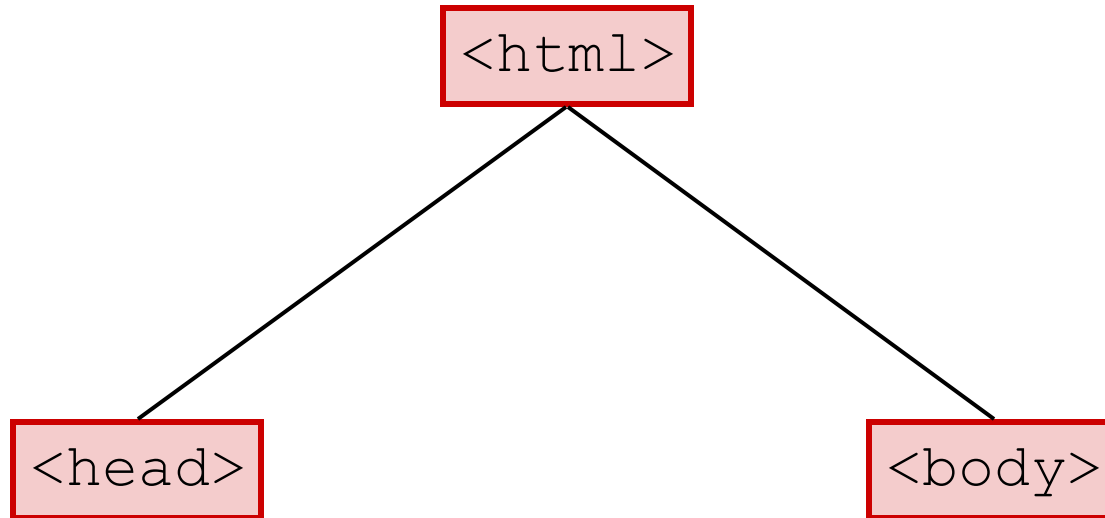
```
document.head  
// => <head> (HTMLHeadElement)
```

```
document.body  
// => <body> (HTMLBodyElement)
```

Les différents types d'éléments



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mon super site</title>
  </head>
  <body>
    <div>Premier enfant</div>
    <div>
      <div>Another</div>
      <div>div</div>
      <div>in paradise</div>
    </div>
    <div>Troisième enfant</div>
  </body>
</html>
```

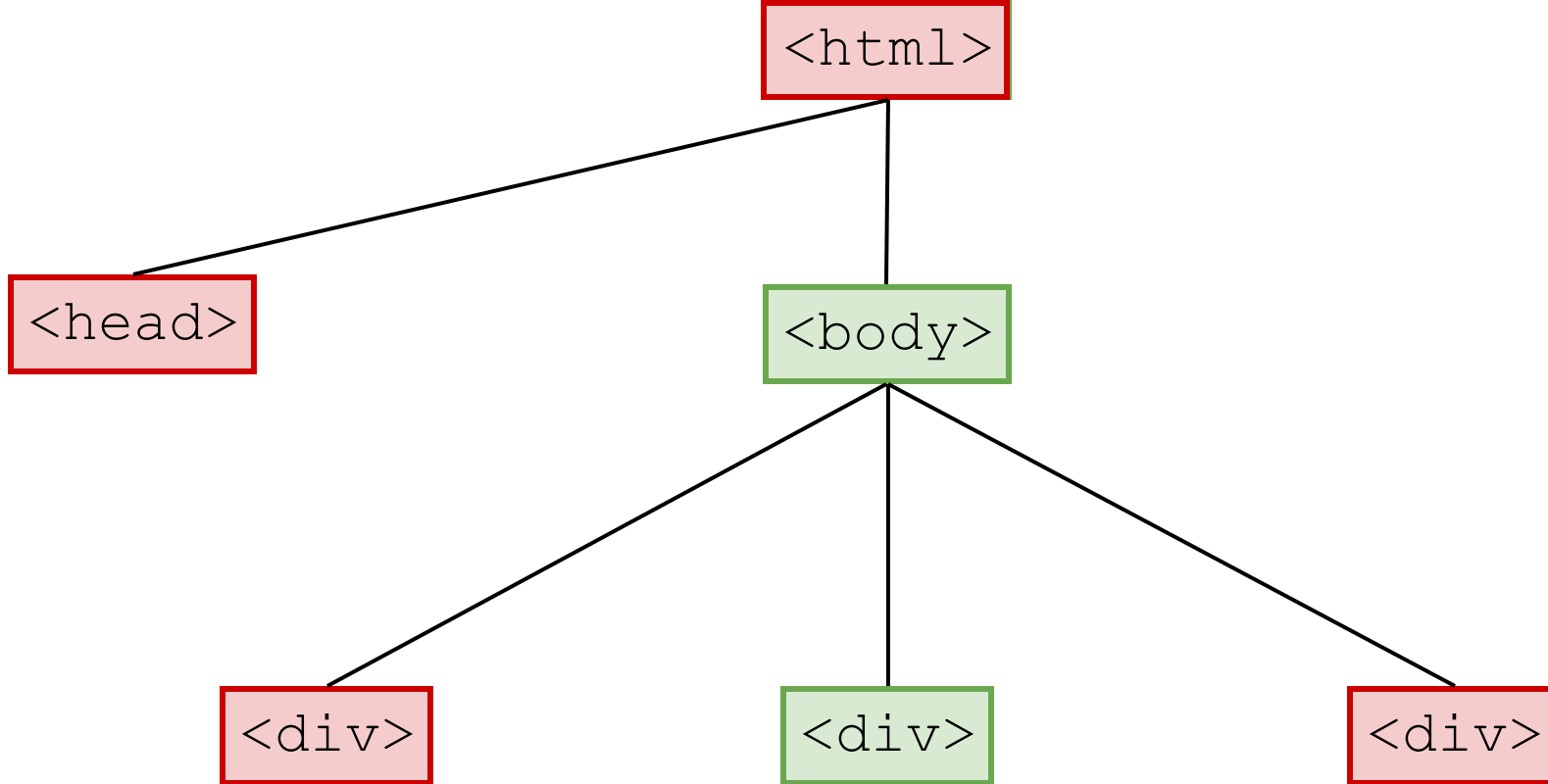


`document.documentElement`

`document.documentElement.children`

`document.documentElement.firstChild`

`document.documentElement.lastElementChild`



`document.body.parentElement`

`document.body.children[1].previousElementSibling`

`document.body.children[1].nextElementSibling`

`document.body.children[1].parentElement.previousElementSibling`

Rechercher des éléments dans le DOM

```
<h1>Bienvenue sur mon site web</h1>
<p>Sur ce site il est question des choses suivantes :</p>
<ul>
  <li>De vidéos de chats</li>
  <li id="unicorns">De licornes</li>
  <li class="important">De galettes</li>
  <li class="important">De cidre</li>
</ul>
```

```
document.querySelector("h1")
```

```
document.querySelector("li")
```

```
document.querySelector("#unicorns")
```

```
document.querySelector(".important")
```

```
<h1>Bienvenue sur mon site web</h1>
<p>Sur ce site il est question des choses suivantes :</p>
<ul>
  <li>De vidéos de chats</li>
  <li id="unicorns">De licornes</li>
  <li class="important">De galettes</li>
  <li class="important">De cidre</li>
</ul>
```

```
document.querySelectorAll("li")
```

```
document.querySelectorAll("#unicorns")
```

```
document.querySelectorAll(".important")
```




*querySelectorAll ne renvoie pas un Array, mais une **NodeList**. Cet objet ressemble à un Array, mais n'en possède pas toutes les méthodes utiles. Il est toutefois possible de transformer une NodeList en Array :*

```
Array.from(document.querySelectorAll( ".important" ) )
```

Accéder aux propriétés d'un élément du DOM

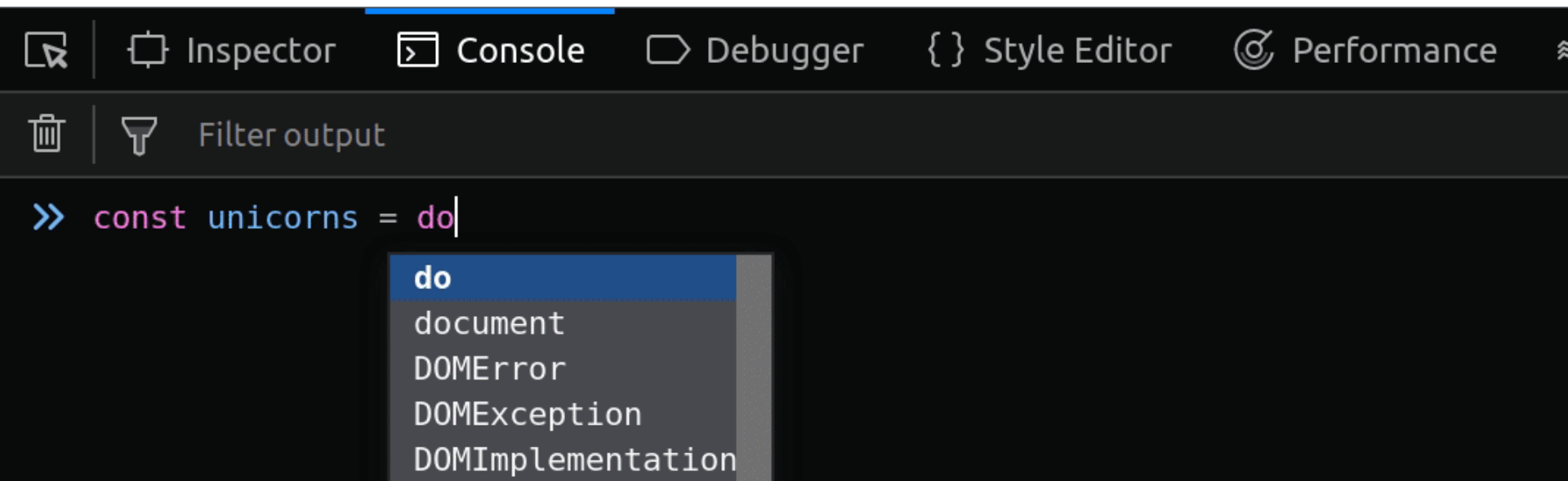
```
<ul>
  <li>De vidéos de chats</li>
  <li id="unicorns">
    De <span class="shiny">licornes</span>
  </li>
  <li class="important">De galettes</li>
  <li class="important">De cidre</li>
</ul>
```

```
const unicorns = document.querySelector("#unicorns")
```

```
console.log(unicorns.innerHTML)
// => "De <span class="shiny">licornes</span>"
```

```
unicorns.innerHTML = "De LICORNES !"
```

- De vidéos de chats
- De licornes
- De galettes
- De cidre



```
<ul>
  <li>De vidéos de chats</li>
  <li id="unicorns">
    De <span class="shiny">licornes</span>
  </li>
  <li class="important">De galettes</li>
  <li class="important">De cidre</li>
</ul>
```

```
const unicorns = document.querySelector("#unicorns")

console.log(unicorns.textContent) // => "De licornes"

unicorns.textContent = "De LICORNES !"
```

```
<body>
  <div id="container" name="container">
    <input
      id="login"
      name="login"
      placeholder="Enter your login"
    />
  </div>
</body>
```

```
const container = document.querySelector("#container")
```

```
container.id // => "container"
```

```
container.name
```

```
// => undefined
```

```
// name n'est pas une propriété standard sur <div>
```

```
const loginInput = document.querySelector("#login");
```

```
loginInput.id // => "login"
```

```
loginInput.name // => "login"
```

```
loginInput.placeholder // => "Enter your login"
```



On peut mettre n'importe quel attribut sur un élément HTML. Mais seuls ceux listés comme étant valides pour cet élément dans la spécification seront automatiquement transformés en propriétés de l'objet représentant l'élément en JS

```
<div id="container" data-current-state="loading">  
  Loading...  
</div>
```

```
const container = document.querySelector("#container")  
container.dataset.currentState // => "loading"  
container.dataset.currentState = "loaded"  
container.dataset.currentState // => "loaded"
```



**Créer, ajouter, modifier,
supprimer un élément du DOM**

```
const item = document.createElement("li")  
item.innerHTML = "De JavaScript"
```

`node.append(...nodes)`

`node.prepend(...nodes)`

`node.before(...nodes)`

`node.after(...nodes)`

`node.replaceWith(...nodes)`

```
<h1>Bienvenue !</h1>
<ul>
  <li>De tennis</li>
  <li>De vidéos de chats</li>
  <li>De licornes</li>
  <li>De galettes</li>
  <li>De cidre</li>
  <li>De JavaScript</li>
</ul>
<p>À bientôt !</p>
```

```
const list = document.querySelector("ul")
const heading = document.createElement("h1")
heading.textContent = "Bienvenue !"
list.before(heading)

const firstItem = document.createElement("li")
firstItem.textContent = "De tennis"
list.prepend(firstItem)

const lastItem = document.createElement("li")
lastItem.textContent = "De JavaScript"
list.prepend(lastItem)

const p = document.createElement("p")
p.textContent = "À bientôt !"
list.prepend(p)
```

```
const list = document.querySelector("ul")
const lastItem = list.lastElementChild

const clone = lastItem.cloneNode()
clone.outerHTML // => "<li></li>"

const cloneDeep = lastItem.cloneNode(true)
cloneDeep.outerHTML
// => "<li>De JavaScript</li>"

cloneDeep.textContent += " (cloned)"
list.append(cloneDeep)
```

```
const list = document.querySelector("ul")  
const lastItem = list.lastElementChild  
  
lastItem.remove()
```

Modifier les classes d'un élément du DOM

```
<p class="alert alert-error">
  Lorem ipsum dolor sit amet consectetur
</p>
```

```
const alert = document.querySelector(".alert")

alert.className // => "alert alert-error"

// Suppression de la classe "alert-error"
alert.className = alert.className.split(" ")[0]

alert.className // => "alert"
```



```
<p class="alert alert-error">
  Lorem ipsum dolor sit amet consectetur
</p>
```

```
const alert = document.querySelector(".alert")

for(const className of alert.classList) {
  console.log(className)
}
// => "alert" "alert-error"
```

```
alert.classList.contains("alert-error")
alert.classList.remove("alert-error")
alert.classList.add("alert-warning")
alert.classList.toggle("alert-warning")
```

**Modifier le style « inline »
d'un élément du DOM**

```
<p style="font-size: 3rem; color: red;">  
  Lorem ipsum dolor sit amet consectetur  
</p>
```

```
const alert = document.querySelector("p")
```

```
alert.style.fontSize // => "3rem"
```

```
alert.style.color // => "red"
```

```
alert.style.fontWeight = "bold"
```

```
alert.style.backgroundColor = "black"
```

Des questions ?



TD