DOM : Parcourir un document

Maintenant que nous savons attaché un script à notre page html, nous pouvons utiliser le DOM pour se balader dans notre document (notre page html).

Se balader grâce aux classes css

Immaginons la page suivante :

```
<html>
  <head></head>
  <body>
    <div class="ma-1er-div">
      <p class="mon-paragraph">Coucou les amis</p>
      <p class="mon-paragraph">Coucou les amis</p>
      <p class="mon-paragraph">Coucou les amis</p>
    </div>
    <script src="./monscript.js">
  </body>
</html>
```

Nous pouvons utiliser la fonction `document.getElementByClassName` ou bien `document.getElementsByClassName` afin de récupérer un ou plusieur éléments correspondant à la classe donnée :

```
// monscript.js
const p = document.getElementByClassName('mon-paragraph')
// Attention document.getElementByClassName peut retourner null si
// aucun élément ne correspond
const t = document.getElementByClassName('existe-pas') // null

// Il suffit d'utiliser une condition afin de s'assurer que l'élément
// est bien présent :
if (t) {
  // J'ai un élément !
} else {
  // L'élément n'existe pas dans la page !
}

// Il est aussi possible de récupérer une liste d'éléments (array)
const paragraphs = document.getElementsByClassName('mon-paragraph')
```

> Aujourd'hui, getElementByClassName, n'est plus trop utilisé ...

Sa balader avec les id, et les noms de balise

Très similairement à la fonction `document.getElementsByClassName`, nous pouvons aussi utiliser :

- `getElementByTagName` : Récupére un élément par le nom de sa balise (ex: `document.getElementByTagName('h1')`)

- `getElementsByTagName` : Récupére une d'éléments par le nom de leurs balise (ex: `document.getElementsByTagName('p')`)
- `getElementById` : Récupére un élément par son id (ex : `document.getElementById('mon-id')`)

Ces fonctions ne sont quasiment plus utilisé, aujourd'hui il éxiste plus puissant : `querySelector`

Se balader avec le querySelector

La fonction `document.querySelector` et `document.querySelectorAll` permet de récupérer un ou plusieurs éléments en utilisant un **sélécteur CSS**.

> Rappel : Un sélécteur css est ce que vous utiliser pour styliser un éléments ex
>
> ```
> .ma-class #mon-id .ma-class .mon-autre-class div .ma-class #mon-id;
> ```

De la même que le CSS, querySelector fonctionne similairement :

```
// Je séléctionne le premier élement correspondant à
// mon sélécteur css : la class mon-paragraph
const p = document.querySelector('.mon-paragraph')

// Il exsiste la possibilité de récupérer TOUT les éléments
// correspondant à une sélécteur en utilisant :
// Ici on récupére une liste (array) de tout les paragraphes
const allParagraph = document.querySelectorAll('p')
```

Les signatures

Voici les signatures de `document.querySelector` et `document.querySelectorAll` :

```
document.querySelector(selector: string): DOMElement
```

```
document.querySelectorAll(selector: string): DOMElement[]
```

Le `DOMElement`

Lorsque l'on utilise la fonction `document.querySelector` et aussi `document.querySelectorAll` nous récupérons un ou plusieurs `DOMElement`

Ce `DOMElement` représente une balise html. Nous pouvons utiliser cet élément pour naviguer dans nore page html

```
// On récupére le premier paragraph de notre page
const p = document.querySelector('p')

// ici la constante p est un objet DOMElement
// Nous pouvons naviguer entre ces « cousins » (sibliings)
const cousin1 = p.nextElementSibling // Attention peut retourner null
const cousin2 = p.previousElementSibling // Attention peut retourner null

// Nous pouvons récuperer les enfants :
const children = p.children // Ici nous récupérons une list de DOMElement (array)
```

Récupérer les attributs d'une balise

Lorsque nous avons un élément (`DOMElement` => balise html) nous pouvons utiliser une suite de fonction afin de manipuler les attributs de notre balise :

- `element.hasAttribute` : Test si un attribut est présent dans la balise
- `element.getAttribute` : Récupére la valeur d'un attribut
- `element.setAttribute` : Change la valeur d'un attribut
- `element.removeAttribute` : Supprime un attribut de notre élément

```
// Je séléctionne le premier paragraph
const p = document.querySelector('p')

// Je souhaiterais savoir si mon paragraph possède l'attribute class ?
p.hasAttribute('class') // true si il posséde un class, false sinon
// Je souhaiterais obtenir les classes
p.getAttribute('class') // toutes les classe présent dans l'attribut
// Je peut changer les class de mon paragraph
p.setAttribute('class', 'big-text super-red')
// Je peut supprimer l'attribut class
p.removeAttribute('class')
```

Jouer et manipuler l'attribut class

Il éxiste dans un `DOMElement` un autre objet nommé la `ClassList`. Cet objet nous permet de manipuler l'attribut class de notre élément plus facilement :

Pour accéder à cet objet `ClassList` nous utilisons `element.classList` et nous pouvons réaliser les opération suivante :

```
// Je séléctionne le premier paragraph
const p = document.querySelector('p')
j
// Je souhaiterais savoir si mon paragraph possède la class super-gros
p.classList.has('super-gros') // true si super-gros est dans la class
p.classList.remove('super-gros') // on supprime la class super-gros
p.classList.replace('super-gros', 'super-petit') // on remplace une class css
p.classList.toggle('super-gros') // Si super-gros éxiste on enléve sinon on
ajoute
p.classList.add('super-rouge') // Ajoute la class super-rouge
```

Quelques signatures

```
document : Document
document.querySelector(selector: string): DOMElement | null
document.querySelectorAll(selector: string): DOMElement[]

DOMElement.nextElementSibling : DOMElement | null
DOMElement.previousElementSibling : DOMElement | null
DOMElement.children : DOMElement[]

DOMElement.hasAttribute(name: string): boolean
DOMElement.getAttribute(name: string): string | null
DOMElement.setAttribute(name: string, value: string): void
DOMElement.removeAttribute(name: string): void

DOMElement.classList : ClassList

DOMElement.classList.has(cssClass: string): boolean
DOMElement.classList.remove(cssClass: string): void
DOMElement.classList.replace(cssClass: string, cssClassReplacement: string): void
DOMElement.classList.toggle(cssClass: string): void
DOMElement.classList.add(cssClass: string): void
```