COLLÈGE
MONTMORENCY

Notions d’algorithme et Manipulation des attributs de balise via le DOM

Programmation interactive et base de données

420 V25 MO

Automne 2022

Plan de cette séance…

- Manipulation des attributs de balise via le DOM
- Utilisation de setTimeout
- Utilisation de setInterval
- Utilisation de eventListener

Manipulation des attributs de balise via le DOM

- La propriété **attributes** de l'interface **Element** renvoie la liste des attributs d'un (nœud) élément spécifié. La liste d'attributs est renvoyée sous la forme « clef / valeur ».
- Si on ne souhaite récupérer que les noms des attributs d'un élément, on peut également utiliser la méthode **getAttributeNames().**
- Pour ne récupérer que la valeur d'un attribut donné pour un élément, on utilisera la méthode **getAttribute().**

- Syntaxe :

```
element.attributes
element.getAttributeNames()
element.getAttribute('nom de l'attribut')
```

Manipulation des attributs de balise via le DOM

- La méthode **hasAttribute()** de l'interface **Element** nous permet de tester la présence d'un attribut en particulier pour un élément. Cette méthode prend en argument le nom de l'attribut qu'on recherche et renvoie la valeur booléenne **true** si l'élément possède bien cet attribut ou **false** sinon.
- Pour ajouter un nouvel attribut ou changer la valeur d'un attribut existant pour un élément, nous allons cette fois-ci utiliser la méthode **setAttribute()** à laquelle on va passer en arguments le nom et la valeur de l'attribut à ajouter ou à modifier

- Syntaxe :

```
element.hasAttribute('nom de l'attribut')
element.setAttribute('nom de l'attribut', 'valeur')
```

Manipulation des nœuds du DOM

- On peut changer le contenu, le style d'un élément du DOM.
- On peut aussi ajouter, supprimer ou remplacer un élément du DOM.
- Exemple ajouter un élément au DOM:

```
const para = document.createElement("p");
para.setAttribute("id", "new");
const node = document.createTextNode(" C'est un nouveau élément.");
para.appendChild(node);
const parent = document.querySelector("body");
parent.appendChild(para);
```

Manipulation des nœuds du DOM

- Exemple supprimer un élément du DOM:

```
const parent = document.querySelector("body");

let child = document.getElementById("new");
parent.removeChild(child);
```

- Exemple remplacer un élément du DOM:

```
let para = document.createElement("p");
let node = document.createTextNode("un autre nouveau élément sans id");
para.appendChild(node);
const parent = document.querySelector("body");

let child = document.getElementById("new");
parent.replaceChild(para, child);
```

Utilisation de eventListener

- La méthode addEventListener() nous permet d'ajouter de nombreux événements au même élément, sans écraser les événements existants
- La méthode removeEventListener() supprime les gestionnaires d'événements qui ont été attachés avec la méthode addEventListener()

```
function displayAlert() {
  alert('tu as cliqué sur le 1er bouton transformer');
}

document.getElementById("minusculeButton").addEventListener("click", displayAlert);
```

Utilisation de setTimeout

- La méthode setTimeout() appelle une fonction après un nombre spécifié de millisecondes.
- Syntaxe :

```
setTimeout(function, milliseconds, param1, param2, ...)
```

- Exp:

```
setTimeout(() => console.log(`votre pizza avec du thon et
des olives est prête 🍕`), 3000
);

console.log("waiting…");
```

Utilisation de setInterval

- La méthode setInterval() appelle une fonction après un nombre spécifié de millisecondes d'une façon répétitive.
- Syntaxe :

```
setInterval(function, milliseconds, param1, param2, ...)
```

- Exp:

```
setInterval(() => console.log(`votre pizza avec du thon et
des olives est prête 🍕`), 1000
);

console.log("waiting…");
```