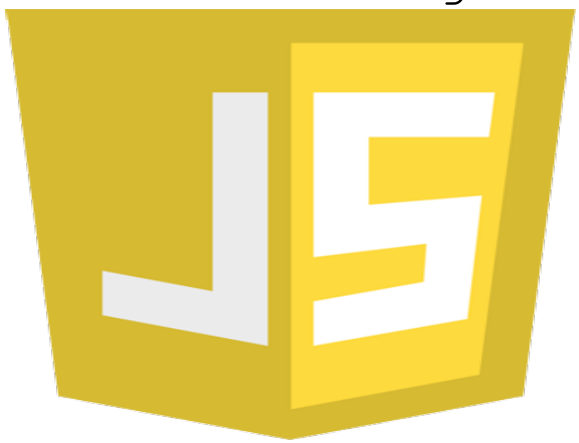
JavaScript : le DOM

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

Plan

Plan

JavaScript

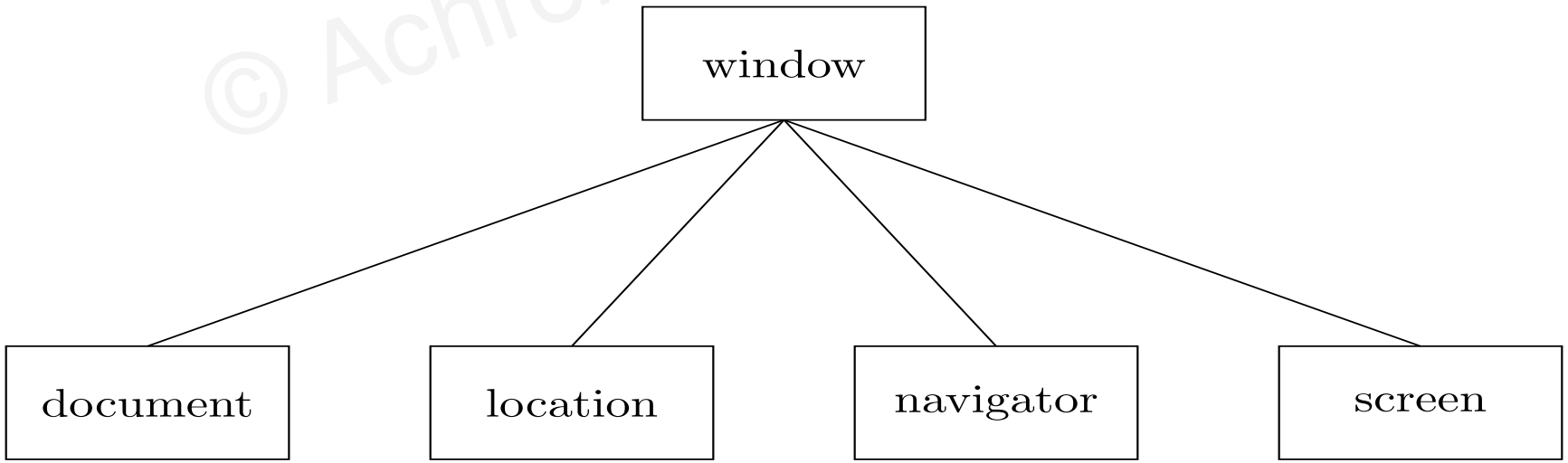
JavaScript nous permet, via des objets prédéfinis, d'avoir des informations sur

- le visiteur
- son navigateur
- la structure de la page (document) qu'il consulte
- son écran
- ...

JavaScript

JavaScript nous permet, via des objets prédéfinis, d'avoir des informations sur

- le visiteur
- son navigateur
- la structure de la page (document) qu'il consulte
- son écran
- ...



JavaScript

L'objet `Window` contient

- quatre autres objets importants : `document`, `location`, `navigator`, `screen`
- des méthodes basiques comme `alert`, `confirm`
- les objets comme `String`, `Date`, `Math`...
- toute variable, objet... définis par le développeur
- ...

L'objet `document` contient

- `contentType` : text/html, text/xml...
- toutes les balises HTML constituant le document (DOM)
- les attributs et leurs valeurs
- ...

L'objet `document` contient

- `contentType` : text/html, text/xml...
- toutes les balises HTML constituant le document (DOM)
- les attributs et leurs valeurs
- ...

L'objet `location` contient

- `hostname` : localhost...
- `port` : 3000
- `protocol` : http
- ...

L'objet `navigator` contient

- `language` : fr_FR...
- `connection` : 4G
- `cookieEnabled` : true
- ...

L'objet `navigator` contient

- `language` : fr_FR...
- `connection` : 4G
- `cookieEnabled` : true
- ...

L'objet `screen` contient

- `height`
- `width`
- `orientation`
- ...

JavaScript

DOM : Document Object Model

- Interface de programmation (API) pour les documents **XML** et **HTML**
- Graphiquement, ça correspond à un arbre dont les nœuds sont les balises **HTML**
- Avant standardisation par le **W3C**, chaque navigateur avait son propre **DOM**.
- Utilisé par les scripts pour modifier un document **HTML** en
  - ajoutant un nœud
  - modifiant un autre
  - remplaçant un premier par un deuxième
  - supprimant un autre

JavaScript

Plusieurs méthodes pour récupérer un élément HTML

- selon l'identifiant : `getElementById()`
- selon le nom de la classe : `getElementsByClassName()`
- selon le nom de la balise : `getElementsByTagName()`
- selon l'attribut `name` : `getElementsByName()`
- selon un sélecteur CSS : `querySelectorAll()` ou `querySelector()`

Considérons la page HTML suivante

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
      =1.0">
    <title>My JS Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id=container>
        Considérons les paragraphes suivants :
        <p class=blue name=parConteneur> bonjour </p>
        <p class=red name=parConteneur> bonsoir, voici
            <a href="http://www.lsis.org/elmouelhia/"> ma page </a>
        </p>
        <p class=blue name=parConteneur> salut </p>
    </div>
    <script src="script.js"></script>
</body>

</html>
```

JavaScript

Contenu de `style.css`

```
.red {
  color: red;
}

.blue {
  color: blue;
}

.green {
  color: green;
}
```

JavaScript

Récupérer un élément selon son `id`, on ajoute dans `script.js`

```
var container = window.document.getElementById("container");
console.log(container.innerHTML);
```

JavaScript

Récupérer un élément selon son `id`, on ajoute dans `script.js`

```
var container = window.document.getElementById("container");
console.log(container.innerHTML);
```

Ou sans passer par l'objet `window`

```
var container = document.getElementById("container");
console.log(container.innerHTML);
```

JavaScript

Récupérer un élément selon son `id`, on ajoute dans `script.js`

```
var container = window.document.getElementById("container");
console.log(container.innerHTML);
```

Ou sans passer par l'objet `window`

```
var container = document.getElementById("container");
console.log(container.innerHTML);
```


Remarque

Il ne faut pas utiliser `#` dans `getElementById()`.

JavaScript

Récupérer des éléments selon le nom de la balise

```
var paragraphes = document.getElementsByTagName("p");

for (let paragraphe of paragraphes) {
    console.log(paragraphe.innerHTML);
}
```

JavaScript

Récupérer des éléments selon le nom de la balise

```
var paragraphes = document.getElementsByTagName("p");

for (let paragraphe of paragraphes) {
    console.log(paragraphe.innerHTML);
}
```

Ou encore

```
var paragraphes = container.getElementsByTagName("p");

for (let paragraphe of paragraphes) {
    console.log(paragraphe.innerHTML);
}
```

Récupérer un élément selon sa classe

```
var bleus = document.getElementsByClassName("blue");

for (let bleu of bleus) {
    console.log(bleu.innerHTML);
}
```

Récupérer un élément selon sa classe

```
var bleus = document.getElementsByClassName("blue");

for (let bleu of bleus) {
    console.log(bleu.innerHTML);
}
```

Ou aussi

```
var bleus = container.getElementsByClassName("blue");

for (let bleu of bleus) {
    console.log(bleu.innerHTML);
}
```

Récupérer un élément selon sa classe

```
var bleus = document.getElementsByClassName("blue");

for (let bleu of bleus) {
    console.log(bleu.innerHTML);
}
```

Ou aussi

```
var bleus = container.getElementsByClassName("blue");

for (let bleu of bleus) {
    console.log(bleu.innerHTML);
}
```

Remarque

Il ne faut pas utiliser `.` dans `getElementsByClassName()`.

JavaScript

Récupérer un élément selon la valeur de son attribut `name`

```
var parConteneurs = document.getElementsByName("parConteneur");
console.log(parConteneurs);
```

JavaScript

Récupérer un élément selon la valeur de son attribut `name`

```
var parConteneurs = document.getElementsByName("parConteneur");
console.log(parConteneurs);
```

Ou aussi

```
for (let parConteneur of parConteneurs) {
    console.log(parConteneur.innerHTML);
}
```

JavaScript

Récupérer un élément selon un sélecteur CSS 3

```
var pbleus = document.querySelectorAll("p.blue");
for (let bleu of pbleus) {
    console.log(bleu.innerHTML);
}
```

JavaScript

Récupérer le premier élément selon un sélecteur CSS 3

```
var pbleu = document.querySelector("p.blue");
console.log(pbleu.innerHTML);
```

JavaScript

Récupérer le premier élément selon un sélecteur CSS 3

```
var pbleu = document.querySelector("p.blue");
console.log(pbleu.innerHTML);
```

Ceci déclenche une erreur

```
var prouges = document.querySelector("p.red");
for (let rouge of prouges) {
    console.log(rouge.innerHTML);
}
```

JavaScript

Récupérer l'attribut d'un élément HTML

```
var lien = document.querySelector('a');

var href = lien.getAttribute('href');
console.log(href);
```

JavaScript

Récupérer l'attribut d'un élément HTML

```
var lien = document.querySelector('a');
lien.setAttribute('href', 'https://www.w3schools.com');

console.log(lien);
// affiche https://www.w3schools.com
```

JavaScript

Supprimer l'attribut d'un élément HTML

```
var lien = document.querySelector('a');
lien.removeAttribute('href');
```

JavaScript

Pour certains attributs comme `href`, on peut accéder directement à la valeur via son nom

```
var lien = document.querySelector('a');
var href = lien.href;
console.log(href);
```

JavaScript

Pour certains attributs comme `href`, on peut accéder directement à la valeur via son nom

```
var lien = document.querySelector('a');
var href = lien.href;
console.log(href);
```

Modifier l'attribut d'un élément HTML

```
lien.href = 'https://www.w3schools.com';
console.log(lien);
```

JavaScript

Remarque

Les attributs `class` et `for` sont des mots-clés réservés en **JavaScript**, on ne peut donc les utiliser pour modifier leurs valeurs **HTML**.

JavaScript

Remarque

Les attributs `class` et `for` sont des mots-clés réservés en **JavaScript**, on ne peut donc les utiliser pour modifier leurs valeurs **HTML**.

Solution

- Utiliser `className` ou `classList` pour l'attribut `class`
- Et `forHtml` pour l'attribut `for`

JavaScript

Récupérer la classe d'un élément HTML

```
var container = document.getElementById("container");
console.log(container.className);
```

JavaScript

Récupérer la classe d'un élément HTML

```
var container = document.getElementById("container");
console.log(container.className);
```

Modifier une classe d'un élément HTML

```
container.className = "blue";
console.log(container.className);
```

JavaScript

Récupérer la classe d'un élément HTML

```
var container = document.getElementById("container");
console.log(container.className);
```

Modifier une classe d'un élément HTML

```
container.className = "blue";
console.log(container.className);
```


Ajouter une nouvelle classe à un élément HTML

```
container.className += " red";
console.log(container.className);
```

JavaScript

Récupérer la liste des classes d'un élément HTML

```
var container = document.getElementById("container");
var list = container.classList;
console.log(list);
```

JavaScript

Récupérer la liste des classes d'un élément HTML

```
var container = document.getElementById("container");
var list = container.classList;
console.log(list);
```

Ajouter une classe à un élément HTML

```
var container = document.getElementById("container");
var list = container.classList;
list.add('green');
console.log(list);
```

JavaScript

Récupérer la liste des classes d'un élément HTML

```
var container = document.getElementById("container");
var list = container.classList;
console.log(list);
```

Ajouter une classe à un élément HTML

```
var container = document.getElementById("container");
var list = container.classList;
list.add('green');
console.log(list);
```

Ajouter des classes à un élément HTML

```
var container = document.getElementById("container");
var list = container.classList;
list.add('red',"blue");
console.log(list);
```

JavaScript

Supprimer une classe d'un élément HTML

```
list.remove('red');
console.log(list);
```

JavaScript

Supprimer une classe d'un élément HTML

```
list.remove('red');
console.log(list);
```

Supprimer plusieurs classes d'un élément HTML

```
list.remove('red', 'green');
console.log(list);
```

JavaScript

Supprimer une classe si elle existe, sinon l'ajouter

```
list.toggle('red');
console.log(list);
```

JavaScript

Supprimer une classe si elle existe, sinon l'ajouter

```
list.toggle('red');
console.log(list);
```

Remplacer une classe

```
list.replace('red', 'blue');
console.log(list);
```

JavaScript

Autres méthodes de `classList`

- `length` : retourne le nombre de classes.
- `contains(classe)` : retourne `true` si `classe` appartient à `classList`, `false` sinon.
- `item(i)` : retourne la classe d'indice `i` si `i` est inférieur à la longueur de la liste, `null` sinon.

JavaScript

Les attributs d'un formulaire

- `value` : pour récupérer la valeur saisie dans une zone de saisie (input, textarea...)
- `checked` : pour déterminer si une case à cocher ou un bouton radio a été coché ou non (`true` ou `false`)
- ...

JavaScript

Récupérer le contenu d'un élément HTML (y compris les balises)

```
var container = document.getElementById("container");
console.log(container.innerHTML);
```

JavaScript

Récupérer le contenu d'un élément HTML (y compris les balises)

```
var container = document.getElementById("container");
console.log(container.innerHTML);
```

Récupérer le contenu d'un élément HTML (sans les balises)

```
var container = document.getElementById("container");
console.log(container.textContent);
```

JavaScript

Modifier le contenu d'un élément HTML avec `innerHTML`

```
var container = document.getElementById("container");
container.innerHTML += "<p> hello </p>";
console.log(container.innerHTML);
```

JavaScript

Modifier le contenu d'un élément HTML avec `innerHTML`

```
var container = document.getElementById("container");
container.innerHTML += "<p> hello </p>";
console.log(container.innerHTML);
```

Récupérer le contenu d'un élément HTML avec `textContent`

```
var container = document.getElementById("container");
container.textContent += "<p> hello </p>";
console.log(container.textContent);
console.log(container.innerHTML);
```

JavaScript

Modifier la couleur

```
var container = document.getElementById("container");
container.style.color = "red";
```

JavaScript

Modifier la couleur

```
var container = document.getElementById("container");
container.style.color = "red";
```

Pour les propriétés CSS composées de deux mots séparés par -, il faut supprimer - mettre la propriété en camelCase

```
var container = document.getElementById("container");
container.style.backgroundColor = "red";
```

JavaScript

Modifier la couleur

```
var container = document.getElementById("container");
container.style.color = "red";
```

Pour les propriétés CSS composées de deux mots séparés par -, il faut supprimer - mettre la propriété en camelCase

```
var container = document.getElementById("container");
container.style.backgroundColor = "red";
```


Remarque

Un élément **HTML**, ayant classe `red` qui modifie la couleur de fond en rouge, n'a pas forcément la valeur `red` attribuée au `style.backgroundColor`

JavaScript

Récupérer le parent d'un élément HTML (l'objet)

```
var container = document.getElementById("container");
var parent = container.parentNode;
console.log(parent);
```

JavaScript

Récupérer le parent d'un élément HTML (l'objet)

```
var container = document.getElementById("container");
var parent = container.parentNode;
console.log(parent);
```

Récupérer le parent d'un élément HTML (le nom)

```
var container = document.getElementById("container");
var parent = container.parentNode;
console.log(parent.nodeName);
```

JavaScript

Récupérer le premier élément fils d'un élément HTML

```
var container = document.getElementById("container");
var premierFils = container.firstElementChild;
console.log(premierFils.nodeName);
```

JavaScript

Récupérer le premier élément fils d'un élément HTML

```
var container = document.getElementById("container");
var premierFils = container.firstElementChild;
console.log(premierFils.nodeName);
```

Récupérer le dernier élément fils d'un élément HTML

```
var container = document.getElementById("container");
var dernierFils = container.lastElementChild;
console.log(dernierFils.nodeName);
```

JavaScript

Récupérer le premier fils (élément, texte ou autre) d'un élément HTML

```
var container = document.getElementById("container");
var premierFils = container.firstChild;
console.log(premierFils.nodeName);
```

JavaScript

Récupérer le premier fils (élément, texte ou autre) d'un élément HTML

```
var container = document.getElementById("container");
var premierFils = container.firstChild;
console.log(premierFils.nodeName);
```

Récupérer le dernier fils (élément, texte ou autre) d'un élément HTML

```
var container = document.getElementById("container");
var dernierFils = container.lastChild;
console.log(dernierFils.nodeName);
```

JavaScript

Récupérer tous les enfants (élément, texte ou autre) d'un élément HTML

```
var enfants = container.childNodes;
for(let enfant of enfants)
    console.log(enfant);
```

JavaScript

Récupérer tous les enfants (élément, texte ou autre) d'un élément HTML

```
var enfants = container.childNodes;
for(let enfant of enfants)
    console.log(enfant);
```

Récupérer tous les éléments enfants d'un élément HTML

```
var enfants = container.children;
for(let enfant of enfants)
    console.log(enfant);
```

JavaScript

Enfants suivant et précédent

- `previousSibling`
- `nextSibling`
- `previousElementSibling`
- `nextElementSibling`

JavaScript

Enfants suivant et précédent

- `previousSibling`
- `nextSibling`
- `previousElementSibling`
- `nextElementSibling`

Attention

À chaque appel d'une de méthodes précédentes, le pointeur passe à l'élément suivant (ou précédent).

JavaScript

Étapes

- Créer l'élément
- Préparer ses attributs [et valeurs]
- L'ajouter au document

JavaScript

Créer un élément de type `p`

```
var par = document.createElement("p");
```

JavaScript

Créer un élément de type `p`

```
var par = document.createElement("p");
```

Préparer ses attributs [et valeurs]

```
par.id = "intro";
par.setAttribute("class", "blue");
var text = document.createTextNode("JS paragraph");
par.appendChild(text);
```

JavaScript

Créer un élément de type `p`

```
var par = document.createElement("p");
```

Préparer ses attributs [et valeurs]

```
par.id = "intro";
par.setAttribute("class", "blue");
var text = document.createTextNode("JS paragraph");
par.appendChild(text);
```

Ajouter l'élément au DOM

```
var container = document.getElementById("container");
container.appendChild(par);
```

JavaScript

Créer un élément de type `p`

```
var par = document.createElement("p");
```

Préparer ses attributs [et valeurs]

```
par.id = "intro";
par.setAttribute("class", "blue");
var text = document.createTextNode("JS paragraph");
par.appendChild(text);
```

Ajouter l'élément au DOM

```
var container = document.getElementById("container");
container.appendChild(par);
```

`appendChild()` ajoute l'élément comme dernier fils de conteneur

JavaScript

On peut aussi utiliser

- `.insertBefore(newnode, existingnode)` : pour insérer `newnode` avant `existingnode`

JavaScript

La méthode `insertAdjacentHTML(position, text)`

permet d'insérer un texte HTML (`text`), dans une position spécifiée (`position`)

- `afterbegin`,
- `afterend`,
- `beforebegin`,
- `beforeend`.

JavaScript

Exemple : HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title> float et clear </title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <p id=first>
      bonjour
    </p>
    <script  src="script.js"></script>
  </body>
</html>
```

Le fichier `style.css`

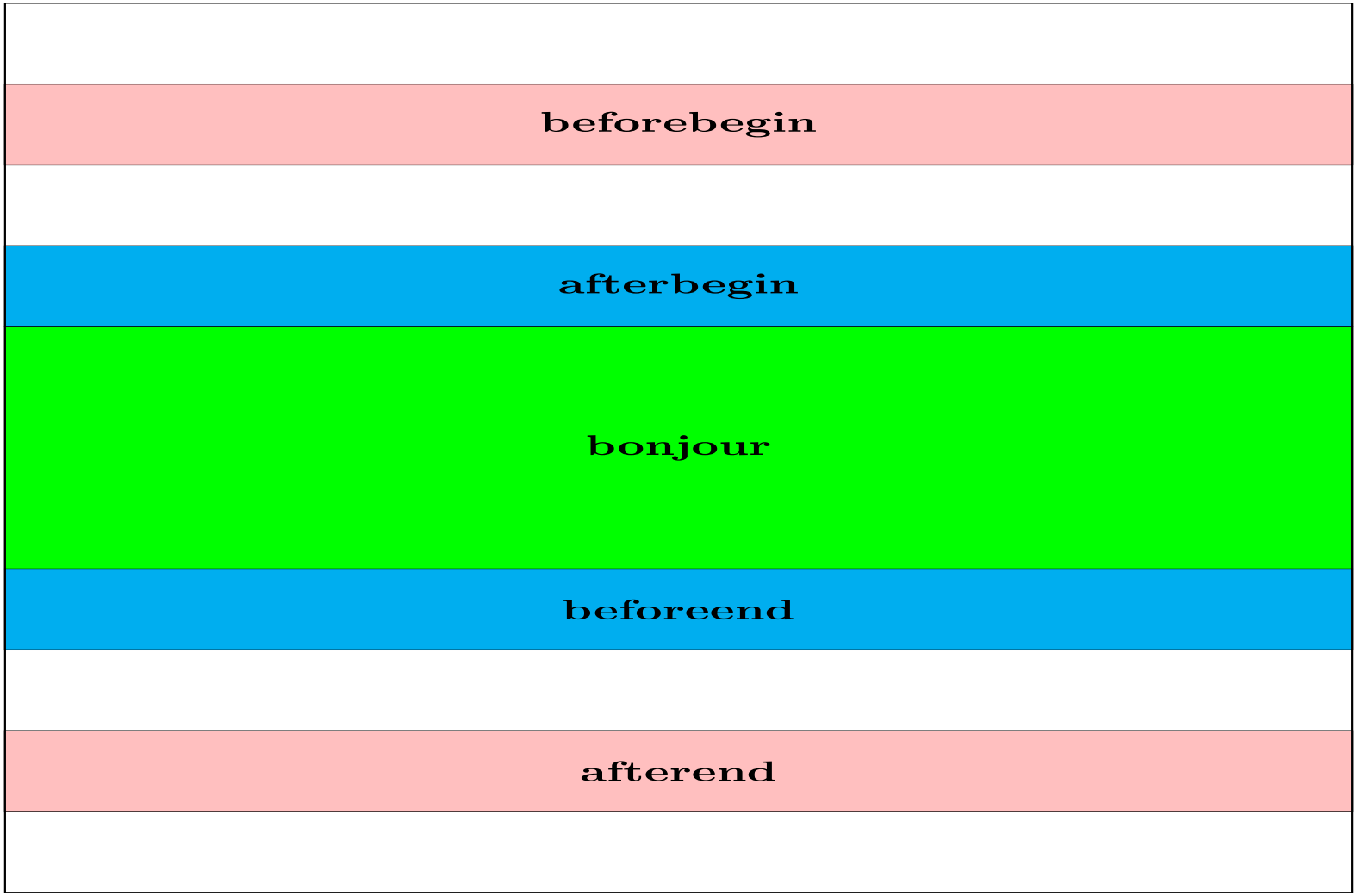
```
.skyblue {
  background-color:
     skyblue;
}
.pink {
  background-color: pink;
}
.green {
  background-color: green;
}
```

Le script `script.js`

```
var p = document.getElementById('first');
p.insertAdjacentHTML("afterbegin","<p class=skyblue> afterbegin </p>");
p.insertAdjacentHTML("afterend","<p class=pink> afterend </p>");
p.insertAdjacentHTML("beforebegin","<p class=pink> beforebegin </p>");
p.insertAdjacentHTML("beforeend","<p class=skyblue> beforeend </p>");
```

JavaScript

Le résultat



JavaScript

Quelques opérations

- `replaceChild` : pour remplacer un nœud
- `cloneNode()` : pour cloner un nœud
- `removeChild()` : pour supprimer un nœud fils
- `hasChildNodes()` : pour vérifier l'existence d'au moins un nœud enfant

JavaScript

Pour récupérer un élément selon son `id`, une solution possible consite à utiliser la méthode `getElementById()`

```
var container = document.getElementById("container");
console.log(container.innerHTML);
```

JavaScript

Pour récupérer un élément selon son `id`, une solution possible consite à utiliser la méthode `getElementById()`

```
var container = document.getElementById("container");
console.log(container.innerHTML);
```

Tous les éléments HTML ayant un identifiant seront chargés dans l'espace global, donc on peut directement faire (sans avoir besoin d'utiliser `getElementById()`)

```
console.log(window.container.innerHTML);;
```

JavaScript

Pour récupérer un élément selon son `id`, une solution possible consite à utiliser la méthode `getElementById()`

```
var container = document.getElementById("container");
console.log(container.innerHTML);
```

Tous les éléments HTML ayant un identifiant seront chargés dans l'espace global, donc on peut directement faire (sans avoir besoin d'utiliser `getElementById()`)

```
console.log(window.container.innerHTML);;
```

Ou en plus simple

```
console.log(container.innerHTML);;
```

JavaScript

Remarque

Pour éviter les conflits avec des éventuelles variables ou fonctions portant le même nom, il est conseillé d'utiliser `getElementById()`.

JavaScript

Quelques évènements

- `click` et `dblclick` : clic et double clic
- `mouseover`, `mouseout`, `mousedown`, `mouseup` et `mousemove` : mouvement de la souris
- `keyup`, `keydown` et `keypress` : touches de clavier
- `focus` et `blur` : ciblage et annulation de ciblage
- `change`, `input` et `select` : pour les éléments d'un formulaire
- `reset` et `submit` : pour la ré-initialisation et la soumission de formulaire
- `load` : chargement de la fenêtre
- `scroll` : évènement lié au mouvement du **scroll**
- ...

JavaScript

Deux façons différentes pour installer un écouteur d'évènement

- comme attribut d'une balise **HTML** en précédant le nom d'évènement par le préfixe `on` ayant comme valeur le nom d'une fonction **JavaScript**
- dans le fichier **JavaScript** avec la méthode `addEventListener()`

JavaScript

Première méthode (la page HTML)

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <link rel="stylesheet" href="style.css">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
      =1.0">
    <title>Event Test</title>
</head>

<body>
    <div id=container>
        <input type=text id=nom>
        <button onclick="direBonjour()" id=cliquer> cliquer </button>
    </div>
    <script src="script.js"></script>
</body>

</html>
```

JavaScript

Contenu du `script.js`

```
function direBonjour(){
    var nom = document.getElementById('nom').value;
    alert ("Bonjour " + nom);
}
```

JavaScript

Première méthode (la page HTML)

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <link rel="stylesheet" href="style.css">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
      =1.0">
    <title>Event Test</title>
</head>

<body>
    <div id=container>
        <input type=text id=nom>
        <button id=cliquer> cliquer </button>
    </div>
    <script src="script.js"></script>
</body>

</html>
```

JavaScript

Contenu du `script.js`

```
var cliquer = document.getElementById('cliquer');

cliquer.addEventListener('click', function() {
    var nom = document.getElementById('nom').value;
    alert ("Bonjour " + nom);
});
```

JavaScript

L'objet `event` permet de récupérer des données sur l'évènement (rien à modifier dans le HTML)

```
var cliquer = document.getElementById('cliquer');

cliquer.addEventListener('click', function (event) {
    console.log(event);
    var nom = document.getElementById('nom').value;
    alert("Bonjour " + nom);
});
```

JavaScript

Pour supprimer un évènement associé à un élément HTML

```
element.removeEventListener(event, nomFunction);
```

JavaScript

Considérons la page `index.html` suivante

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <link rel="stylesheet" href="style.css">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
      =1.0">
    <title>Prevent event</title>
</head>

<body>
    <div>
        Saisir seulement des caractères alphabétiques en minuscule :
        <input type="text" id="nom">
    </div>
    <script src="script.js"></script>
</body>

</html>
```

JavaScript

Pour annuler une saisie qui n'est pas en minuscule

```
let nom = document.getElementById('nom');

nom.addEventListener('keypress', function (event) {
    let pressedKey = event.key;
    if (pressedKey < 'a' || pressedKey > 'z') {
        event.preventDefault();
        alert("Merci de tout écrire en minuscule");
    }
});
```

JavaScript

On ne peut annuler que certains évènements

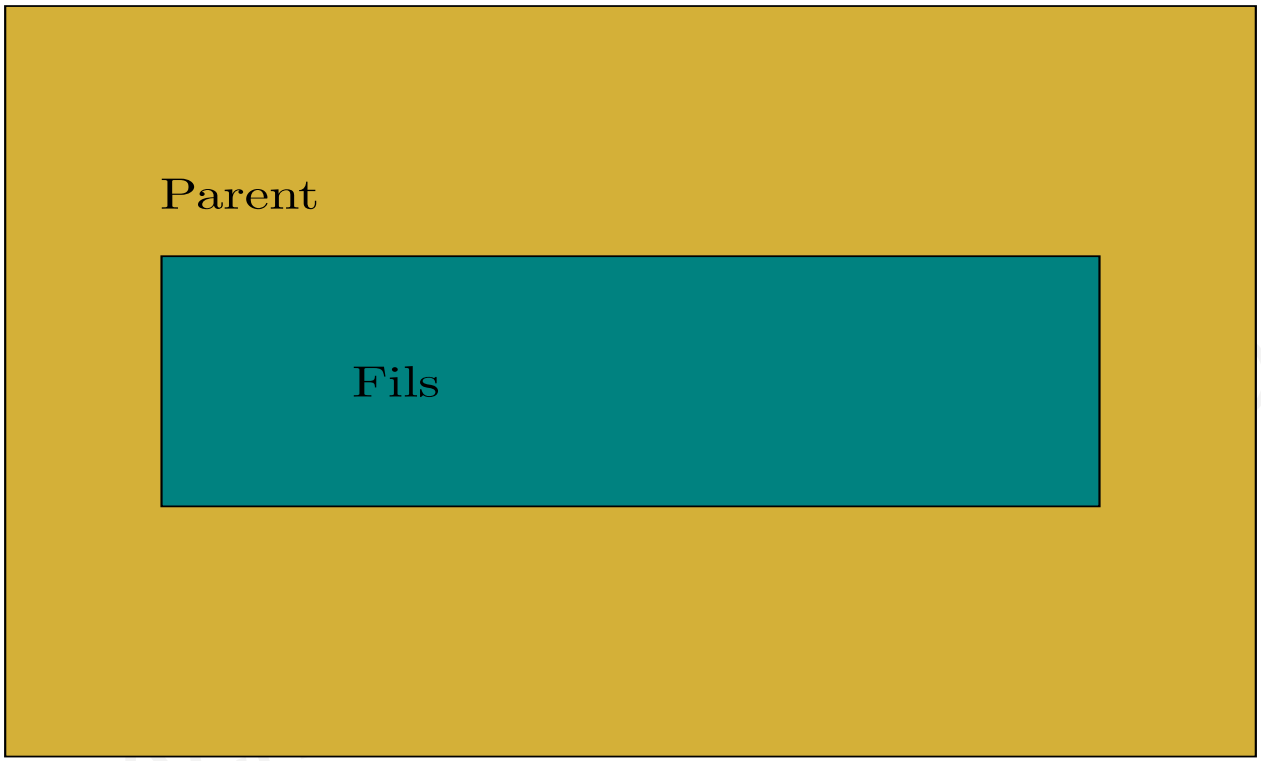
- saisir un texte
- cliquer sur une case à cocher
- soumettre un formulaire
- cliquer sur un lien
- glisser un élément (drag and drop)

JavaScript

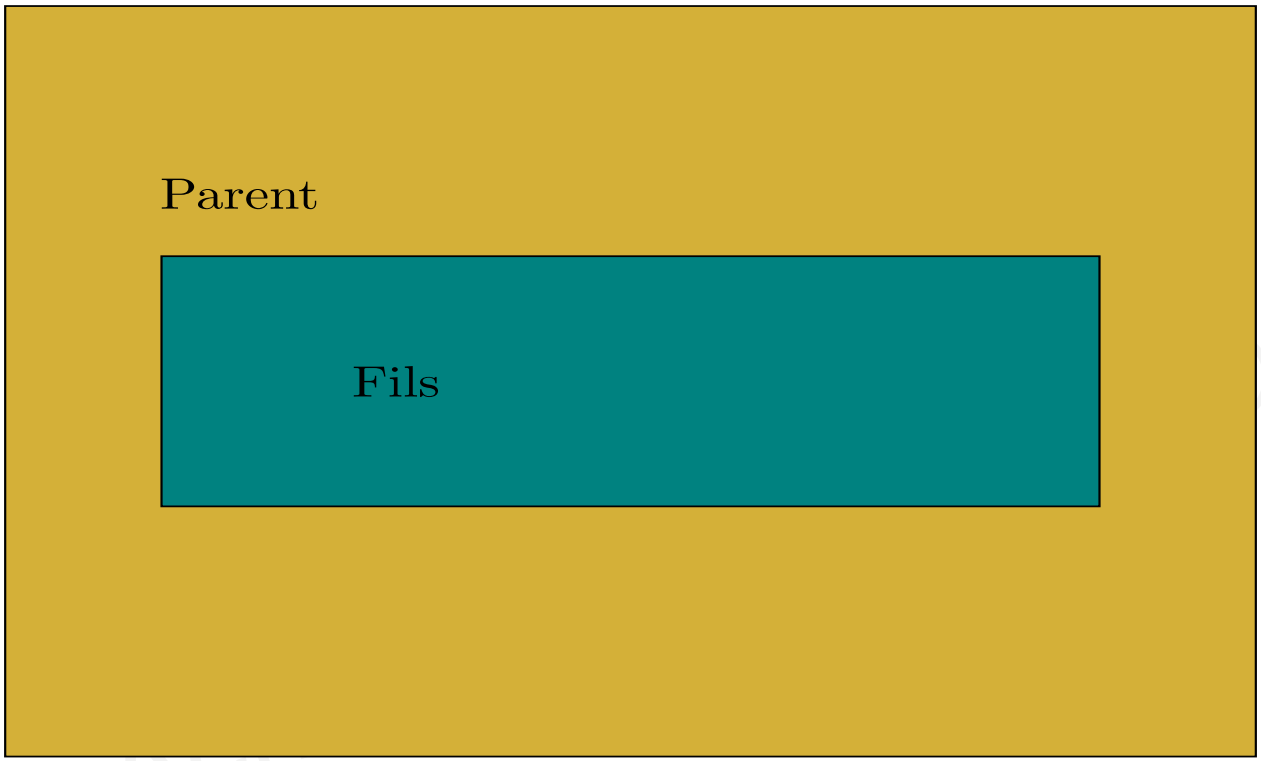
Considérons la page `index.html` suivante

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Stop propagation</title>
    <style>
        div {
            padding: 50px;
            cursor: pointer;
        }
    </style>
</head>

<body>
    <div style="background-color:gold" onclick="jaune()">
        Parent
        <div style="background-color:teal" onclick="vert()">
            Fils
        </div>
    </div>
    <script src="script.js"></script>
</body>

</html>
```

Les deux div parent et fils



Les deux div parent et fils



Code JavaScript de `vert()` et `jaune()`

```
function vert(){
  alert("vert");
}
function jaune(){
  alert("jaune");
}
```

JavaScript

Constats

- En cliquant sur le jaune, jaune est affiché
- En cliquant sur le vert, jaune et vert sont affichés

JavaScript

Constats

- En cliquant sur le jaune, jaune est affiché
- En cliquant sur le vert, jaune et vert sont affichés

Solution

Utiliser la méthode `stopPropagation`

JavaScript

Nouveau code de la fonction `vert()` (pas de changement pour `jaune()`)

```
function vert(event){
  alert("vert");
  event.stopPropagation();
}

function jaune(){
  alert("jaune");
}
```

JavaScript

Nouveau code de la fonction `vert()` (pas de changement pour `jaune()`)

```
function vert(event){
  alert("vert");
  event.stopPropagation();
}

function jaune(){
  alert("jaune");
}
```

Une modification dans le HTML

```
<div style="background-color:gold" onclick="jaune()" >
  Parent
  <div style="background-color:teal" onclick="vert(event)">
    Fils
  </div>
</div>
```