

Introduction à JavaScript



Plan du cours

V. Le Document Object Model

1. Accès simple aux balises html
2. Structure en arborescence du DOM
3. Modification du contenu d'une balise
4. Action sur les attributs d'une balise
5. Modification de propriétés css
6. Insertion d'une balise
7. Suppression d'une balise

VI. Événements en JavaScript

1. Gestion simple
2. Les principaux événements
3. Les écouteurs d'événements
4. L'interface event

Le Document Object Model

Nous utiliserons beaucoup JavaScript au travers de l'interface DOM (Document Object Model) qui permet d'agir sur le contenu html de la page web :

- parcours de l'arborescence html, ...
- modification du contenu de balises html,
- action sur les attributs de balises html,
- modification de propriétés css, de classes css
- insertion, suppression de balises html,

Parmi les interfaces les plus communes, on trouve :

- Document, pour le document courant
- Event, pour tous les événements sur la page

Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.getElementById("maBalise")` qui récupère la balise dont l'identifiant est "maBalise".

```
1  <!DOCTYPE html>
2  <html>
3  <head> ...
9  </head>
10 <body>
11     <input type="button" id="bouton" value="OK">
12     <script type="text/javascript">
13         let b = document.getElementById("bouton");
14         b.addEventListener("click",function() {
15             alert("coucou !");
16         })
17     </script>
18 </body>
19 </html>
```

Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.getElementsByTagName(...)` récupère un tableau de balises du type passé en argument.

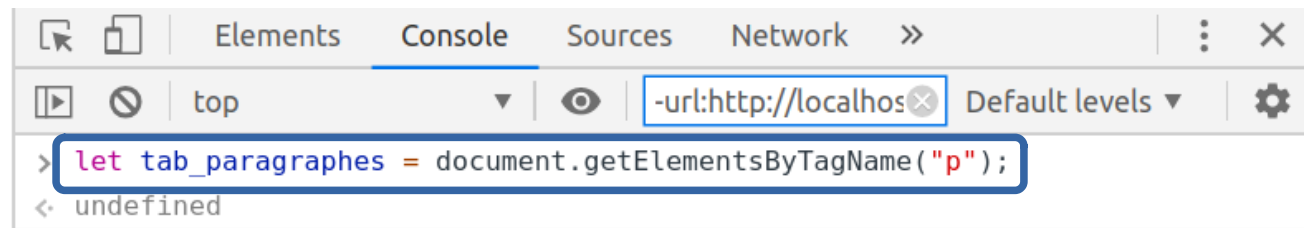
paragraphe 1

paragraphe 2

paragraphe 3

paragraphe 4

paragraphe 5



Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.getElementsByTagName(...)` récupère un tableau de balises du type passé en argument.

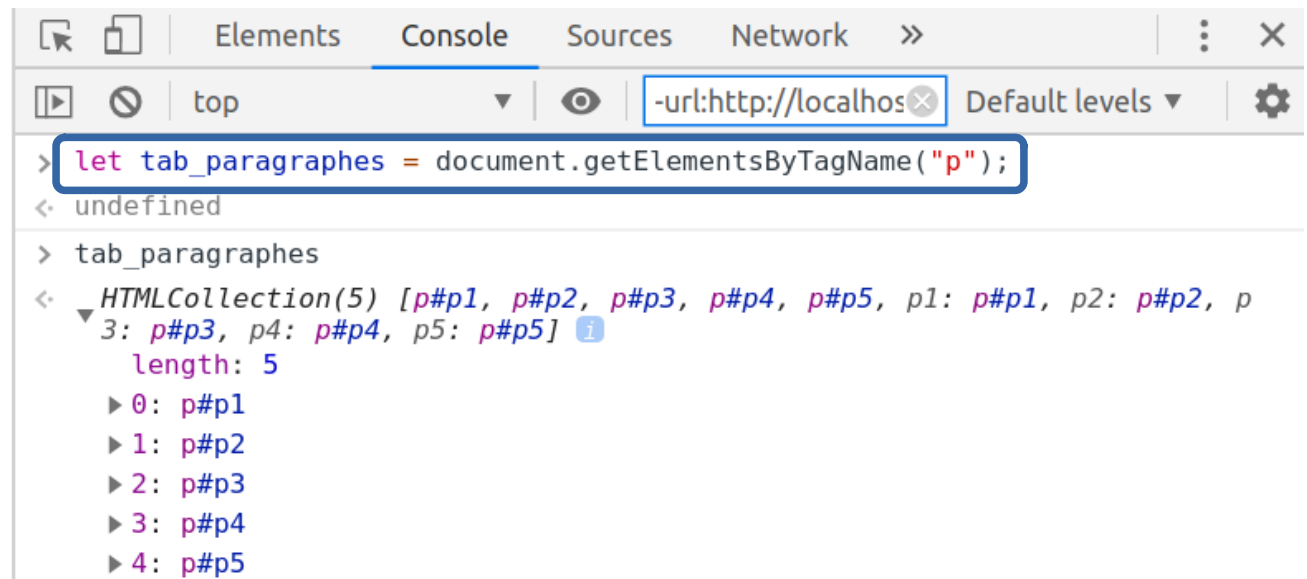
paragraphe 1

paragraphe 2

paragraphe 3

paragraphe 4

paragraphe 5



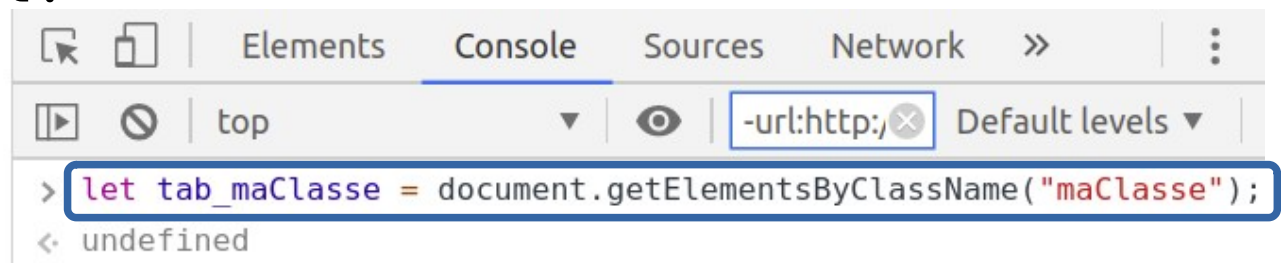
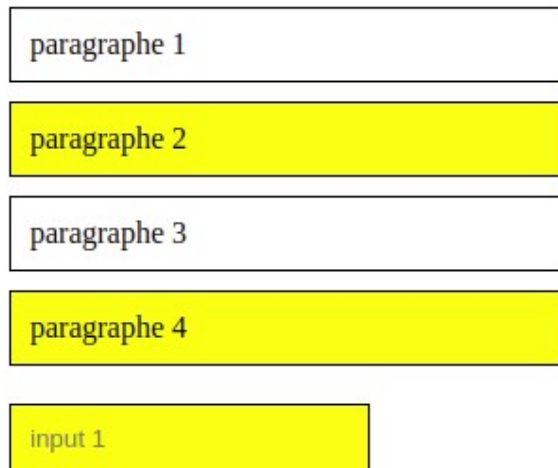
```
let tab_paragraphes = document.getElementsByTagName("p");
< undefined
> tab_paragraphes
< HTMLCollection(5) [p#p1, p#p2, p#p3, p#p4, p#p5, p1: p#p1, p2: p#p2, p3: p#p3, p4: p#p4, p5: p#p5]
  length: 5
    0: p#p1
    1: p#p2
    2: p#p3
    3: p#p4
    4: p#p5
```

Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.getElementsByClassName("..")` récupère un tableau de balises qui ont la classe css passée en argument.

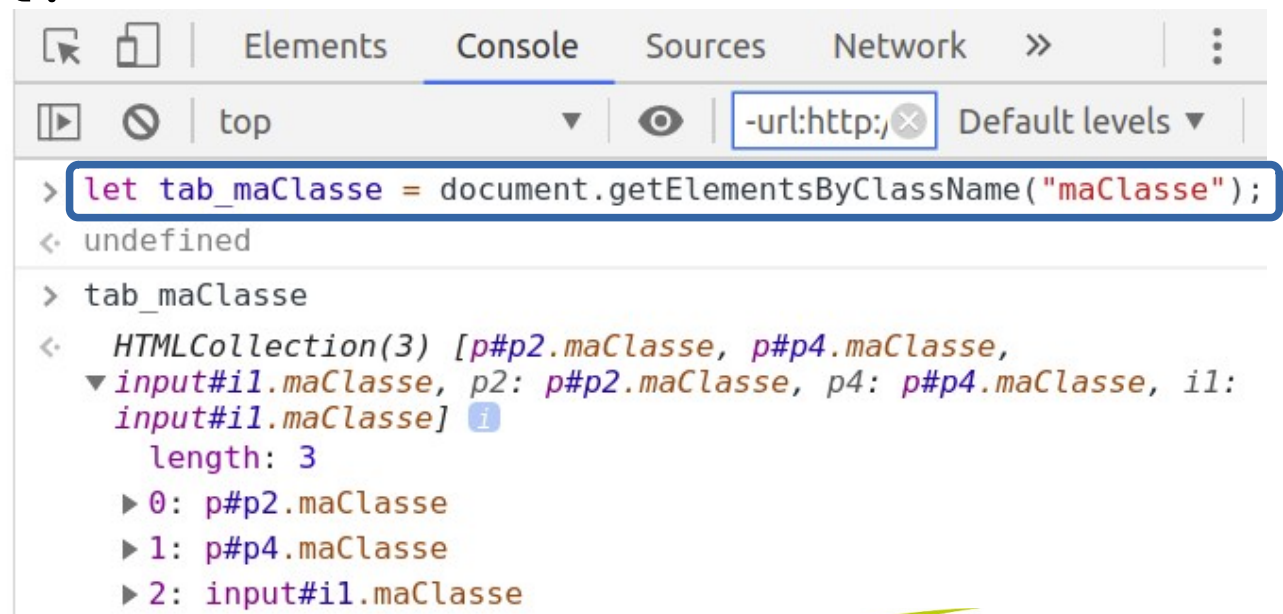
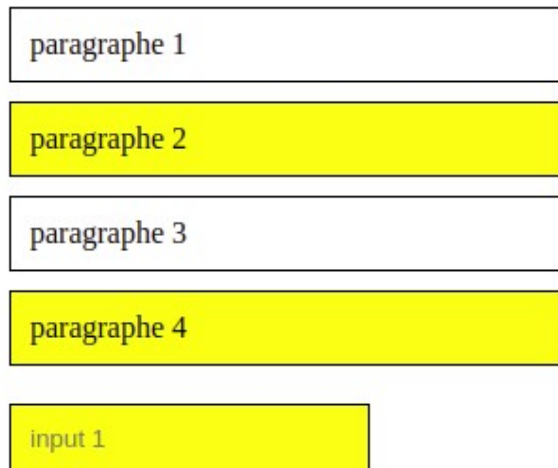


Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.getElementsByClassName(...)` récupère un tableau de balises qui ont la classe css passée en argument.

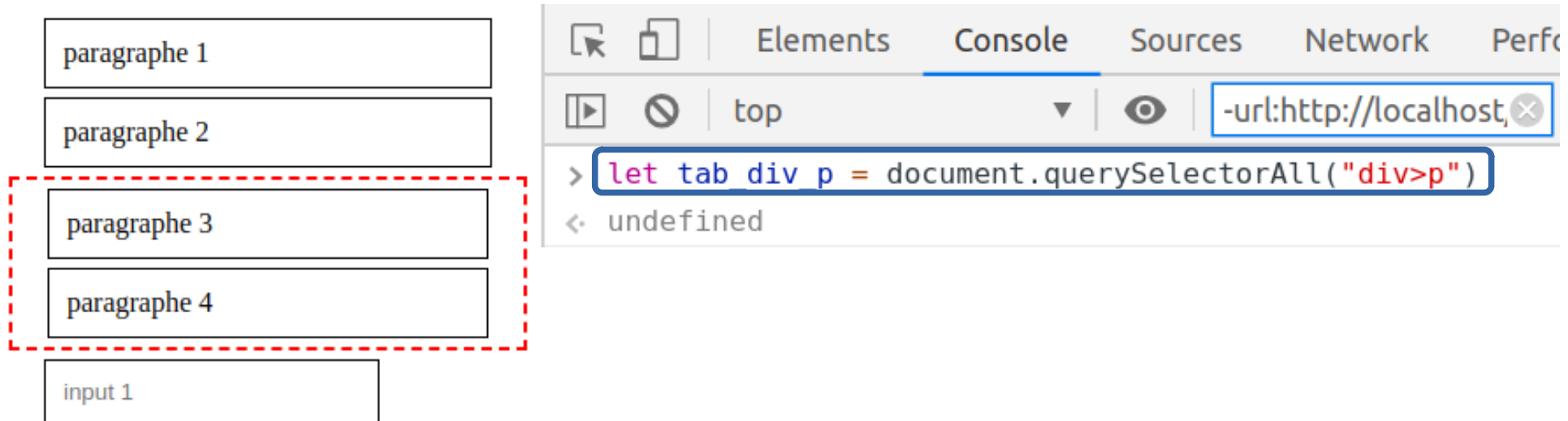


Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.querySelectorAll("..")` renvoie un tableau de balises correspondant au sélecteur css passé en argument.



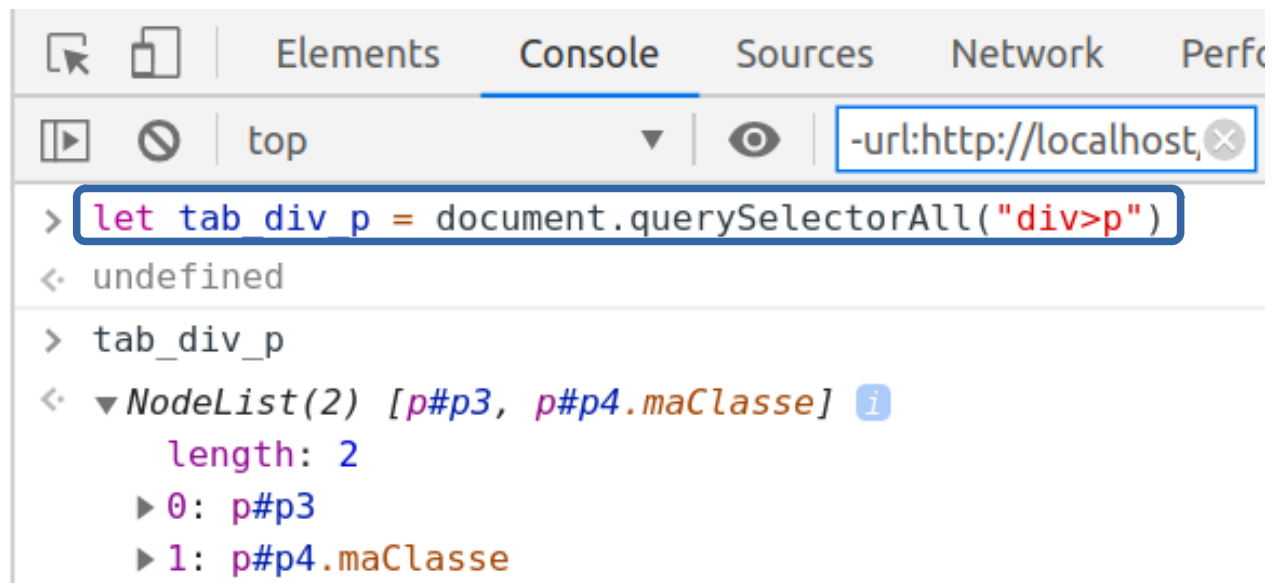
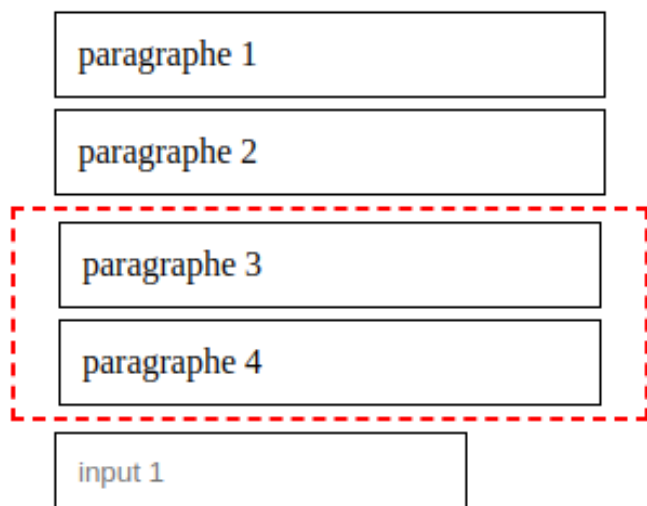
The diagram illustrates the DOM structure and a JavaScript query. On the left, a vertical stack of HTML elements is shown: 'paragraphe 1', 'paragraphe 2', 'paragraphe 3', 'paragraphe 4', and 'input 1'. A red dashed rectangle encloses 'paragraphe 3' and 'paragraphe 4'. On the right, a browser's developer console is shown with the 'Console' tab active. The address bar displays '-url:http://localhost'. The console contains the command `let tab div p = document.querySelectorAll("div>p")` and the result `undefined`.

Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.querySelectorAll(...)` renvoie un tableau de balises correspondant au sélecteur css passé en argument.

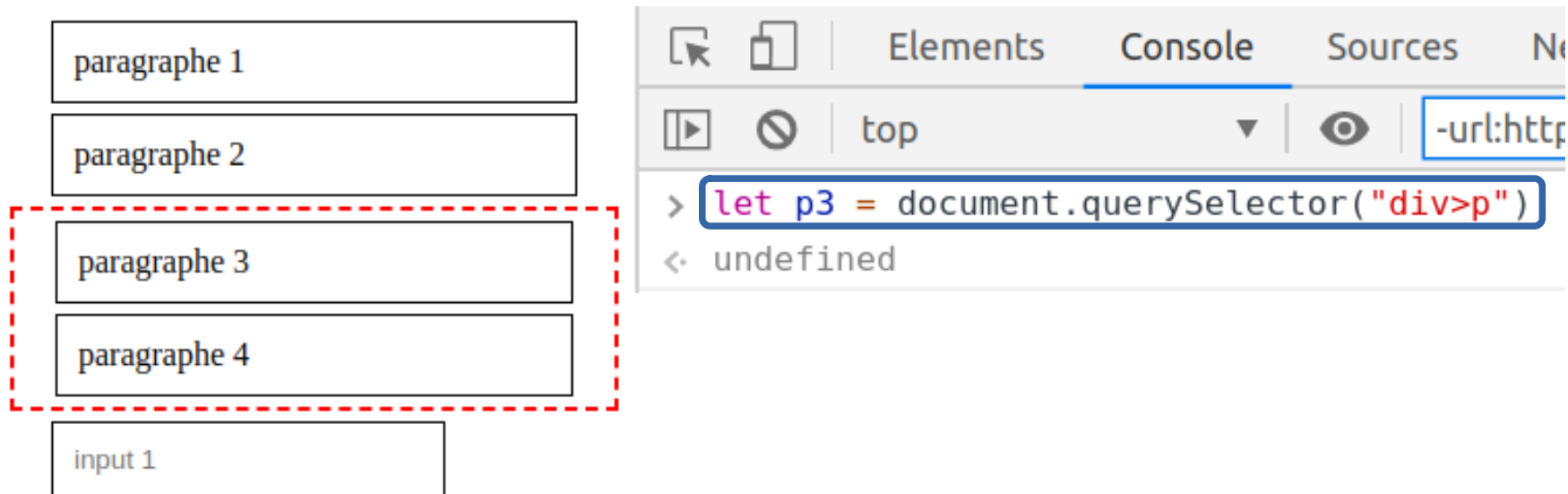


Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.querySelector("...")` renvoie le premier élément du tableau de balises correspondant au sélecteur css passé en argument.



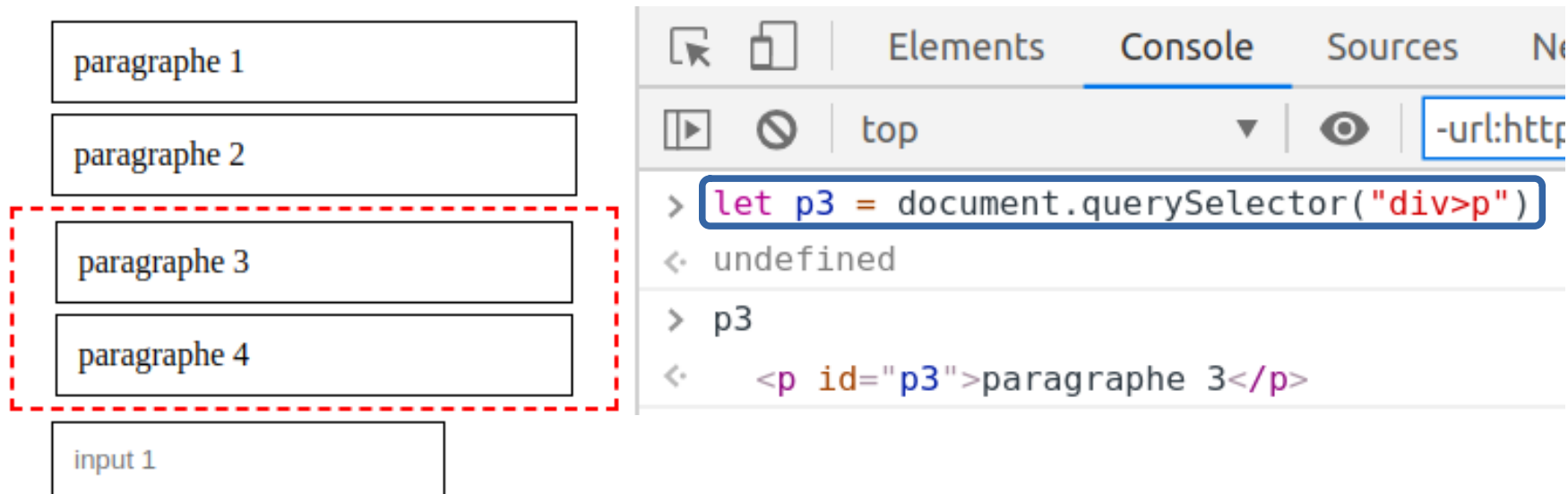
The diagram illustrates the DOM structure on the left and a browser console snippet on the right. On the left, a vertical stack of five boxes represents HTML elements: 'paragraphe 1', 'paragraphe 2', 'paragraphe 3', 'paragraphe 4', and 'input 1'. A red dashed rectangle encloses 'paragraphe 3' and 'paragraphe 4', indicating they are the first elements selected by the CSS selector 'div>p'. On the right, a browser console snippet shows the command `let p3 = document.querySelector("div>p")` being executed, with the result `undefined` displayed below it.

Le Document Object Model

1. Accès simple aux balises html

L'interface `Document` fournit au document courant des méthodes pour accéder aux différentes balises. Les méthodes les plus utilisées sont :

- `document.querySelector("...")` renvoie le premier élément du tableau de balises correspondant au sélecteur css passé en argument.



The diagram illustrates the DOM structure on the left and a browser console snippet on the right.

DOM Structure:

- paragraphe 1
- paragraphe 2
- paragraphe 3 (highlighted with a red dashed box)
- paragraphe 4 (highlighted with a red dashed box)
- input 1

Browser Console Snippet:

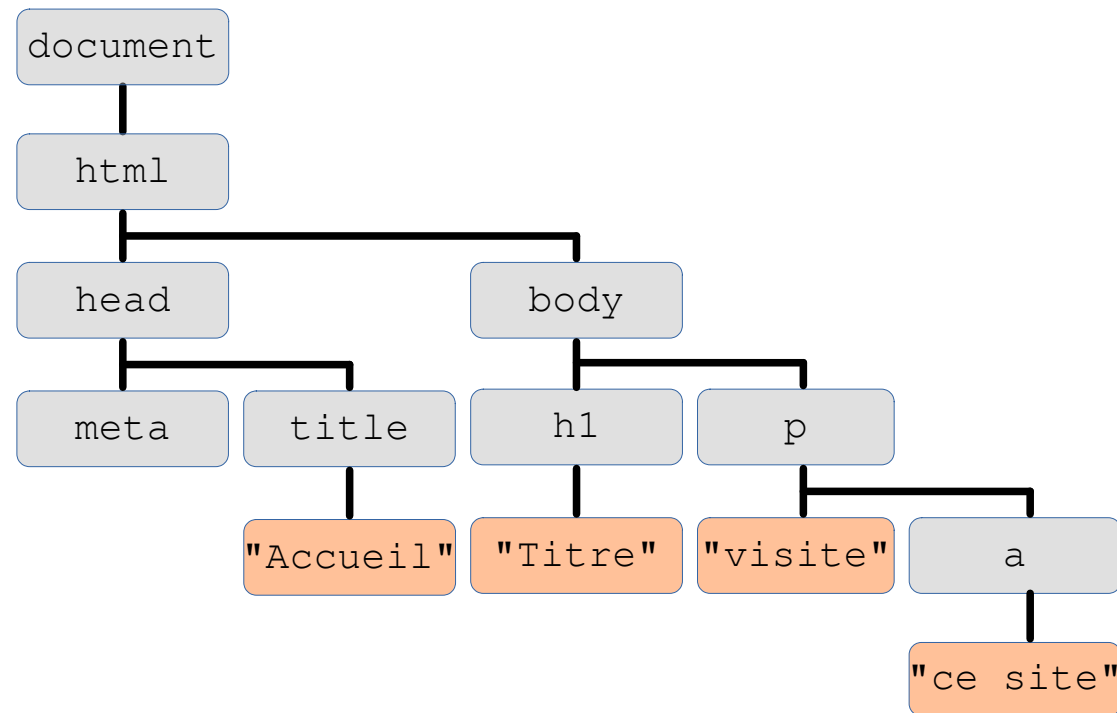
```
> let p3 = document.querySelector("div>p")
< undefined
> p3
< <p id="p3">paragraphe 3</p>
```

Le Document Object Model

2. Structure en arborescence du DOM

Le DOM est élaboré selon la hiérarchie des balises et contenus de la page. Cette hiérarchie correspond à ce que donne une indentation habituelle du code :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Accueil</title>
</head>
<body>
  <h1>Titre</h1>
  <p>
    visite
    <a href="http://www.314.fr">
      ce site
    </a>
  </p>
</body>
</html>
```



Le Document Object Model

2. Structure en arborescence du DOM

Cette structure permet d'accéder aux étages hiérarchiques du document. L'autocomplétion de la console est très utile. On pourra par ex utiliser :

- `document.body` (on récupère le body)
- `document.body.children` (tableau des enfants)
- `document.body.children[0]` (premier enfant)
- `document.body.firstElementChild` (premier enfant)
- `...nextElementSibling` (frère suivant)
- `...previousElementSibling` (frère précédent)
- etc

Le Document Object Model

2. Structure en arborescence du DOM

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Accueil</title>
</head>
<body>
  <h1>Titre</h1>
  <p>
    visite
    <a href="http://www.314.fr">
      ce site
    </a>
  </p>
</body>
</html>
```

```
> document.body
< ▶ <body>...</body>

> document.body.children
< ▶ HTMLCollection(2) [h1, p]

> document.body.children[1]
< ▶ <p>...</p>

> let p = document.body.lastElementChild
< undefined

> p
< ▶ <p>...</p>

> p.firstElementChild
< <a href="http://www.314.fr">
    ce site
    </a>

> let h1 = document.body.firstElementChild
< undefined

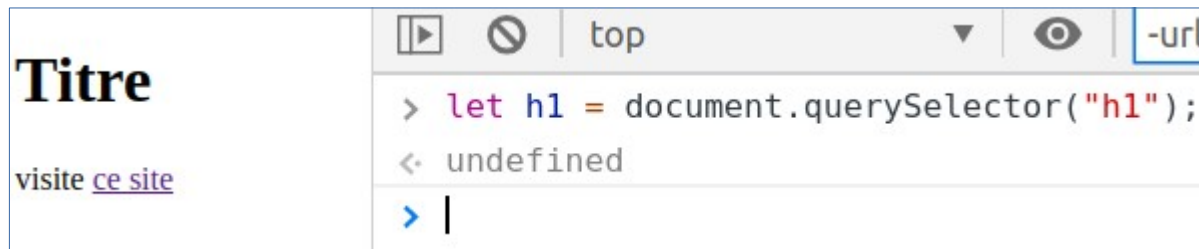
> h1.nextElementSibling
< ▶ <p>...</p>
```

Le Document Object Model

3. Modification du contenu d'une balise

On peut facilement modifier le contenu d'une balise en changeant la valeur de son `innerHTML`.

Cela s'applique aux balises classiques : `div`, `p`, `h1`, `a`, etc.

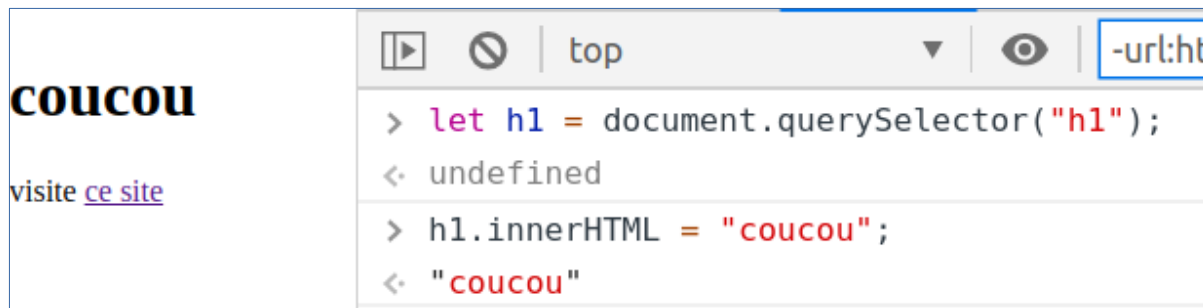
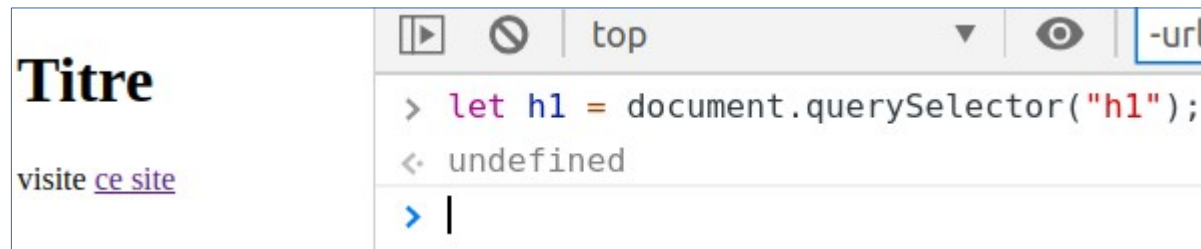


Le Document Object Model

3. Modification du contenu d'une balise

On peut facilement modifier le contenu d'une balise en changeant la valeur de son `innerHTML`.


Cela s'applique aux balises classiques : `div`, `p`, `h1`, `a`, etc.



Le Document Object Model

4. Action sur les attributs d'une balise

On peut changer la valeur de l'attribut d'une balise html. On peut aussi créer un attribut et lui donner une valeur, ou le supprimer.

	<pre>> let inp = document.querySelector("input") < undefined > inp < <input type="number" min="10" value="15"> > inp.max = "25" < "25" > inp < <input type="number" min="10" value="15" max="25"> > inp.setAttribute("name", "monInput") < undefined > inp < <input type="number" min="10" value="15" max="25" name="monInput"> > inp.removeAttribute("min") < undefined > inp < <input type="number" value="15" max="25" name="monInput"></pre>
---	--

Le Document Object Model

5. Modification de propriétés css

a) modification du style d'une balise

Comme au TD1, toute balise html peut voir son style modifié (style «inline») par l'action d'un script.

Titre

visite [ce site](#)

```
> let h1 = document.getElementsByTagName("h1")[0]
```

```
< undefined
```

```
> h1
```

```
< <h1>Titre</h1>
```

```
> h1.style.color = "red"
```

```
< "red"
```

```
> h1.style.fontSize = "3em"
```

```
< "3em"
```

```
> h1.style.fontStyle = "italic"
```

```
< "italic"
```

```
> h1
```

```
< <h1 style="color: red; font-size: 3em; font-style: italic;">Titre</h1>
```

Le Document Object Model

5. Modification de propriétés css

b) changement de la feuille de style - exercice

Exercice simple où l'on reprend divers thèmes abordés dans le cours ou en TD1, en anticipant un peu sur la suite du cours, à savoir :

- Agir sur un attribut d'une balise. Ici, on modifiera la valeur de l'attribut `href` de la balise `link`.
- Gérer un événement par définition d'un attribut (ici `onchange`) de la balise `select`.
- Coder le script dans une fonction JavaScript nommée ici `change_css`.

Le Document Object Model

5. Modification de propriétés css

b) changement de la feuille de style - exercice

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>changer le css</title>
6      <link rel="stylesheet" type="text/css" href="style1.css">
7  </head>
8  <body>
9      <select id="select_css" onchange="change_css();">
10         <option value="style1.css">style 1</option>
11         <option value="style2.css">style 2</option>
12         <option value="style3.css">style 3</option>
13     </select>
14     <div>
15         <p>
16             Lorem ipsum ...
17         </p>
18     </div>
19     <script type="text/javascript">
20         function change_css() {
21             // ...
22         }
23     </script>
24 </body>
25 </html>
```

Le Document Object Model

5. Modification de propriétés css

b) changement de la feuille de style – solutions possibles

```
function change_css() {  
    // récupération de la balise link  
    let balise_link = document.querySelector("link");  
    // récupération de la balise select  
    let balise_select = document.getElementById("select_css");  
    // récupération de la valeur du select  
    let style_choisi = balise_select.value;  
    // affectation de cette valeur au href de balise_link  
    balise_link.href = style_choisi;  
}
```

```
function change_css() {  
    document.querySelector("link").href = document.getElementById("select_css").value;  
}
```

Le Document Object Model

5. Modification de propriétés css

c) modification de classes css

On peut accéder à la liste des classes css d'une balise html (voir TD1) :

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>classes css</title>
6 </head>
7 <body>
8   <div class="c1 c2 c3"></div>
9 </body>
10 </html>
```

```
> let balise_div = document.querySelector("div")
< undefined
> balise_div
< <div class="c1 c2 c3"></div>
> balise_div.classList
< ▼ DOMTokenList(3) ["c1", "c2", "c3", value: "c1 c2 c3"]
  length: 3
  value: "c1 c2 c3"
  0: "c1"
  1: "c2"
  2: "c3"
```


Le Document Object Model

5. Modification de propriétés css

c) modification de classes css

On peut accéder à la liste des classes css d'une balise html (voir TD1) :

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>classes css</title>
6 </head>
7 <body>
8   <div class="c1 c2 c3"></div>
9 </body>
10 </html>
```

```
> let balise_div = document.querySelector("div")
< undefined
> balise_div
< <div class="c1 c2 c3"></div>
> balise_div.classList
< ▶ DOMTokenList(3) ["c1", "c2", "c3", value: "c1 c2 c3"]
> balise_div.classList.remove("c2")
< undefined
> balise_div
< <div class="c1 c3"></div>
> balise_div.classList.add("c4")
< undefined
> balise_div
< <div class="c1 c3 c4"></div>
```


Le Document Object Model

6. Insertion de balises html

a) en mode «brutal»

On peut insérer brutalement des balises html en modifiant de façon textuelle le `innerHTML` de la balise englobante :

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>insertion</title>
6  </head>
7  <body>
8      <div id="div_p"></div>
9  </body>
10 </html>
```

Le Document Object Model

6. Insertion de balises html

a) en mode «brutal»

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>insertion</title>
6 </head>
7 <body>
8     <div id="div_p"></div>
9 </body>
10 </html>
```

```
> let div_p = document.getElementById("div_p")
< undefined

> div_p
< <div id="div_p"></div>

> let tab_p = ["coucou", "hello", "salut"]
< undefined

> for(let mot of tab_p) {
    div_p.innerHTML += "<p>" + mot + "</p>";
}
< "<p>coucou</p><p>hello</p><p>salut</p>"

> div_p
< ▼<div id="div_p">
    <p>coucou</p>
    <p>hello</p>
    <p>salut</p>
</div>
```

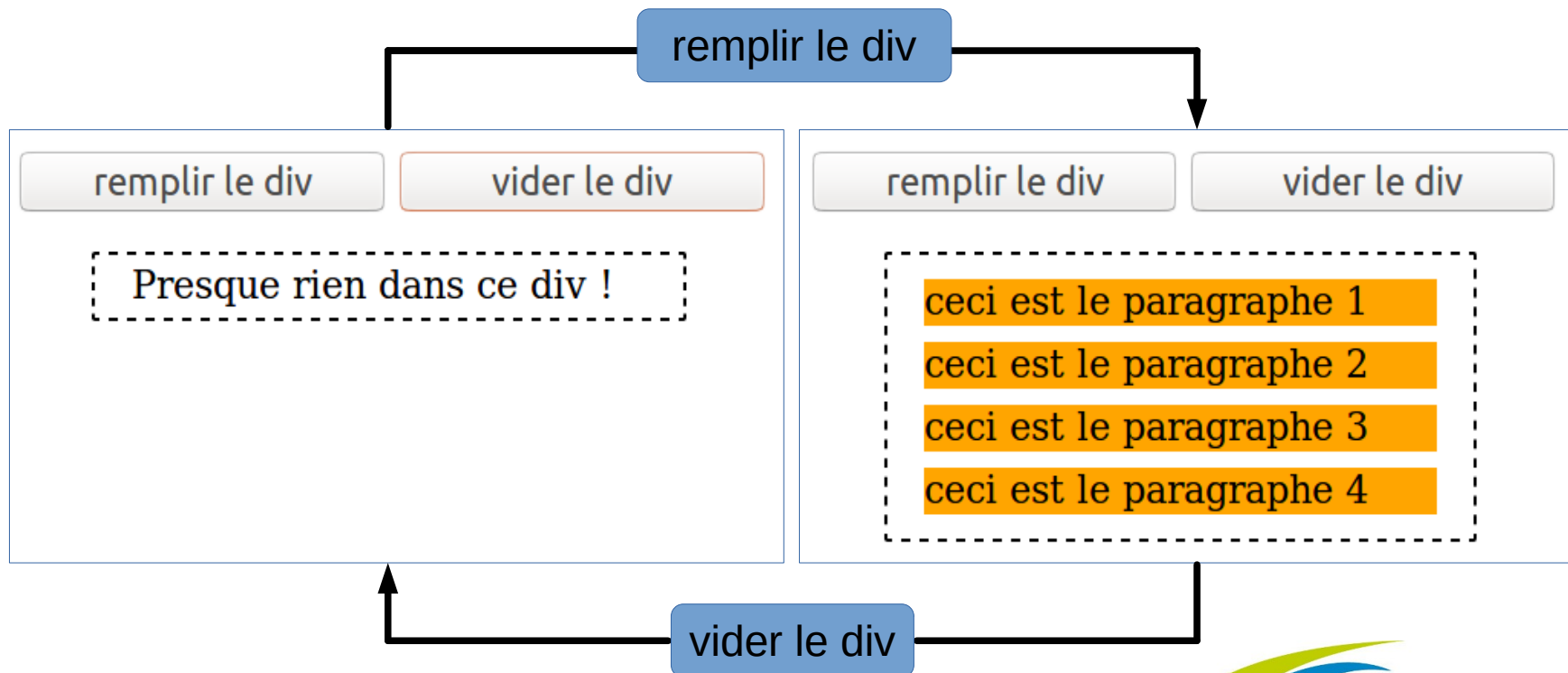
Cette façon de faire est efficace quand la hiérarchie des balises à insérer est simple.

Le Document Object Model

6. Insertion de balises html

b) exercice – agir sur le innerHTML – coder les fonctions

```
<body>  
  <input type="button" id="remplir" value="remplir le div" onclick="remplirMonDiv();">  
  <input type="button" id="vider" value="vider le div" onclick="viderMonDiv();">  
  <div id="monDiv"><p>Presque rien dans ce div !</p></div>  
</body>
```



Le Document Object Model

6. Insertion de balises html

b) exercice – agir sur le innerHTML – coder les fonctions

```
<body>
  <input type="button" onclick="remplirMonDiv();" value="remplir le div">
  <input type="button" onclick="viderMonDiv();" value="vider le div">
  <div id="monDiv">Presque rien dans ce div !</div>
</body>
```

```
function remplirMonDiv() {
  let contenu = "";
  for(i = 1; i <= 4; i++) {
    contenu += "<p id='p_" + i + "'>";
    contenu += "ceci est le paragraphe " + i;
    contenu += "</p>";
  }
  document.querySelector('#monDiv').innerHTML = contenu;
}

function viderMonDiv() {
  document.querySelector('#monDiv').innerHTML = "<p>Presque rien dans ce div !</p>";
}
```

Le Document Object Model

6. Insertion de balises html

c) en mode plus évolué

On crée un élément `<p>`, on lui donne un `innerHTML` et des attributs, on crée le `div_p` qui va l'adopter et on procède à l'adoption (en fin de fratrie) :

```
> let new_p = document.createElement("p")
```

```
> new_p.innerHTML = "paragraphe 2"
```

```
> new_p.setAttribute("id", "p2")
```

```
> let div_p = document.getElementById("div_p")
```

```
> div_p.appendChild(new_p)
```

```
> div_p
```

```
< ▼<div id="div_p">  
    <p id="p2">paragraphe 2</p>  
</div>
```

Le Document Object Model

6. Insertion de balises html

c) en mode plus évolué

On crée un autre élément `<p>`, `innerHTML`, attributs, et on l'insère dans la fratrie avant un élément :

```
> let other_p = document.createElement("p")
```

```
> other_p.innerHTML = "paragraphe 1"
```

```
> other_p.setAttribute("id", "p1")
```

```
> div_p.insertBefore(other_p, new_p)
```

```
< ▼<div id="div_p">  
    <p id="p1">paragraphe 1</p>  
    <p id="p2">paragraphe 2</p>  
</div>
```

Le Document Object Model

7. Suppression de balises html

a) en mode «brutal»

Si l'objectif est de supprimer l'intégralité des balises qui descendent d'une balise englobante, on peut brutalement affecter une chaîne vide au innerHTML de la balise englobante...

```
> div_p
< ▼<div id="div_p">
    <p id="p1">paragraphe 1</p>
    <p id="p2">paragraphe 2</p>
</div>

> div_p.innerHTML = ""
< ""

> div_p
< <div id="div_p"></div>
```


Le Document Object Model

7. Suppression de balises html

b) en mode plus évolué

```
> div_p
< ▼<div id="div_p">
    <p id="p1">paragraphe 1</p>
    <p id="p2">paragraphe 2</p>
</div>

> div_p.children
< ►HTMLCollection(2) [p#p1, p#p2, p1: p#p1, p2: p#p2]

> div_p.removeChild(div_p.children[1])
< <p id="p2">paragraphe 2</p>

> div_p
< ▼<div id="div_p">
    <p id="p1">paragraphe 1</p>
</div>
```

`removeChild` retourne l'élément supprimé

Événements en JavaScript

1. Gestion simple

Dès le TD1 on commence à gérer des événements dont le contexte est la page web : clics, double-clics, survols à la souris par exemple.

Ces événements ont été gérés de façon très simple, en donnant à la balise concernée un attribut relatif à l'événement traité (`onclick`, `ondblclick`, `onmouseout`, etc).

On donne à cet attribut une valeur textuelle avec une chaîne de caractères contenant un script. Le plus souvent, l'appel à une fonction JavaScript.

Code classique :

```
<button id="activer" onclick="activation();" > OK </button>
```

Le clic sur  lancera la fonction `activation`

Événements en JavaScript

1. Gestion simple

On peut produire le même effet en séparant plus le code html et le code JavaScript (comme on essaie de séparer le html du css) :

côté html

```
<button id="activer"> OK </button>
```

côté JavaScript

```
function activation() {  
    // mon code  
}  
  
let b = document.getElementById("activer");  
b.onclick = activation;
```

Événements en JavaScript

1. Gestion simple

On peut produire le même effet en séparant plus le code html et le code JavaScript (comme on essaie de séparer le html du css) :

côté html

```
<button id="activer"> OK </button>
```

côté JavaScript, avec une fonction anonyme

```
let b = document.getElementById("activer");  
  
b.onclick = function() {  
    // mon code  
}
```

Événements en JavaScript

2. Les principaux événements

« matériel »	événement	descriptif
souris	click	clic sur élément
souris	dblclick	double-clic sur élément
souris	mouseover	survol de l'élément
souris	mouseout	sortie de l'élément
souris	mousemove	déplacement dans l'élément
clavier	keydown	maintenir appuyée une touche
clavier	keyup	relâcher une touche
clavier	keypress	frapper une touche (appui puis relâcher)
élément modifiable	change	changer la valeur de l'élément
contenu sélectionnable	select	sélectionner le contenu
formulaire	submit	envoyer
formulaire	reset	réinitialiser
document	load	document chargé

Événements en JavaScript

3. Les écouteurs d'événements

On peut encore produire le même effet en adjoignant un écouteur d'événements (eventListener) à la variable par la méthode `addEventListener` :

côté html

```
<button id="activer"> OK </button>
```

côté JavaScript, avec une fonction classique

```
function activation() {  
    // mon code  
}  
  
let b = document.getElementById("activer");  
b.addEventListener('click', activation);
```

Événements en JavaScript

3. Les écouteurs d'événements

On peut encore produire le même effet en adjoignant un écouteur d'événements (eventListener) à la variable par la méthode `addEventListener` :

côté html

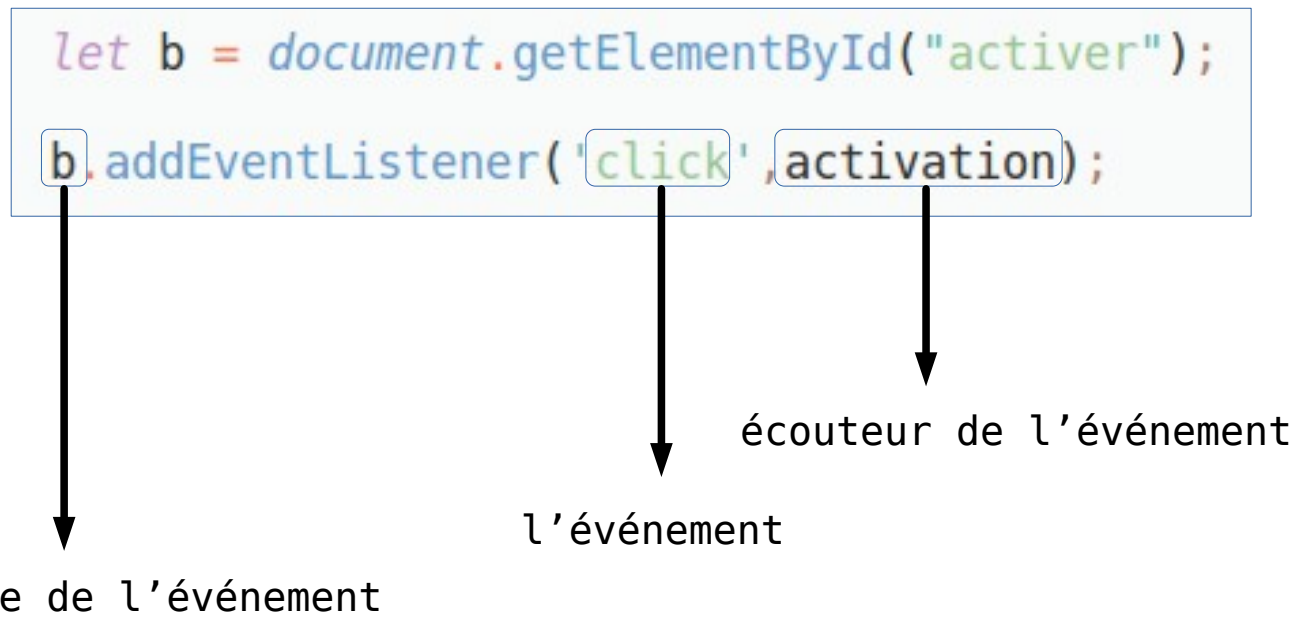
```
<button id="activer"> OK </button>
```

côté JavaScript, avec une fonction anonyme

```
let b = document.getElementById("activer");  
  
b.addEventListener('click',function() {  
    // mon code  
});
```

Événements en JavaScript

3. Les écouteurs d'événements



Événements en JavaScript

3. Les écouteurs d'événements

Les écouteurs d'événements se gèrent simplement :

- On peut définir plusieurs écouteurs d'événement à une même variable :

```
bouton.addEventListener("click",action_1);  
bouton.addEventListener("click",action_2);
```

- On peut à tout moment retirer à une variable un écouteur d'événement précédemment attribué :

```
bouton.removeEventListener("click",activation);
```


Événements en JavaScript

4. L'interface Event

Quand un événement se produit, il y a création d'un objet implémentant l'interface `Event` (ou plutôt une interface qui étend `Event`).

On peut alors récupérer cet objet et en extraire des renseignements très utiles, par analyse de ses nombreux attributs :

La portée de l'objet récupéré est limitée à la fonction traitante !

```
> document.body.addEventListener("click",function(e) {  
    console.log(e);  
});
```

action traitante

l'événement intercepté

Événements en JavaScript

4. L'interface Event

```
> document.body.addEventListener("click",function(e) {  
    console.log(e);  
});  
← undefined
```



```
▼ MouseEvent {isTrusted: true, screenX: 733, screenY: 482, cli  
  isTrusted: true  
  screenX: 733  
  screenY: 482  
  clientX: 338  
  clientY: 358  
  ctrlKey: false  
  shiftKey: false  
  altKey: false  
  metaKey: false  
  button: 0  
  buttons: 0
```

→ coordonnées du clic

Événements en JavaScript

4. L'interface Event

```
> document.body.addEventListener("keydown",function(e) {  
    console.log("une touche a été pressée.");  
    console.log("touche : " + e.key);  
    console.log("code   : " + e.keyCode);  
});
```

```
< undefined
```

```
une touche a été pressée.
```

```
touche : a
```

```
code   : 65
```

```
une touche a été pressée.
```

```
touche : Shift
```

```
code   : 16
```

```
une touche a été pressée.
```

```
touche : Enter
```

```
code   : 13
```

Événements en JavaScript

4. L'interface Event

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>exemple-event</title>
  </head>
  <body>
    <p>appuyez sur une touche</p>
    <p id="codeTouche"></p>
  </body>
  <script type="text/javascript" src="script4.js"></script>
</html>
```

```
function touche(event){
  var touche = event.keyCode;
  var p = document.getElementById("codeTouche");
  p.innerHTML = 'le code de cette touche est ' + touche;
}

var b = document.body;
b.addEventListener('keydown', touche);
```