JavaScript et HTML

Cours 5

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-js

Plan

- solutions des exercices
- fonctions en JavaScript
- portée des variables — mode strict
- méthodes statiques
- bibliothèque mathématique
- HTML et le DOM
- navigation dans le DOM
- animation JavaScript en HTML

JavaScript et fonctions

- plusieurs syntaxes pour définir des fonctions

```
function plus (x, y) {return x + y}

let plus = function (x, y) { return x+y }

let plus = (x,y) => {return x + y}
```

- la valeur des expressions Javascript peut être des fonctions

JavaScript et fonctions

- Une fonction peut retourner une valeur fonctionnelle

```
function add(a){
    return function(b){return a+b}
}

let plusUn = add (1)

let x = 0
let y = plusUn (x)
console.log (x, y)
```

Exercice Écrire la fonction `foldRight`

Exercice Donner un exemple où `foldRight` et `foldRight` donnent des résultats différents

- Une fonction peut avoir un argument fonctionnel

```
function foldLeft (f, a, v) {
    let r = v;
    a.forEach (m => {
        r = f (r, m)
    })
    return r
}

function plus (x, y) { return x + y }

let a = [1, 2, 3, 4, 5]
console.log (foldLeft (plus, a, 0)

let s = foldLeft (function(x, y) {return x*y}, a, 1)
console.log (s)
```

JavaScript et fonctions

- Considérons la méthode hello de la classe suivante:

```
class Personne {
    constructor (nom, age) {
        this.nom = nom
        this.age = age
    }

    hello () {console.log (`hello, c'est ${this.nom}`)}
}

jj = new Personne ('Jean-Jacques', 19)
wei = new Personne ('Wei', 28)
console.log (wei.hello())
```

- ou en utilisant des valeurs fonctionnelles

```
    hello = function () {console.log (`hello, c'est ${this.nom}`)}

    hello = () => {console.log (`hello, c'est ${this.nom}`)}
```

Affectation déstructurante

- Une autre manière d'accéder aux propriétés des objets

```
class Personne {
    constructor (nom, age) {
        this.nom = nom
        this.age = age
    }

    hello () {console.log (`hello, c'est ${this.nom}`)}
}

jj = new Personne ('Jean-Jacques', 19)
wei = new Personne ('Wei', 28)
console.log (wei.hello())
```

- avec une affectation déstructurante

```
let { nom, age } = wei
console.log (nom, age)
```

attention: nom et age ne peuvent plus être définis dans le *scope* courant

Portée des variables

- Ne jamais utiliser le mot-clé `var` et préférer le mot-clé `let`

```
let nom = 'Jean-Jacques'
{
    let nom = 'Wei'

    console.log(nom)
}
console.log(nom)
```

- les accolades délimitent la portée (*scope*) des variables déclarées avec `let`

Classes et méthodes statiques

- Une méthode statique est attachée à une classe (pas à un objet)

```
class Personne {
    constructor (nom, age) {
        this.nom = nom
        this.age = age
    }

    static hello () {console.log ("hello tout le monde!")}
}

Personne.hello()
```

[comme en Java]

- Une méthode statique est attachée à une classe (pas à un objet)

Quelques objets utiles

- `Math` est un objet unique préexistant

```
Math.E        // returns Euler's number
Math.PI       // returns PI
```

- `Math.random` donne un nombre réel aléatoire entre 0 et 1

Exercice Initialiser un tableau de 10 valeurs entières entre 0 et 99.

- `Date` est une classe pour créer des objets représentant des dates

```
console.log (new Date())
```

méthodes de `Math`

```
abs(x)
acos(x)
acosh(x)
asin(x)
asinh(x)
atan(x)
atan2(y, x)
atanh(x)
cbrt(x)
ceil(x)
cos(x)
cosh(x)
exp(x)
floor(x)
log(x)
max(x, y, z, ..., n)
min(x, y, z, ..., n)
pow(x, y)
random()
round(x)
sign(x)
sin(x)
sinh(x)
sqrt(x)
tan(x)
tanh(x)
trunc(x)
```

HTML

- le langage des pages web
- un bon tutoriel HTML : `http://www.w3schools.com/html`
- une page HTML est composée de:
  - un en-tête (<*head*>)
  - un style (<*style*>)  avec des fichiers `css`
  - un corps (<*body*>)

```
<html>
<body>

<p>This is a paragraph
<p>This is a paragraph

</body>
</html>
```

HTML

- la syntaxe est rudimentaire, mais bien parenthésée

```
<html> . . . </html>

<head> . . . </head>

<body> . . . </body>

<p> . . . </p>
```

- pas de différence entre majuscules et minuscules

```
<HTML> . . . </HTML>

<HEAD> . . . </HEAD>
```

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<p>Voici un paragraphe</p>

<p>et un autre paragraphe</p>

</body>
</html>
```

HTML

- têtes de chapîtres ou de sections (*headings*)

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

- commentaires

```
<!-- Voici un commentaire -->
```

- saut de ligne

```
<br>
```

- listes

```
<ul>Heading 1</ul>

<ol>Heading 2</ol>

<li>Heading 3</li>
```

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

HTML

- ancres - liens (*hyperlinks*)

```
<a href="url">texte de description</a>
```

- URL (*Universal Resource Location*)

```
http://jeanjacqueslevy.net/courses/index.html
file:///Users/levy/public_html/pubs/index.html
```

[index.html est optionnel]

- insertion d'une image

```
<img src="jjl.jpg">
```

- insertion d'un code JavaScript

```
<script> . . . </script>
```

- on verra les styles (CSS) plus tard

HTML

- chaque commande peut avoir des attributs (taille, couleur, fichier)

```
<h1 color=red>
```

```
<p id="EAAA">
```

- URL avec identifiant

```
http://jeanjacqueslevy.net/courses/index.html#EAAA
```

- URL (*Universal Resource Location*)

```
http://jeanjacqueslevy.net/courses/index.html
```

```
file:///Users/levy/public_html/pubs/index.html
```

[index.html est optionnel]

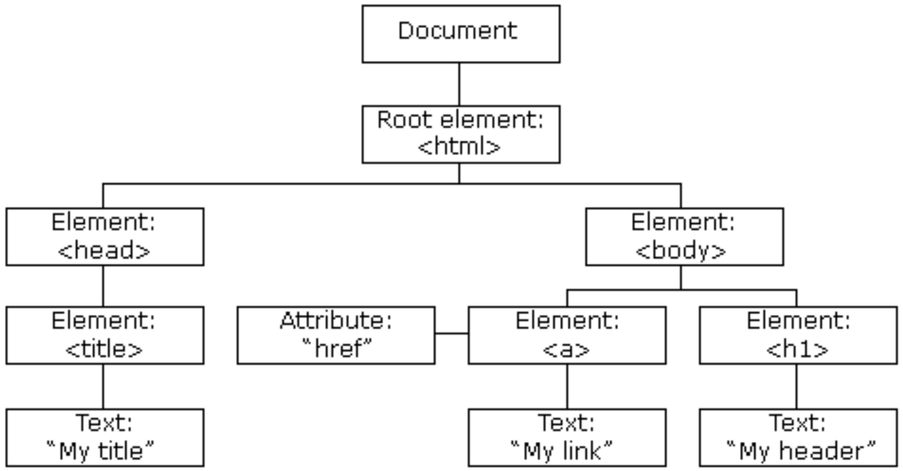
- insertion d'une image

```
<img src="jjl.jpg">
```

- on verra le style (CSS) plus tard

HTML Document Object Model

- le DOM est la représentation arborescente d'un document HTML



- le DOM est représenté en JavaScript par un objet

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

HTML Document Object Model

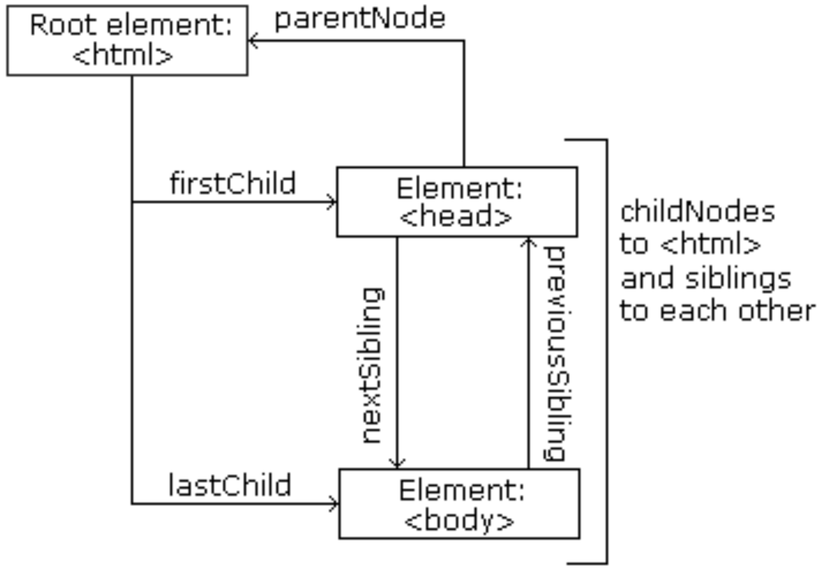
- naviguer dans le DOM

```
<html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML = document.getElementById("id01").innerHTML;
</script>

</body>
</html>
```



- autre manière

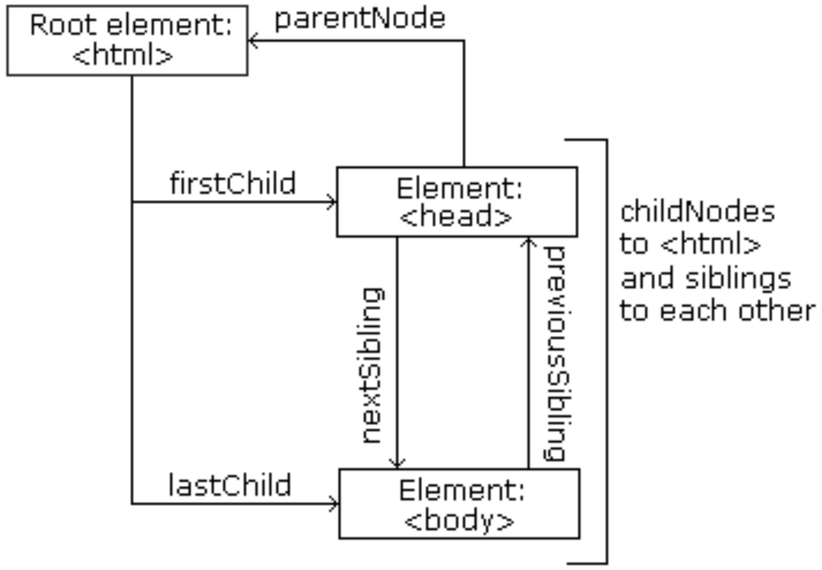
```
<html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML = document.getElementById("id01").firstChild.nodeValue;
</script>

</body>
</html>
```

HTML Document Object Model



- et encore une autre

```
<html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02">Hello!</p>

<script>
document.getElementById("id02").innerHTML = document.getElementById("id01").childNodes[0].nodeValue;
</script>

</body>
</html>
```

- pour être plus exhaustif, voir*:* `http://www.w3schools.com/js/js_htmldom_navigation.asp`

HTML Document Object Model

document.getElementById(*id*)

document.getElementsByTagName(*name*)

document.getElementsByClassName(*name*)

element.innerHTML = *new html content*

element.*attribute* = *new value*

element.style.*property* = *new style*

element.setAttribute*(attribute, value)*

document.createElement(*element*)

document.removeChild(*element*)

document.appendChild(*element*)

document.replaceChild(*new, old*)

document.write(*text*)

HTML avec Animation javascript

```
<!DOCTYPE html>
<html>

<style>
#container {
  width: 400px;
  height: 400px;
  position: relative;
  background: yellow;
}
#animate {
  width: 50px;
  height: 50px;
  position: absolute;
  background-color: red;
}
</style>


<body>

<p><button onclick="myMove()">Click Me</button></p>

<div id ="container">
  <div id ="animate"></div>
</div>
```


```
<script>
function myMove() {
  let id = null;
  const elem = document.getElementById("animate");
  let pos = 0;
  clearInterval(id);
  id = setInterval(frame, 5);
  function frame() {
    if (pos == 350) {
      clearInterval(id);
    } else {
      pos++;
      elem.style.top = pos + "px";
      elem.style.left = pos + "px";
    }
  }
}
</script>

</body>
</html>
```

- pour être plus exhaustif, voir*:* `http://www.w3schools.com/js/js_htmldom_animate.asp`

HTML avec *Input*

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
</script>
</head>
<body>

<h2>JavaScript Validation</h2>

<form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">
  Name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

input

- pour être plus exhaustif, voir: `http://www.w3schools.com/js/js_validation.asp`

Prochain cours

- un bon tutoriel JavaScript: `http://www.programiz.com/javascript`
- un autre tutoriel JavaScript: `http://www.w3schools.com/js`
- notions encore plus avancées de JavaScript (exceptions, prototypes, modules)
- gestion des événements
- les fichiers de style
- essais de pages web sexy !