JavaScript et HTML

Cours 8

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-js

Plan

- exemple (liste sélective — suite)
- formulaires et input
- exceptions, prototypes, modules

Listes sélectives

```
<!DOCTYPE html>
<html>

<head>
  <meta http-equiv="content-type" content="text/html;charset=UTF-8" />
  <link rel="stylesheet" href="slists.css" type="text/css" charset="utf-8"/>
  <title>Test d'affichage sélectif</title>
  <script type="text/javascript" charset="utf-8" src="slists.js"></script>
</head>
```

```
<body>
  <h1>Test d'affichage sélectif d'une liste</h1>

  <h2>Liste avec ouverture/fermeture en CSS</h2>
  <ul>
  <li class="dynCSS">item1
      <ul>
      <li>sous-item 1.1</li>
      <li>sous-item 1.2</li>
      <li>sous-item 1.3</li>
      </ul>
  </li>
  <li class="dynCSS">item2
      <ul>
      <li>sous-item 2.1</li>
      <li>sous-item 2.2</li>
      </ul>
  </li>
  <li class="dynCSS">item 3</li>
  </ul>
```

```
  <h2>Liste avec ouverture/fermeture en javascript</h2>
  <ul>
  <li><span class="toggle">item1</span><span style="display:inline">...</span>
      <ul>
      <li>sous-item 1.1</li>
      <li>sous-item 1.2</li>
      <li>sous-item 1.3</li>
      </ul>
  </li>
  <li><span  class="toggle">item2</span><span style="display:inline">...</span>
      <ul>
      <li><a href="#test2.1">sous-item 2.1</a></li>
      <li><a href="#test2.2">sous-item 2.2</a></li>
      </ul>
  </li>
  <li><span class="toggle">item3</span><span style="display:inline">...</span></li>
  </ul>

</body>
</html>
```

Listes sélectives

```
<body>
  <h1>Test d'affichage sélectif d'une liste</h1>

  <h2>Liste avec ouverture/fermeture en CSS</h2>
  <ul>
  <li class="dynCSS">item1
      <ul>
      <li>sous-item 1.1</li>
      <li>sous-item 1.2</li>
      <li>sous-item 1.3</li>
      </ul>
  </li>
  <li class="dynCSS">item2
      <ul>
      <li>sous-item 2.1</li>
      <li>sous-item 2.2</li>
      </ul>
  </li>
  <li class="dynCSS">item 3</li>
  </ul>
```

affichage des noeuds

```
li.dynCSS:hover {color: red}
li.dynCSS:hover ul{display: block}
li.dynCSS:hover ul li{color: green}

/*pour la partie javascript*/
li {color: blue}
li ul {display: none}
li ul li {color: green}
```

Listes sélectives

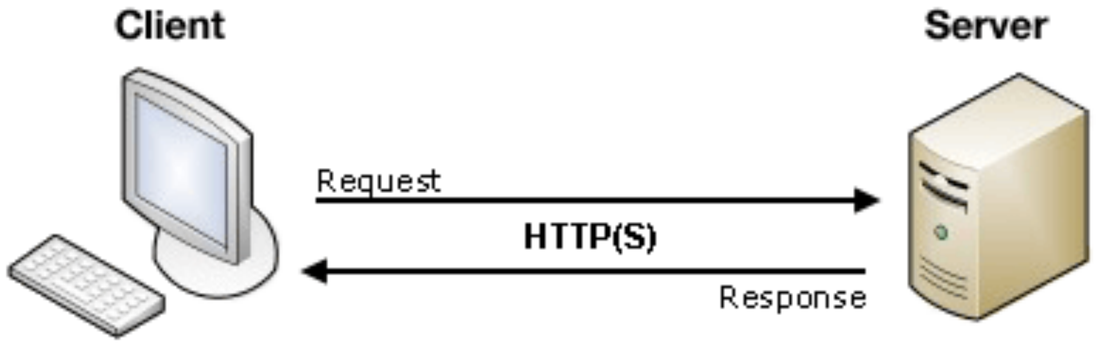
```
function next (name, node) {
    for (let r = node; r != null; r = r.nextSibling)
      if (r.nodeName == name)
          return r;
    return null;
}

function toggle (e) {
    let item = e.target;
    let ulNode = next ("UL", item);
    if (item.nextSibling.style.display == "inline") {
        if (ulNode != null)
            ulNode.style.display = "block";
        item.nextSibling.style.display = "none";
    } else {
        if (ulNode != null)
            ulNode.style.display = "none";
        item.nextSibling.style.display = "inline";
    }
}

window.addEventListener ("load", function() {
    let ts = document.getElementsByClassName ("toggle");
    for (let i = 0; i < ts.length; ++i)
        ts[i].addEventListener ("click", toggle, false);
    },false)
```

```
<h2>Liste avec ouverture/fermeture en javascript</h2>
<ul>
<li><span class="toggle">item1</span><span style="display:inline">...</span>
    <ul>
    <li>sous-item 1.1</li>
    <li>sous-item 1.2</li>
    <li>sous-item 1.3</li>
    </ul>
</li>
<li><span  class="toggle">item2</span><span style="display:inline">...</span>
    <ul>
    <li><a href="#test2.1">sous-item 2.1</a></li>
    <li><a href="#test2.2">sous-item 2.2</a></li>
    </ul>
</li>
<li><span class="toggle">item3</span><span style="display:inline">...</span></li>
</ul>

</body>
</html>
```

Formulaires

- l'interaction avec une page HTML se fait avec les événements (souris, chargement, clavier)
- formulaires <form> en HTML pour rentrer des données textuelles
- ces données sont souvent envoyées à un serveur avec le protocole HTTP
- une manière simple consiste à surcharger l'adresse de l'URL du serveur avec des paramètres. Par exemple:

```
https://www.google.fr/search?q=JJL+75&client=safari&...
```



Formulaires

- le serveur exécute un code qui retourne par exemple du code HTML que l'on pourra afficher

- le code du serveur peut être en PHP, dans un CGI (*Common Gateway Interface*) et retourne un texte HTML
  [le code du CGI est dans un langage de programmation arbitraire (Perl, Python, C++, Ocaml, Haskell …)]

- PHP est un préprocesseur pour HTML (*Hypertext Preprocessor*)

```
<!DOCTYPE html>
<html>
<body>

<?php
$txt = "PHP";
echo "Salut ". $_GET['fname'] . "!<br>";
echo "J'aime $txt!";
?>

</body>
</html>
```

le code PHP qui retourne le paramètre 'fname' de l'URL

Formulaires

- un premier exemple de formulaire

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
</head>

<body>

<h2>HTML Forms</h2>

<form action="http://localhost/progs/hello.php" target="_blank">
  <label for="fname">Prénom:</label><br>
  <input type="text" id="fname" name="fname" value="Jean-Jacques"><br>
  <input type="submit" value="Envoyer">
</form>

<p>Cliquer sur le bouton "Envoyer", et le formulaire est envoyé à la page "hello.php".</p>

</body>
</html>
```

- l'action déclenchée par `submit` envoie une requête http pour l'URL

```
http://localhost/progs/hello.php&fname=Iteki
```

Formulaires

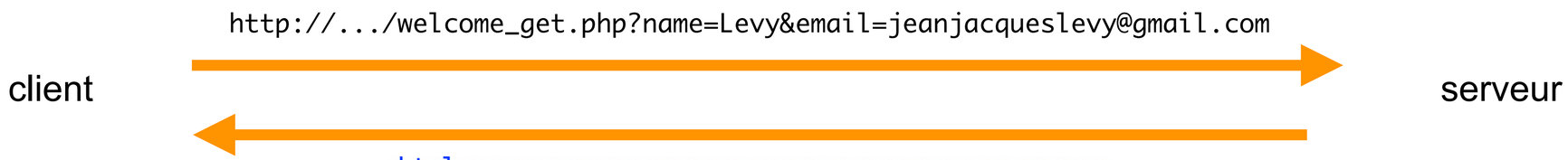
- un formulaire plus évolué

form1.html

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

welcome_get.php

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```




```
<html>
<body>

Welcome Levy<br>
Your email address is: jeanjacqueslevy@gmail.com

</body>
</html>
```

Formulaires

- l'élément <form> peut être du texte, des boutons radio, un carré de validation, un bouton d'envoi, un simple bouton (*text, radio, checkbox, submit, button*)

- les champs de texte sont de la forme:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

- voir `http://www.w3schools.com/html/html_forms.asp` pour une description détaillée

Formulaires

- l'attribut `action` insère le résultat dans la page courante ou dans une nouvelle page

```
<form action="/action_page.php" target="_blank">
```

- l'attribut `action` formule la requête dans les paramètre d'un URL (elle est alors visible de tous)

```
<form action="/action_page.php" method="get">
```

- l'attribut `action` formule la requête dans le contenu de la requête HTTP (elle n'est pas visible)

```
<form action="/action_page.php" method="post">
```

- voir `http://www.w3schools.com/html/html_forms.asp`

  pour une description détaillée

Exceptions en JavaScript

- le traitement des exceptions (*errors*) se fait avec `try` et `catch`

```
let display; // texte de l'affichage lors du "load"

function calc(){
    try {
        display.nodeValue =  eval(display.nodeValue);
    } catch (e) {
        display.nodeValue = "erreur";
    }
}

function clear(){
    display.nodeValue="";
}

function backspace(){
    display.nodeValue = display.nodeValue.slice(0,-1);
}

function addKey(key){
    display.nodeValue += key;
}
```

exemple de la calculette

- voir `http://www.w3schools.com/js/js_errors.asp`

  pour une description détaillée

Exceptions en JavaScript

- le traitement des exceptions (*errors*) se fait avec `try` et `catch`

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Error Handling</h2>

<p>How to use <b>catch</b> to display an error.</p>

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

<script>
try {
  addlert("Welcome guest!");
}
catch(err) {
  document.getElementById("demo").innerHTML = err.message;
}
</script>

</body>
</html>
```

`alert` mal épelé

- voir `http://www.w3schools.com/js/js_errors.asp`

  pour une description détaillée

Exceptions en JavaScript

- comme dans beaucoup de langages, on peut soi-même créer une exception

```
throw "Too big";    // throw a text
throw 500;          // throw a number
```

- on a aussi : `try .. finally` pour traiter le code et une exception inconditionnellement

- voir `http://www.w3schools.com/js/js_errors.asp`

  pour une description détaillée

Mode strict

- le mode strict a été rajouté à JavaScript

```
"use strict";
```

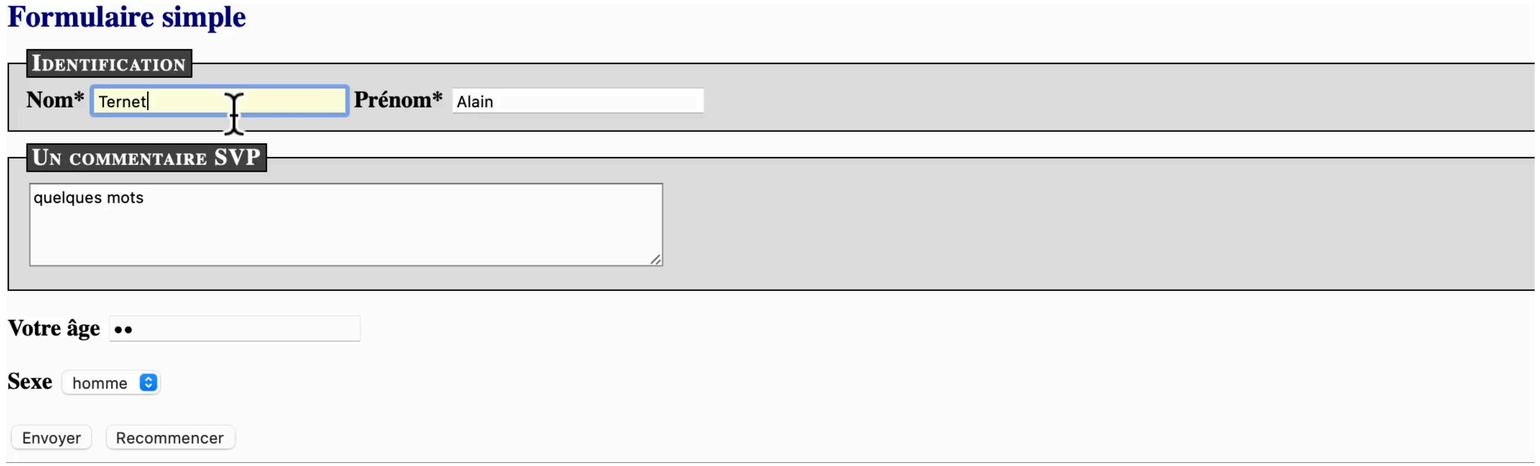
directive en première ligne du code

- toutes les variables et objets doivent être déclarés par un `let`
- on ne peut supprimer une variable par un `delete`
- `eval` est un mot-clé non redéfinissable
- etc

le but est de rendre le code plus sûr

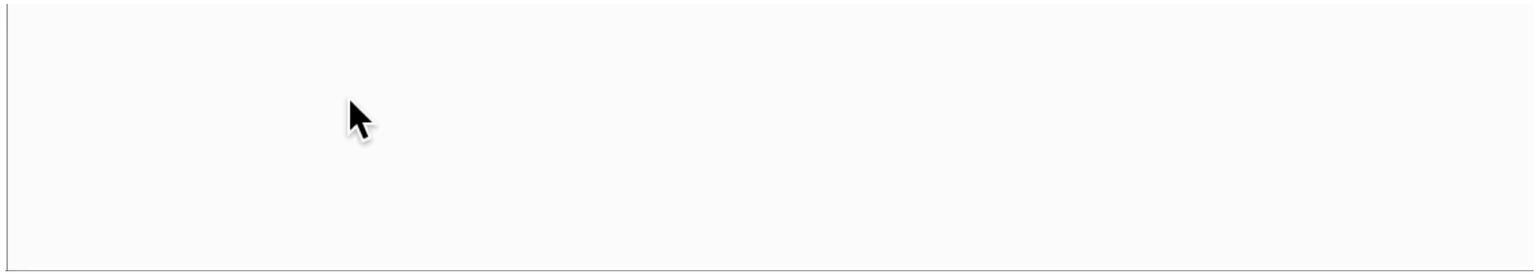
Exercices

Exercice Faire la page web donnant le formulaire ci-joint



Exercice Faire un texte d'alerte qui suit le curseur comme ci-joint

```
function displayIt(e) {
  with ($("message").style) {
     left = (e.clientX - 130) + "px";
     top = (e.clientY - 25) + "px";
     visibility = "visible";
    }
}
```



Prochain cours

- un bon tutoriel JavaScript: `http://www.programiz.com/javascript`
- un autre tutoriel JavaScript: `http://www.w3schools.com/js`
- modules, prototypes, `getter`, `setter` en JavaScript
- quelques bibliothèques JavaScript
- protocole HTTP