Scripter l'interaction avec JavaScript

Introduction à la programmation Web (3/3)

Benjamin Canou

13 avril 2018

Cours INE11 @ ENSTA

Plan de ce cours

La semaine dernière

- Historique & caractéristiques
- Premiers pas et langage de base
- Structures de données et modèle objet

Cette semaine

- Accès à et modification de la page
- Gestion d'évènements
- Requêtes HTTP depuis JavaScript

Applications Web

La mode est aux applications Web :

- Plus seulement des sites Web : le client devient intelligent.
- On veut donc programmer le navigateur :
  - Interagir avec l'utilisateur.
  - S'adapter au matériel (ex. capacités spécifiques aux mobiles).
  - Communiquer avec le serveur.
  - Modifier dynamiquement la page Web.

Accès à la représentation interne du document dans le navigateur.

- Arbre (sans cycle ni partage).
- Initialement :
  - Nœud HTML ⇝ un nœud DOM = un objet JavaScript.
  - Attribut HTML ⇝ attribut DOM = propriété JavaScript.
- Possibilité de modifier les nœuds, d'en ajouter, retirer, etc.

Comme en XML, les nœuds représentent :

- La structure (nœuds élément div, p, table) ;
- le contenu (nœuds texte) ;
- les méta-données (nœuds élément style, script, meta, etc.)

Les attributs stockent les données annexes associées aux nœuds.

- Les gestionnaires d'évènements deviennent des fonctions JavaScript.
- Les styles deviennent des objets JavaScript complexes.
- Certains attributs subissent un traitement particulier (id, class, src, etc.)

- **Un petit exemple en HTML :**

```
1 : <body>
2 :   <p style="font-family: 'Comic Sans MS'">
3 :     Hello World
4 :   </p>
5 : </body>
```

- **En version DOM :**

```
1 : var p = document.createElement ("p");
2 : var t = document.createTextNode ("Hello World");
3 : p.style.fontFamily = "Comic Sans MS" ;
4 : p.appendChild (t);
5 : document.body.appendChild (p);
```

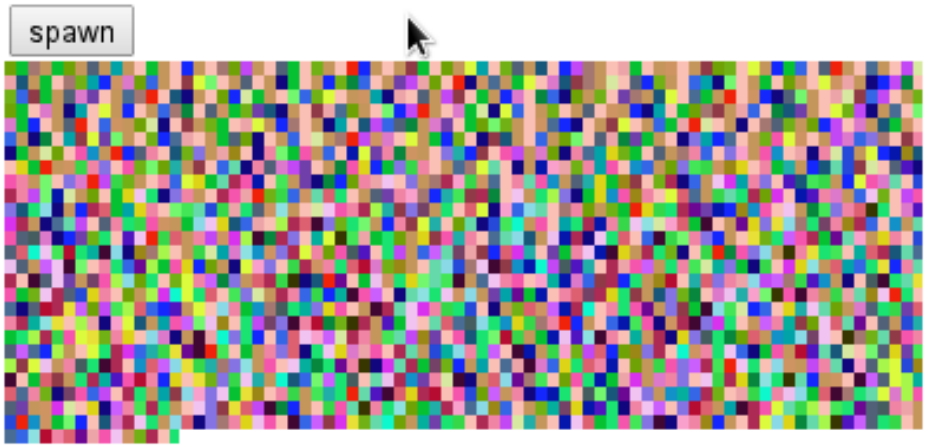
Fonctionnement des navigateurs

Modèle d'exécution : la boucle d'évènements

- Pas de mise à jour de l'affichage pendant la gestion d'évènements.
- Impossible de toucher à la queue d'évènements de l'instant courant.
- Exécution bloquée durant la gestion d'évènements.
- Déclenchement d'événement : `setTimeout(fun, delay)`.

Gestion d'évènements :

- Prendre en charge un évènement = lui affecter une fermeture JavaScript
  `window.onclick = function (evt) { ... })`
- Évènements de ressources : matériel, requêtes réseau, etc.
- Évènements d'interface : souris, clavier, etc.
- Mécanisme particulier d'inhibition/propagation d'évènements (le bullage).



En trois niveaux de complexité :

- Étape 1 : clic $\rightarrow$ une case de couleur aléatoire
- Étape 2 : clic $\rightarrow$ un processus générateur de cases
- Étape 3 : clic sur une case $\rightarrow$ disparition de la couleur et mort du processus

```
 1 : <script>
 2 :   function spawn () {
 3 :     /* chosse random color and delay */
 4 :     var r = Math.floor (Math.random() * 255);
 5 :     var g = Math.floor (Math.random() * 255);
 6 :     var b = Math.floor (Math.random() * 255);
 7 :     /* retrieve marked HTML element */
 8 :     var basket = document.getElementById ("basket");
 9 :     /* create a color block */
10 :     var li = document.createElement ("span");
11 :     li.innerHTML = "nbsp;nbsp;nbsp;nbsp;";
12 :     li.style.backgroundColor = "rgb(" + r + ",␣" + g + ",␣" + b + ")";
13 :     /* insert our new item */
14 :     basket.appendChild (li);
15 :   }
16 : </script>
17 : <button onclick="spawn␣()">spawn</button><br/>
18 : <span id="basket" style="font-size:5px"></span>
```

```
 1 :   function spawn () {
 2 :     var timeout, r, g, b /* ... */
 3 :     /* async loop */
 4 :     function loop () {
 5 :       var basket, li = /* ... with local r, g, b */
 6 :       basket.appendChild (li);
 7 :       /* delayed recursion */
 8 :       setTimeout (loop, timeout);
 9 :     }
10 :     loop ()
11 :   }
```

```
1 :   function spawn () {
2 :     var timeout = Math.floor (Math.random() * 900) + 100;
3 :     var r, g, b = /* ... */
4 :     /* store blocks */
5 :     var all = [];
6 :     /* async loop */
7 :     var stop = false;
8 :     function loop () {
9 :       if (stop) return;
10 :      var basket, li = /* ... */
11 :      li.onclick = function () {
12 :        /* clearTimeout (loop) */
13 :        stop = true;
14 :        for (i in all) basket.removeChild (all[i]);
15 :      }
16 :      all.push (li);
17 :      basket.appendChild (li);
18 :      setTimeout (loop, timeout);
19 :    }
20 :    loop ();
21 :  }
```

Programmation Client / Serveur

Objet XMLHttpRequest

Microsoft a introduit l'objet XMLHttpRequest qui permet, depuis le client, d'effectuer une requête HTTP (vers le domaine d'où provient la page).

```
 1 : function get_http (url, callback) {
 2 :   var req = new XMLHttpRequest();
 3 :   req.onreadystatechange = function() {
 4 :     if (req.readyState == 4) { // reponse OK
 5 :       if (req.status == 200) {
 6 :         callback (req.responseText) // String
 7 :         // ou
 8 :         callback (req.responseXML) // DOM
 9 :       } } }
10 :   req.open('GET', url, true);
11 :   req.send();
12 : }
```

Codage des messages

Suivant la tâche :

- **XML : parser intégré, bien pour màj des parties de documents**
- **JSON : parsing facile, lisible, proche de JavaScript**
- **Format textuel spécialisé (texte, base64, etc.)**

JSON : sous-ensemble de définition de données de JavaScript.

- **tableau d'objets entre crochets**
- **table associative entre accolades et clés suffixées par deux points**
- **nombres, booléens, chaînes**

```
1 : [ { a: 1, foo: [-1e-23, 2.0, true] },
2 :   "bla␣bla␣bla" ]
```

Analyse syntaxique :

- **Facile mais dangereux : *eval***
- **Analyseur syntaxique spécialisé :**

```
1 : JSON.parse('{"x":3,"y":4}') // -> {"x":3,"y":4}
2 : JSON.stringify({"x":3,"y":4}) // -> '{"x":3,"y":4}'
```

Pour aller plus loin

Pour aller plus loin

Bibliothèques à regarder :

- jQuery (boîte à outils portable)
- D3.js (visualisations)
- React.js, Angular.js (architecture et modularisation)
- Cordova, jQuery mobile (interfaces adaptables)

Alternatives à JavaScript :

- Sur-couches : CofeeScript, TypeScript
- Typeur: Flow
- Compilateurs d'autres langages : GWT, Js_of_ocaml

Fin

http://benjamin.canou.fr/Cours/ENSTA/INE11

benjamin@ocamlpro.com