Introduction à HTML5 - Animations et jeux

Notes personnelles du MOOC FUN

Auteur: Jérémie DECOCK
Contact: jd.jdhp@gmail.com

Version: 1
Date: November 6, 2016

Sommaire

1 Semaine 1

1.1 Vidéo 1: HTML

Définitions:

- *pages statiques*: pour une page donnée, le serveur envoi toujours le même contenu
- *pages dynamiques*: pour une page donnée, le contenu envoyé par le serveur change (d'une requête à l'autre)

Outils de validation: http://validator.w3.org

Squelette type d'une page HTML:

```
<!DOCTYPE html>
<html>
    <head>
        ...
    </head>
    <body>
        ...
    </body>
</html>
```

Balises auto-fermantes:

- <img> (notation HTML5) ou <img /> (notation XHTML)
-
 (notation HTML5) ou
 (notation XHTML)
- ...

Attributs: "src" dans `<img src="...">`

Il existe 3 types de balises:

- contenu de flux (élément de structuration): a, p, div, header, ...
- contenu de phrasé (possède une valeur sémantique): a, em, strong, ...
- metadonnées: link, style, meta, script, ...

Encodage: utf-8

```
<head>
    <meta charset="utf-8">
    ...
</head>
```

Multipurpose Internet Mail Extension (MIME):

- défini le type de fichier
- dans l'entête HTTP

1.2 Vidéo 2: CSS

CSS = Cascading Style Sheet

Outil de validation: http://jigsaw.w3.org/css-validator

1.2.1 Utilisation

- Dans le document HTML

```
<head>
    <style type="text/css">
        ...
    </style>
    ...
</head>
```

- Dans un fichier texte externe

```
<head>
    <link type="text/css" rel="stylesheet" media="screen" href="style.css">
    ...
</head>
```

1.2.2 Règles CSS

Les *règles* CSS sont composées de 3 parties:

1. le *sélecteur* (ce qui est affecté par la règle): "body" dans l'exemple suivant
2. la *propriété* (quel aspect de l'affichage est paramètré): "color" dans l'exemple suivant
3. la *valeur* (la valeur de la propriété): "purple" dans l'exemple suivant

```
body {
    color: purple;
}
```

Le sélecteur peut être:

- une balise
- une classe d'éléments (ex: `.nom` affecte les éléments `<... class="nom">`)
- un ensemble d'éléments

Exemples de sélecteurs:

- `p.classname`
- `.classname`
- `#idname`
- `p#idname` (affecte les éléments `<... id="nom">`)

1.2.3 Types d'éléments

Éléments *bloc*: div, h1, h2, p, ...

- peuvent contenir:
    - d'autres éléments bloc
    - des éléments inline
    - du texte
- placés les uns sous les autres
- alignés à gauche

Éléments *inline*: span, a, img, code, ...

- ne peuvent contenir que:
    - d'autres éléments inline
    - du texte
- placés les uns à la suite des autres sur une même ligne
- alignés à gauche

1.2.4 Positionnement

1.2.4.1 Positionnement relatif

```
position: relative;
bottom:   5px;
left:     15px;
```

- permet d'inscrire un élément bloc ou inline normalement mais avec un décalage horizontal et/ou vertical
- le contenu suivant n'est pas affecté

1.2.4.2 Positionnement flottant

```
float:  right;
width:  100px;
margin: 0px;
```

- retire un élément de son flux normal (bloc ou inline) pour le placer le plus à droite possible (right) ou le plus à gauche possible (left) à l'intérieur de son conteneur
- le texte du conteneur est adapté pour contourner la boîte flottante

1.2.4.3 Positionnement absolu

```
position: absolute;
top:      10px;
left:     50px;
width:    100px;
```

- retire l'élément concerné du flux normal
- sa position est déterminée par rapport aux limites de l'objet qui le contient

2 Semaine 2

2.1 Vidéo 1: Javascript

- Créé par Netscape en 1995
- Langage orienté objet
- *methodes* (ou *proprietes*) *DOM*: mécanisme qui permet de lire/modifier/supprimer des éléments du document HTML

- JS s'appui sur DOM pour transformer le document HTML
- *Manipulations statiques*: quand toutles les transformations (du document via DOM) sont faites côté client
- *Manipulations dynamiques*: quand les transformations font intervenir un serveur (AJAX, PHP, ...)

2.1.1 Exemple simple

```
<div id="foo">Bar</div>
<script>
    document.getElementById("foo").innerHTML = "Baz";
</script>
```

où *ìnnerHTML`* est une *propriété*.

Alternative:

```
<div id="foo">Bar</div>
<script src="baz.js"></script>
```

où le fichier *baz.js* contient:

```
/* Un commentaire */
document.getElementById("foo").innerHTML = "Baz";
```

Dans tous les cas, `<script>...</script>` doit être inscrit **après** `<div>...</div>`.

2.1.2 Langage Javascript

2.1.2.1 Variables

- les variables sont non typés
- le nom des variables peut contenir le caractère '&'

Exemple:

```
<div id="foo"></div>    <!-- conteneur vide -->
<script>
    var v1 = "foo";
    document.write(v1 + "<br>");

    v1 = 3;
    document.getElementById("foo").innerHTML = v1 + 5;
</script>
```

2.1.2.2 Tableaux

Peut stocker des éléments de type différents.

Initialisation:

```
// Methode 1

var t1 = new Array();
t1[0] = "foo";
```

```
t1[3] = 5;            // il peut y avoir des "trous" dans les indices

// Methode 2

var t1 = new Array("foo", 5, "baz");

// Methode 3

var t1 = ["foo", 5, "baz"];
```

Lecture d'un élément:

```
... = t1[0];
```

Écriture d'un élément:

```
t1[0] = ...;
```

Taille d'un tableau:

```
... = t1.length;
```

Concaténation des éléments d'un tableau:

```
... = t1.join(';');  // "foo;2;bar"
```

2.1.2.3 Fonctions

Sans arguments et sans valeur de retour:

```
function foo() {
    v1 = v1 + 3;
    document.getElementById("...").innerHTML = v1;
}

var v1 = 5;
foo();
```

Arguments:

```
function somme(a, b) {
    document.write(a + b);
}

somme(2, 3);
```

Valeur de retour:

```
function somme(a, b) {
    var res = a + b;
    return res;
}
```

```
... = somme(2, 3);
```

2.1.2.4 Portée des variables

```
var v1 = "bar";

function foo() {
    var v2 = "bar";  // Déclaration AVEC le préfixe "var" => variable LOCALE
    v3 = "bar";      // Déclaration SANS le préfixe "var" => variable GLOBALE
}

document.write(v1);  // OK: v1 est une variable globale
document.write(v2);  // ERREUR: v2 est une variable locale de foo()
document.write(v3);  // OK: v3 est une variable globale
```

2.1.2.5 Structures conditionnelles

Boucles for:

```
var t1 = ...;

for(var i=0 ; i < tab.length ; i++) {
    ...
}
```

If/then/else:

```
if(...) {
    ...
} else {
    ...
}
```

2.1.2.6 Chaines de caractère

```
var foo = "...";

... = foo.charAt(0);  // Récupère le premier caractère de la chaine
... = foo.[0];        // Récupère le premier caractère de la chaine

... = foo.length;
```

2.1.2.7 Exécuter le code JS après que le document HTML soit entièrement chargé

Souvent nécessaire pour éviter des bugs!

```
window.onload = function(){
    // code javascript concerné (les fonctions peuvent être définies en-dehors)
    ...
}
```

2.2 Vidéo 2: Outils de debug

Outils intégrés aux navigateurs: - console d'erreur - debugger - inspecteur

Console d'erreur Firefox: - outils / dev web / console web - affiche les erreurs, les avertissements et les messages console JS `console.log("...");` - permet de filtrer les message en fonction de leur provenance: réseau, CSS, JS, ... (cliquer sur les boutons pour activer/désactiver les sources de message) - permet d'écrire interactivement du code (en bas) ; il y a aussi l'*ardoise JS* pour ça

- exemple: taper le nom d'une variable dans la barre de commande du bas puis cliquer sur le nom de la variable pour afficher son contenu (contenu détaillé si c'est un objet, ...)

Debugger: - définir les points d'arrêts - utiliser les "expressions espionnes" pour voir la valeur des variables, etc. (à droite)

3 Semaine 3

...

4 Licence