Formation JavaScript

Auteur : Cyrille Tuzi

Plan de la formation

1. Outils, compatibilité et mise en place
2. Données, boucles, conditions et fonctions
3. Objets et fonctions natifs
4. Manipuler la page HTML : le DOM
5. Modifier et animer les styles CSS
6. Les événements
7. Requêtes HTTP(S) : l'AJAX

Compatibilité

Standards et doctype

- Standards CSS : depuis Internet Explorer 8
- **Standards JS : depuis Internet Explorer 9**
- Le seul doctype à utiliser :

```
<!DOCTYPE html>
```

Le cas Internet Explorer

- Internet Explorer 6 et 7 : **< 1%** d'usage
  - http://gs.statcounter.com/

- Windows XP IE (6-) **8**
  - pays en développement, grands comptes, service public
- Windows Vista IE (7-) 9
- Windows 7 IE (8-) **11**
- Windows 8 IE (10-) **11**

- **Pas de vieux IE sur mobiles et tablettes**

Polyfill

- Un "polyfill" permet d'avoir un code JavaScript unique compatible avec tous les navigateurs.
  - Polyfill.io
    https://github.com/jonathantneal/polyfill
  - jQuery
    http://jquery.com/

- Cet intérêt n'a plus de sens depuis Internet Explorer 9.

Outils

Manuels

- WebPlatform
  - http://docs.webplatform.org
- Mozilla Developer Network (MDN)
  - https://developer.mozilla.org/
- W3Schools
  - http://www.w3schools.com/

Validateurs et templates

- Validateurs
  - HTML : http://validator.w3.org/
  - CSS : http://jigsaw.w3.org/css-validator/

- Templates de démarrage propres
  - HTML : http://html5boilerplate.com/
  - CSS : http://necolas.github.com/normalize.css/

Mise en place de JavaScript

JavaScript disponible ?

- JavaScript n'est pas toujours disponible :
  - accessibilité
  - référencement
  - chargement long, voire qui échoue
  - erreurs non prévues

- Les fonctionnalités importantes d'un site doivent donc être opérationnelles sans JavaScript, qui doit seulement être une couche facultative.

Sécurité

- JavaScript se trouve côté client, et n'est donc pas sécurisé :
  - les contrôles de sécurité doivent être faits impérativement côté serveur (côté client, c'est seulement pour l'ergonomie)

Inclusion

- Pour inclure un script :

```
<script src="script.js"></script>
```

- Mauvaises pratiques :

```
<script>function myFunction() {}</script>
<a onclick="myFunction()">Lien</a>
```

Chargement et performances

- Pour un chargement rapide de la page, en production :
  - inclure **le moins de fichiers possible** (CSS, JavaScript, images, etc.)
  - **minifier** les feuilles de style et les scripts
  - sauf exceptions, **inclure le(s) script(s) en bas de page**, à la fin du corps de la page (*body*), et non dans la partie préliminaire (*head*)

Mode strict

```
'use strict';
```

- Améliore :
  - le debug, et donc, la fiabilité du code
  - les performances
  - la compatibilité avec les futures versions de JavaScript

Commentaires

- Il est important d'annoter son code :

```
// Commentaire sur une seule ligne

/* Commentaire
 * sur plusieurs lignes */
```

Bases :
les données

Les variables

- Les données sont stockées dans des variables, qu'il faut déclarer :

```
var maVariable = 10;
```

  - Pas de caractères spéciaux ni d'accents
  - Sensible à la casse (majuscules / minuscules)
  - Convention de nommage : camelCase
  - Attention aux mots réservés : utiliser des préfixes

Les nombres

```
var dataNumber = -10.5;
```

- Opérations : + - * /
- Modulo : %
- Erreur : NaN (Not A Number)

- Raccourcis :

```
dataNumber += 1;
dataNumber++;
```

Les chaînes de caractères

```
var dataString1 = "J'écris";
var dataString2 = '<p id="title">Titre</p>';
```

- Concaténation :

  `dataString1 + dataString2;`

  - Raccourci : `dataString1 += dataString2;`

Booléens et valeurs spéciales

```
var dataBoolean1 = true;
var dataBoolean2 = false;

var dataNull = null;

var dataUndefined = undefined;
```

Listes de données

- Une liste rassemble plusieurs données auxquels on attribut un nom : on parle d'index et de valeurs.
  - Tableaux : index automatiques et numériques
  - Objets : index choisis et textuels (on parle de propriétés)

Listes automatiques : tableaux

- Tableaux : index automatiques et numériques

```
var dataArray = ['valeur 1', 'valeur 2'];
```

- Usage : `dataArray[0];`
- Ajout : `dataArray.push('valeur 3');`
- Taille : `dataArray.length;`

Listes nommées : objets

```
var dataObject = {
    propriete1: 'valeur 1',
    propriete2: 'valeur 2'
};
```

- Usage : dataObject.propriete1;

Bases : boucles et conditions

Opérateurs de comparaison

- Arithmétiques : < <= >= >
- Comparaison des valeurs :
  - Egalité : ==
  - Différence : !=
- Combinaisons :
  - et : &&
  - ou : ||

Conditions

```
if (10 == dataNumber) {

} else if (dataBoolean1 && !dataBoolean2) {

}  else if (dataBoolean1 || !dataBoolean2) {

} else {

}
```

Boucles

```
while (10 > dataNumber) {

}

for (var i = 0; i < 10; i++) {

}
```

Itération

```
var dataArray = ['valeur 1', 'valeur 2'];

for (var i = 0; i < dataArray.length; i++) {

    dataArray[0];

}
```

Bases : les fonctions

Déclaration et exécution

- 1ère étape : déclaration

```
function maFonction() {
    // Traitements
}
```

- 2e étape : exécution (appel)

```
maFonction();
```

Portée des variables

```
var maVariable = 'variable globale';

function maFonction() {

    var maVariable = 'variable locale';

}
```

Paramètres

- Lors de l'exécution, on veut utiliser des paramètres variables :

```
maFonction(5, 10);
```

- On aura prévu de recevoir ces paramètres (arguments) dans la déclaration

```
function maFonction(param1, param2) {
    param1;
}
```

Valeur de retour

- Une fonction est souvent faite pour générer un résultat :

```
function maFonction() {
    return 'résultat';
}
```

  - Stoppe la fonction
  - N'affiche pas le résultat à l'écran

Fonction anonyme

- 1ère étape : déclaration

```
var maFonction = function() {
    // Traitements
};
```

- 2e étape : exécution (appel)

```
maFonction();
```

Objets natifs

L'objet principal

- Tout, en JavaScript, est contenu dans l'objet

```
window
    .alert("Texte");
    .confirm("Question ?");
    .open('url', 'nomoucible', options);
    .close();
    .print();
```

Les autres objets natifs

- Il contient tous les autres objets :

  window

.document	Page HTML
.navigator	Navigateur et demandes
.screen	Ecran
.location	URL
.history	Historique de navigation

Le document

document

.title	Titre de la page
.write()	Ecrire la page de zéro
.cookie	Accès aux cookies

- Permet l'accès aux éléments de la page via le DOM.

Le navigateur

navigator

.cookieEnabled	Cookies activés ou non
.appName	Mauvaise pratique
.userAgent	Mauvaise pratique

- Ce qui nécessite la permission de l'utilisateur, comme la géolocalisation en HTML5, est aussi gérée par le navigateur.

L'URL

location

.href	URL complète
.protocol	HTTP ou HTTPS
.hostname	Hôte (nom de domaine / IP)
.pathname	Chemin du fichier
.search	Paramètres (?param1=value1&param2=value2)
.hash	Ancre (#)

L'écran

screen

.availWidth Taille totale

.availHeight

.width Taille disponible pour le navigateur

.height

L'historique

```
history
    .back()
    .forward()
```

- En HTML5, gestion propre de l'historique.

Fonctions natives

Gestion des nombres

window

.parseInt('10...'); Transforme en nombre entier

Math

.abs(-10); Valeur absolue

.round(10.2); .floor(10.2); .ceil(10.2);

.min(10, 20); .max(10, 20);

.random(); Valeur aléatoire entre 0 et 1

Gestion des chaînes de caractères

maChaine

.indexOf('@') Première position d'un caractère

.lastIndexOf('@') Dernière position

.replace("à remplacer", "par ça")

.substr(0, 5) De telle position, tant de caractères

.substring(0, 5) De telle à telle position

.toLowerCase()

.toUpperCase()

Gestion des tableaux

monTableau

.indexOf('valeur')	Première position d'une valeur
.lastIndexOf('valeur')	Dernière position d'une valeur
.push('valeur')	Ajouter une valeur à la fin
.pop()	Retirer la dernière valeur
.unshift('valeur')	Ajouter une valeur au début
.shift()	Retirer la première valeur
.sort()	Trier par ordre alphabétique
.reverse()	Inverser l'ordre

Chaîne < > tableau

- Transformer un texte en tableau :

```
maChaine.split(',');
```

- Transformer un tableau en chaîne :

```
monTableau.join(',');
```

Gestion des dates

- Timestamp actuel :

```
Date.now;
```

Manipulation de la page : le DOM

Sélection

```
document
    .getElementById('mon-identifiant');
    .querySelector('#mon-identifiant');

    .getElementsByTagName('h1')[0];
    .getElementsByClassName('ma-classe')[0];
    .querySelectorAll('#mon-identifiant p')[0];
```

Navigation par l'arborescence

- DOM = Document Object Model
  - Arborescence d'objets

  - Remonter : .parent;
  - Descendre : .children[0];
  - Suivant : .nextElementSibling;
  - Précédent : .previousElementSibling;

Attributs et contenu

- Lecture et modification des attributs :

  .getAttribute('value');

  .setAttribute('value', 'nouvelle valeur');

- Lecture et modification du contenu:

  .innerHTML; / .textContent;

Ajout de contenu direct

```
elementParent.insertAdjacentHTML('position', '<p>Texte</p>');
```

Positions :

- beforebegin
- afterbegin
- beforeend
- afterend

Ajout de contenu via le DOM

- 1ère étape : création des éléments

```
document
    .createElement('p');
    .createTextNode("Texte");
```

- 2e étape : placement dans la page

```
elementParent
    .appendChild(nouvelElement);
    .insertBefore(nouveau, reference);
```

Remplacements et suppressions

```
elementParent
    .replaceChild(nouvel, ancien);
    .removeChild(enfant);

element.removeAttribute('target');

element.cloneNode(true);
```

Tests

- Vérifier un élément :

  .tagName;

  .matches['.selecteur'];

- Vérifier un attribut :

  .hasAttribute('target');

- Vérifier par rapport à d'autres éléments :

  .hasChildNodes();

  .contains(element);

Formulaires

form.

.email / elements.email / elements[0] / length

input.

defaultValue / value

checkboxOrRadio.

defaultChecked / checked

selected.

options[0] / selectedIndex

option.

defaultSelected / selected

Animer les styles CSS

Modifier un style CSS

```
element
    .style.borderTop = '1px solid red';
```

Modifier les classes

```
element.classList
    .add('ma-classe');
    .remove('ma-classe');
    .toggle('ma-classe');
    .contains('ma-classe');
```

Lire les styles variables

- Dimensions réelles :
  - .offsetWidth;
  - .offsetHeight;
- Position relative :
  - .offsetTop;
  - .offsetLeft;

Actions différées ou répétées

```
var differed = setTimeout(function(){
                    // Traitements
                }, 2000);

var repeated = setInterval(function(){}, 2000);
```

- Les temps sont toujours exprimés en millisecondes.

Annuler un timer

```
clearTimeout(differed);

clearInterval(repeated);
```

Les événements

Les événements

- Navigation :
  - **click**, scroll, hashchange
- Souris / survol :
  - **mouseover**, **mouseout**, mousemove
- Clavier :
  - keydown, **keyup**, **input**
- Formulaire :
  - **focus**, **blur**
  - select, change
  - **submit**

Écouter un événement

- Commencer l'écoute :

  element.**addEventListener**('click', maFonction);

- Arrêter l'écoute :

  element.**removeEventListener**('click', maFonction);

Chargement de la page

```
document.addEventListener(
    'DOMContentLoaded', function() {

    /* Traitements */

});
```

Origine de l'événement

```
e.addEventListener('focus', function() {
    this.value = '';
});

e.addEventListener('focus', function(event) {
    event.target;
});
```

Comportement par défaut

```
e.addEventListener('click', function(event) {

    event.preventDefault();

});
```

Informations sur l'événement

event.

target / currentTarget / relatedTarget

timeStamp / type

keyCode / altKey / ctrlKey / shiftKey

clientX/Y / pageX/Y / screenX/Y

Requêtes HTTP : l'AJAX

AJAX

- **A**synchronous **J**avaScript **A**nd **X**ML

- Une **requête HTTP**(S), c'est :
  - une **méthode** HTTP, une URL et des paramètres

  - des en-têtes HTTP de demande (navigateur)
  - des en-têtes HTTP de réponse (serveur), dont :
    - un code HTTP de réponse (200, 404, etc.)
    - le format du contenu (Content-Type)

  - une réponse (texte, HTML, XML, JSON, etc.)

Méthodes HTTP

- **GET** : opérations de lecture
  - paramètres dans l'URL
    - limite de taille et de caractères spéciaux
    - enregistré par le navigateur (historique) et les proxys (entreprise, FAI, etc.)
  - ex. : charger un contenu, recherche, etc.

- **POST** : opérations de modification
  - paramètres non visibles dans l'URL
  - ex.: connexion, inscription, commentaire, etc.

Requête POST

```
var ajax = new XMLHttpRequest;

ajax.open('POST', '/chemin/script.php');

ajax.send
('param1=value1&param2=value2');
```

Requête GET

```
var ajax = new XMLHttpRequest;

ajax.addEventListener('readystatechange', function () {
    if ((4 === this.readyState) && (this.status >= 200) &&
(this.status < 400)) {
        this.responseText;
    }
});

ajax.open('GET', '/chemin/script.php?param1=value1');

ajax.send();
```

Formats particuliers

- JSON :

```
JSON.parse(this.responseText);
```

- DOM XML:

```
this.responseXML;
```

AJAX avec jQuery

```
$.get('http://exemple.com/script.php',
function(response){
    // Traitements
});

$.post('http://exemple.com/script.php',
function(response){});
```

Paramètres

- Paramètres manuels :

```
$.get('url', {page: 2}, function(response){});
```

- Récupérer le contenu d'un formulaire :

```
$.post('url', $('#form-id').serialize(), function(response){});
```

Historique avancé

- Nouvelles méthodes dans l'objet history :
  - **pushState**()
  - replaceState()
- Paramètres :
  - objet, stocké dans history.state
  - Titre
  - URL classique
- Nouvel événement :
  - **popstate**