Programmation Web – Avancé

BINV2150 A : JavaScript (& JAVA SERVLETS)

Week 5

R. Baroni / J.L. Collinet / C. Damas

Attribution – Partage dans les Mêmes Conditions 4.0 International (CC BY-SA 4.0)

Presentation template by SlidesCarnival

0

Table des matières

Tous les sujets traités pendant ce cours...

Table des matières

Table des matières

3

Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Découvrons le langage côté client...

Table des matières :

3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Table des matières :

3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Table des matières :
3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Table des matières :
3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Les objets en JS

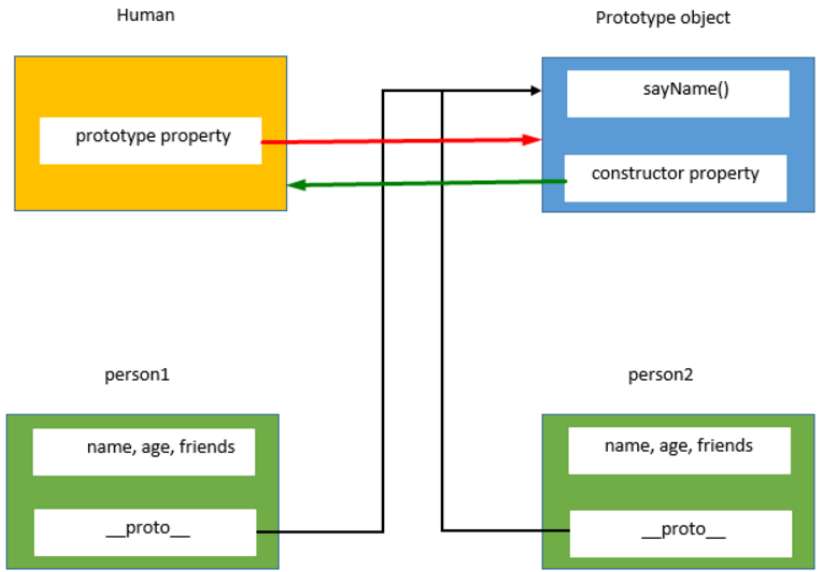
- Prototype-based language (VS class-based, comme Java)
- La plupart des éléments du langage sont des objets : chaînes, tableaux, APIs du browser, objets personnalisés...
- Pas de distinction entre une classe et une instance

Les objets en JS

- Tout objet peut être le prototype d'un autre objet, permettant à l'autre objet de partager les propriétés du 1er objet
- Détails : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model

Les objets en JS

- Understanding prototypes [10]



- Object specific props & methods in constructor
- Shared props & methods in prototype
- JS look for a property on the object, then up the chain through the dunder proto(s)

Création d'un objet

- Via un « object litteral » (liste de paires nom/valeur)

```
var nomObjet = {
propriété1 : valeur ,
propriété2 : valeur ,
propriété3 : valeur ,
...
propriétéz : valeur
}
```

Création d'un objet

- Via **new** ou **{}**

```
var nomObjet = {} ; // équivalent de var nomObjet = new Object();
nomObjet.propriété1 = valeur ;
```

Accéder aux propriétés d'un objet de manière externe

- Soit via un point

```
utilisateur.nom ;
```

- soit via des accolades

```
utilisateur['nom'] ou utilisateur['adresse']['rue']);
```

Les objets en JS

- **DEMO-09** : création d'un objet représentant une personne spécifique, avec les différentes techniques.

Création d'un "modèle objet" ("classe") et accès de manière interne aux propriétés de cet objet

1. Création d'un "modèle objet" via le type de function "class" et la méthode **constructor()**
   - Nouveau depuis ES6
   - Node.js : attention au **'strict mode'**
   - **constructor()** pas supporté par IE11 !
   - The end of life of IE11 [5] : https://medium.com/@burger.neal/the-end-of-life-of-internet-explorer-11-12736f9ff75f
   - 1 seul constructeur possible sinon erreur

Création d'un "modèle objet" ("classe") et accès de manière interne aux propriétés de cet objet

1. Création d'un "modèle objet" via le type de function "class" et la méthode **constructor()**

```
class nomModelObjet {
  constructor(property1) {
    this.property1 = property1;
          this.property2 = function() {
            return ... this.property1 ...
            }
          };
}
```

Création d'un "modèle objet" ("classe") et accès de manière interne aux propriétés de cet objet

2. Création d'un "modèle objet" via une fonction constructeur

```
function nomModelObjet(property1) {
        this.property1 = property1;
        this.property2 = function() {
                return ... this.property1 ...
        };
}
```

Création d'un "modèle objet" ("classe") et accès de manière interne aux propriétés de cet objet

3. Création d'un "modèle objet" via une fonction normale

```
function nomModelObjet(property1) {
        var obj = {};
        obj.property1 = property1;
        obj.property2 = function() {
                return ... this.property1 ...
        };
        return obj;
}
```

Getter & setters

```
"use strict"; // Ca permet, par exemple, d'éviter qu'un getter, sans setter, soit assigné
class Person{
        constructor() {
                this.id = 'id_1';  }
        set name(name) {
                this._name = name.charAt(0).toUpperCase() + name.slice(1);
}
        get name() {
                return this._name;  }
        sayHello() {
        console.log('Hello, my name is ' + this.name + ', I have ID: ' + this.id);  }
}
```

Strict mode (ES5)

- Plus de feedback d'erreurs sur le code
- Déclaration : **"use strict";** au début d'un script ou d'une fonction
- Détails : https://www.w3schools.com/js/js_strict.asp

Création d'une instance et accès à ses propriétés via l'appel au constructeur

```
nvar nomInstance = new nomModelObjet(value1);
nomInstance.property1;
```

Les objets en JS

- **DEMO-10** : création d'un modèle objet « Car » et d'une de ses instances.

Introduction aux modules (ES6)

- Un module = librairie JS, fournissant des fonctions et / ou des objets
- Inclure un module :
  - **import** ne peut être utilisé que si dans **scriptName.js** : ajout de type='module'

```
<script src="./DEMO-11A.js" type="module"></script>
```

Introduction aux modules (ES6)

- Inclure un module :
  - Dans un script **scriptName.js** :

```
import {Car} from "./DEMO-11A-CAR.js"; // import named export
import auto from "./DEMO-11A-CAR.js"; // importing defaults
```

Introduction aux modules (ES6)

- Créer un module :
  - Créer un fichier **nomModule.js**
  - Au sein du fichier **nomModule.js**, utiliser **export** pour rendre les propriétés et méthodes disponibles en dehors du fichier module :

```
export {Car};
// ou
export default Car;
```

Introduction aux modules (ES6)

- Détails :
  - Import: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import
  - Export : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/export

Introduction aux modules (ES6)

- **DEMO-11A** : Utilisation basique de notre objet “Car” au sein d’une page HTML.

Introduction aux modules sous Node.js

- Un module = librairie JS, fournissant des fonctions et / ou des objets.
- Inclure un module intégré ou installé :

```
const shortid = require("shortid");
```

- Inclure un propre module :

```
var qqch = require('./Car.js') ;
```

Introduction aux modules sous Node.js

- Créer un propre module :
  - Créer un fichier **nomModule.js**
  - Au sein du fichier **nomModule.js**, utiliser **exports** pour rendre les propriétés et méthodes disponibles en dehors du fichier module
- Tous les fichiers nécessaires pour un module sont contenus dans un package

Introduction aux modules sous Node.js

- Installer un package (via terminal) : **npm install package_name**
- NPM : gestionnaire de packages : www.npmjs.com
- **DEMO-11B** : Génération aléatoire d'un ID à l'aide d'un package NPM (shortid par exemple) en reprenant la classe « Car » **DEMO-10**.

Introduction aux modules sous Node.js

- **DEMO-11C** : Que se passe-t-il si l'on essaie d'utiliser un package npm côté client ? Reprise de la **DEMO-11A**

Introduction aux modules (node.JS) mis à disposition du browser (ES6)

- Package bundlers :
  - Faciliter l'accès aux packages disponibles sur le web côté client en utilisant des packages faits pour Node.js
  - Utilisation de la méthode **require** côté client
  - Gestion aisée des packages et de leurs dépendances...
- Parmis les plus connus : webpack, browserify

Introduction aux modules (node.JS) mis à disposition du browser (ES6) : webpack

- Créer un package.json pour votre application (après création d'un répertoire)
  - npm init –y
  - Assurer que nos packages soient privés :
    + "private": true,
  - Enlever le « main entry »: – "main": "index.js«
- Installer webpack via npm :
  - npm i webpack -D
  - npm i webpack-cli -D

Introduction aux modules (node.JS) mis à disposition du browser (ES6) : webpack

- Créer une structure standard pour votre application

```
./package.json
./webpack.config.js
./dist
    ./dist/index.html
./src
    ./src/index.js
```

Introduction aux modules (node.JS) mis à disposition du browser (ES6) : webpack

- Inclure votre « bundle file » généré par webpack au sein de ./dist/main.js dans votre fichier HTML

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

Introduction aux modules (node.JS) mis à disposition du browser (ES6) : webpack

- Générer le bundle sans fichier de configuration : npx webpack

Introduction aux modules (node.JS) mis à disposition du browser (ES6) : webpack

- Créer un ficher de configuration webpack.config.js à la racine :

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'main.js',
    path: path.resolve(__dirname, 'dist')
  }
};
```

```
./package.json
./webpack.config.js
./dist
    ./dist/index.html
./src
    ./src/index.js
```

Introduction aux modules (node.JS) mis à disposition du browser (ES6) : webpack

- **DEMO-11D** : Mettre en oeuvre webpack pour inclure le package **shortid** dans la classe « Car » /Reprise de la **DEMO-11C** pour la faire fonctionner :)

Introduction aux modules (node.JS) mis à disposition du browser (ES6) : webpack

- NPM scripts : utiliser **npm run dev** ou **npm run build** au lieu de **webpack-cli**. Dans **./package.json**
  - \+ **"build": "webpack"** ou
  - \+ **"build": "webpack --mode production"**
  - \+ **"dev": "webpack --mode development"**

- Tout savoir sur webpack : https://webpack.js.org/guides/

Exercices

A vous de jouer...

Les objets en JS, les modules ES6 et les modules Node.js

- **EX-06A** :
  - A) Structurez votre projet pour utiliser Webpack, installer les packages associés à Webpack.
  - B) Création de votre modèle objet (ou classe) : Création du modèle objet “Student” et implémentation de plusieurs instances de ce modèle objet. On indiquera à la création d’un objet son nom, prénom, orientation (« Informatique de gestion », « biologie »…). Veuillez formatez le nom et prénom pour que la 1ère lettre soit une majuscule. Un ID sera généré de manière aléatoire à la création de l’objet, sur base d’un package NPM de votre choix. Une méthode (**details()**) renverra toutes les propriétés de l’étudiant (les afficher également dans la console). Exécutez votre script student.js avec node.

Les objets en JS, les modules ES6 et les modules Node.js

- **EX-06A** :
  - C) Créer une page HTML permettant, via un formulaire, de créer des instances d'étudiants.
  - D) Affichez au sein de votre page HTML, lors de l'ajout d'un étudiant, la liste de tous les étudiants ajoutés (en faisant appel à la méthode **details()**).
  - E) Faites un build de DEV via **npm run dev** : observer les fichiers générés (et leur taille), débuggez votre application
  - F) Faites un build de PROD via **npm run build** : observer les fichiers générés (et leur taille)
  - NB : webpack tutorial : https://www.valentinog.com/blog/webpack/

Les objets en JS, les modules ES6 et les modules Node.js

- **EX-06B** (optionnel) :
  - A) Reprenez l'exercice précédent et peaufinez l'aspect design : ajoutez deux photos à votre application, ainsi qu'un CSS.
  - B) Expérimentez avec webpack l'Asset Management pour charger vos photos et votre CSS, ainsi qu'avec son serveur de développement pour mettre à disposition votre application.
  - C) Utilisez d'autres packages npm pour compléter le modèle objet « Student ».

Références

[1]	MDN web docs, Introduction to web APIs. Lien : https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction
[2]	MDN web docs, JavaScript Guide. Lien : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
[3]	w3schools.com, JavaScript Tutorial. Lien : https://www.w3schools.com/js/default.asp
[4]	tutorialspoints.com, Javascript Tutorial : Lien : https://www.tutorialspoint.com/javascript/index.htm

Références

[5]	Medium.com, Neal Burger, The end of life of IE11. Lien : https://medium.com/@burger.neal/the-end-of-life-of-internet-explorer-11-12736f9ff75f
[6]	w3schools.com, JS HTML DOM. Lien : http://www.w3schools.com/js/js_htmldom.asp
[10]	medium.com, Prototypes in JavaScript, Rupesh Mishra. Lien : https://medium.com/better-programming/prototypes-in-javascript-5bba2990e04b