Programmation Web Avancée et Mobile

CM 2 : Bibliothèques et frameworks côté client

Aurélien Tabard - Lionel Médini

Plan du cours

- Rappels JavaScript
- Introduction aux principes partagés par les frameworks JavaScripts modernes
- Introduction à Vue

Ressources

Page à compléter :

- Liens utiles : https://aurelient.github.io/mif13/2018/hack

Livres

- Eloquent Javascript (3rd edition EN)
- Eloquent Javascript (1e edition FR)

Rappels JavaScript

- Faiblement typé
  Conversions de types implicites, opérateur ===
- Fonctionnel
  Basé sur le scope des variables
- Orienté évènement
  Trick : une callback peut être une alternative aux threads
- Orienté prototype
  Les objets peuvent partager un prototype, mais pas d'héritage entre classes

→ Langage très permissif ("assembleur du Web")

Rappels JavaScript

À quoi sert la notation suivante ?

```
(function() {
        var toto = 12;
        console.log(toto);
})();
```

Rappels JavaScript

À quoi sert la notation suivante ?

```
(function() {
        var toto = 12;
        console.log(toto);
})();
```

Comprendre la portée

Closures (fermetures)

- Permet de capturer l'environnement d'une fonction.
  C'est-à-dire les variables des scopes externes à celui de la fonction.

```
function makeFunc() {
  var name = "Mozilla";
  function displayName() {
    alert(name);
  }
  return displayName;
}

var myFunc = makeFunc();
myFunc();
```

Function factory

- Permet de passer des paramètres au moment de la création d'une fonction

```
function creerAdditionneur(a) {
  return function(b) {
    return a + b;
  }
}
var add5 = creerAdditionneur(5);
var add20 = creerAdditionneur(20);
```

À quoi servent les Closures ?

- Gérer la nature asynchrone de JavaScript
  - Ajouter des données locales à un callback
- Gérer des "objets"
- Émuler des méthodes privées

En savoir plus : Closures (MDN)

Web apps

Années 2005-2010 : bibliothèques JS

Milieu des années 2000 : bascule d'un Web centré documents et formulaires vers un Web centré application

- Essor des applications Web riches (RIA)
- Toujours beaucoup de logique côté serveur
- Développement d'AJAX et de REST
- Bibliothèques très permissives

Années 2005-2010 : bibliothèques JS

La plus emblématique : **JQuery**

- Implémentations incomplètes des standards existants
- Standards eux-mêmes incomplets

→ Besoin de "workarounds" pour homogénéiser les comportements des navigateurs

Années 2010 : frameworks JS

- Émergence des Single Page Applications
- Déplacement de la logique sur le client
- Structuration du code
- Développement du "tooling" : Paquets, CSS, Javascript.

Les frameworks JS aujourd'hui

- Angular
- React
- **Vue**
- Emberjs
- Meteor
- Backbone
- ...

The ultimate guide to javascript frameworks
Liste sur wikipedia

Frameworks côté client

Objectif

- Faciliter le développement d'applications "single-page" (SPA) côté client

Propriétés

- un pattern MV*
- une sorte de pattern IoC
- réactives

Frameworks côté client

Caractéristiques :

- Interceptent le changement d'URL (hash)
- S'appuient sur une structure modulaire
- S'appuient sur d'autres bibliothèques
  - Gestion des objets JS
  - Gestion des événements
  - Templating

Principe 1 : Routage

Objectif : Simuler des pages web différentes

- Intercepter le changement de hash dans l'URL
- Récupérer les - éventuels - paramètres
- Déclencher un callback

Moyens :

- événement `hashchange`
- hash `window.location.hash`

Principe 2 : Liens entre modèle et vue

Voir cours reactive programming pour plus de détails

One-way data binding

- Une action sur la vue provoque la mise à jour du modèle

Two-way data binding

1. Une action sur la vue provoque la mise à jour du modèle
2. Toute modification d'une propriété du modèle provoque une mise à jour de la vue

Principe 3 : Templating

Vu au dernier semestre avec Mustache

- Interpolations (texte, variables, expressions JS)
  - {{ ok ? 'YES' : 'NO' }}
- Directives (if, for, on)
  - `<p v-if="seen">Now you see me</p>`

Principe 4 : Composants

Un des principes les plus utiles.

- Permet de créer des composants réutilisables à travers toute l'application (et même partageable entre projets).
- Une application Vue est dotée d'un arbre de composants : un parent et plusieurs enfants.
- Voir aussi :
  - Polymer
  - Web components

Introduction à Vue.js

Tooling

- npm
- vue-cli
- Webpack
- ESlint

Retour sur le TP de la semaine dernière

Exposés

Exposés

- Tout le monde inscrit ce soir.

La prochaine séance :

- Django
- TypeScript

-> ordre de passage en ligne demain matin.