YEOMAN BACKBONE BOX2D

FRANÇOIS MICHAUDON & HUGO MALLET

OBJECTIFS

1. SAVOIR UTILISER L'OUTIL YEOMAN

1. STRUCTURER UNE APPLICATION AVEC BACKBONE.JS

1. AJOUTER UN MOTEUR PHYSIQUE A TOUT ET N'IMPORTE QUOI !!!

=> PROGRAMMER DE BELLES WEBAPPS EN JAVASCRIPT

YEOMAN ?

YEOMAN

C’est quoi?

Un outil qui regroupe plein d’outils…

Ca sert à quoi?

A gagner du temps! En automatisant les tâches répétitives de la création web.

Ca vient d’où?

Depuis Mai 2012, d’une communauté de développeurs reconnus dans le monde du web.

YEOMAN
LES OUTILS

- GRUNT
- TWITTER BOWER
- BOILERPLATE
- NODE
- COFFESCRIPT
- MOCHA
- PHANTOMJS
- REQUIRE.JS
- OPTIPNG

ETC…

YEOMAN
PERMET DONC DE

- Démarrer un projet en quelques secondes
- Compiler automatiquement les fichiers coffee, sass…
- Détecter les erreurs Javascript
- Démarrer un serveur de dev
- Recharger la page lorsqu'une modification est détectée
- Optimiser les images
- Minifier tous les fichiers javascript, css, html
- Gérer les dépendances
- Lancer des tests

ETC..

YEOMAN INSTALLATION

Disponible pour: Linux, Mac OS, Windows

Instructions

http://yeoman.io/installation.html

https://github.com/yeoman/yeoman/wiki/Manual-Install

Et notre TP…

YEOMAN
DEMARRER

1. **Créer un projet**

   C’est à dire créer un dossier et se positionner dedans

   | mkdir MyProject

   | cd MyProject

2. **Initialiser le projet**

   | yeoman init

3. **Prévisualiser le projet**

   | yeoman server

YEOMAN
ARCHITECTURE BASIQUE

MyProject/

- app/
  - index.html
  - .htaccess
  - scripts/
    - main.js
  - styles/
  - …
- test/
- Gruntfile.js
- …

YEOMAN GENERATEURS

| yeoman init

| yeoman init << generatorName >>

Exemples:

| yeoman init angular

| yeoman init backbone

| yeoman init backbone:view customer

| yeoman init backbone:model customer

YEOMAN
COMMANDES UTILES

| yeoman server

| yeoman build

| yeoman test

| yeoman server:dist

| yeoman install / update

YEOMAN
POUR ALLER PLUS LOIN

http://yeoman.io/index.html

https://github.com/yeoman/yeoman

http://gruntjs.com/

http://compass-style.org/

http://coffeescript.org/

ETC…

BACKBONE ?

C’est quoi?

Une librairie Javascript.

BACKBONE.JS

Ca sert à quoi?

A développer des applications web notamment des single-page webapps.

Ca vient d’où?

Depuis fin 2010, initialement développé par le créateur de coffeescript et underscore.js

BACKBONE STRUCTURE

DESIGN MV* (~ MVC)

BACKBONE CORE:

- **MODEL**
- **COLLECTION**
- **VIEW**
- **ROUTER**

DEPENDANCES: underscore.js

DOM MANIPULATION OPTIONS: jQuery / Zepto

BACKBONE MODEL

Définition d’un modèle:

```
var MyModel = Backbone.Model.extend({
        … définition …
});
```

Initialisation d’un modèle:

```
var book = new Book({
        title: « One thousand and One Night »,
        author: « Scherazade »
});
```

BACKBONE MODEL (SUITE)

Getters

```
console.log( book.get(« title ») );
```

Setters

```
// Un seul attribut
Book.set(« content », «  Lorem ipsum dolor sit amet… »;
// Plusieurs attributs via JSON
book.set({
        title: «  One Day »,
        content: « Lorem ipsum dolor sit amet… »
});
```

BACKBONE
MODEL EVENTS

Les modèles sont sensibles aux événements et déclenchent des événements.

```
var BookModel = Backbone.Model.extend({
        initialize: function() {
                this.on(« change:title », function() {
                        console.log(« Le titre du livre a changé »);
                });
        }
});
```

this.on() -> enregistre un écouteur d'événements
« change:title » -> le modèle a déclenché un événement

BACKBONE
MODEL SYNC

Récupèrer les attributs du modèle sur le serveur:

```
model.fetch();
```

Sauvegarder le modèle sur le serveur:

```
model.save();
```

Supprimer le modèle côté serveur (et côté client) :

```
model.destroy();
```

BACKBONE
MODEL SYNC (SUITE)

Comment Backbone connaît la requête à effectuer pour synchroniser le modèle client et le modèle sur le serveur?

⇒ RESTful JSON Interface

Autrement dit, si votre serveur suit la norme REST et est capable de recevoir et produire des données au format JSON, Il n'y a RIEN D'AUTRE A ECRIRE !!!

Exemple d'URL appelée après un comment.fetch():

http://monserveur.com/articles/14/comments/36

Où 36 == model.id

BACKBONE COLLECTION

Les collections sont des sets de models

Définition:

```
var Library = Backbone.Collection.extend({
        model: Book,
        …
});
```

Initialisation:

```
var students = new StudentCollection([
        {name: « Jean », id: 1}, {name: « Elisa », id: 2}
]);
```

BACKBONE
COLLECTION METHODS

Quelques méthodes sur les collections

```
collection.add( model );
collection.remove( model );
collection.reset();
collection.where( attributes );
collection.sort();
…
```

Plus toutes les méthodes de underscore.js:

forEach, sortBy, map, find, last, filter, ….

BACKBONE
COLLECTION SYNC

Comme pour les modèles, les collections peuvent être synchronisés avec le serveur:

```
collection.fetch();


collection.create( {
        name: « Tim »
} );
```

BACKBONE VIEW

Les vues correspondent à l'interface utilisateur, elles sont attachés au DOM et peuvent interagir avec un modèle ou une collection.

Définition:

```
var BookView = Backbone.View.extend({
        tagName: «  li »,
        ….
});
```

Initialisation:

```
var oneDayView = new BookView({
        model: oneDayBook
});
```

BACKBONE
VIEW DOM ELEMENT

Une vue correspond à un élément du DOM.

Elle peut être attachée à un élément existant:

```
var Cart = Backbone.View.extend({
        el: « #cart-id »
});
```

Elle peut créer un élément:

```
var CartProduct = Backbone.View.extend({
        tagName: « li »,
        className: « cart-item »
});
```

BACKBONE
VIEW TEMPLATES

Les vues peuvent utiliser le système de template fourni par la libraire underscore.js

Exemple de définition d'un template:

```
<script type=« text/template » id=« cart-item-template »>
<li class=« cart-item »> <%= label %> </li>
</script>
```

Utilisation d'un template dans une vue:

```
var CartProduct = Backbone.View.extend({
        template: _.template( $(« #cart-item-template »).html() )
});
```

BACKBONE
VIEW TEMPLATES (SUITE)

Que signifie « <%= label %> » dans le slide précédent?

Cette expression sera remplacée via le moteur de template de underscore.js par la valeur de label.

Exécution du template:

```
var CartProduct = Backbone.View.extend({
        template: _.template( $(« #cart-item-template »).html() ),
        render: function() {
                this.$el.html( this.template( this.model.attributes) );
        }
});
```

BACKBONE
VIEW MODEL EVENTS

Une vue observe son modèle attaché.

Mettre en place l’écouteur d’événements:

```
var CartProduct = Backbone.View.extend({
        initialize: function() {
                this.model.on(« change:quantity », this.render, this);
        }
});
```

BACKBONE
VIEW USER EVENTS

Une vue écoute les événements déclenchés par l'utilisateur.

Mettre en place l'écouteur d'événements:

```
var CartProduct = Backbone.View.extend({
        events: {
                « click .addOne »: « addOne »
        },
        addOne: function() {
                var newQuantity = this.model.get(« quantity »)++;
                this.model.set(« quantity », newQuantity);
        }
});
```

BACKBONE ROUTER

SINGLE PAGE APPLICATION

⇒ **Les actions de l'utilisateur n'entrainent pas un changement de l'URL**

Problème: Comment conserver un état de mon application et le reproduire après, par exemple, un refresh de la page?

Le router va nous permettre de gérer les états de notre application et de lier certains événements à une URL.

BACKBONE ROUTER (SUITE)

Définition:

```
var ArticleRouter = Backbone.Router.extend({
        routes: {
                « help »:           « showHelp »
                « articles/:id »:   « showArticle »,
                « *other »:                 « defaultRoute »
        },

        showArticle: function(id) {
                …
        }
});
Backbone.history.start();
```

BACKBONE
ROUTER INITIALISATION

Initialisation:

new ArticleRouter;

Démarrage:

Backbone.history.start();

BACKBONE
ROUTER METHODS

Ajout manuel d’une route:

myRouter.route(/^(.*?)\/special$/ , « special », function() {});

[Utilisation d’une expression régulière]

Mise à jour de l’URL:

this.navigate(« comment/ » + id);

Mise à jour de l’URL en déclenchant l’événement correspondant:

myRouter.navigate(« cart », {trigger: true});

BACKBONE
POUR ALLER PLUS LOIN

Documentation officielle:

http://backbonejs.org/

http://underscorejs.org/

Tutos:

http://weblog.bocoup.com/

Exemple:

http://backbonejs.org/docs/todos.html

BOX2D ?

C’est quoi?

Une librairie de moteur physique 2D.

Ca sert à quoi?

A simuler une gravité, une collision …

Ca vient d’où?

Initialement créée en C++, la librairie a été portée dans plusieurs langages dont récemment en Javascript.

BOX2D
OBJETS FONDAMENTAUX

- **World: le monde box2d, c'est à dire tous les objets, contraintes, joints créés**
- **Shape: les formes (cercles, rectangles, …)**
- **RigidBody: « les corps de matières »**
- **Fixture: lien entre shape et body avec des propriétés comme la densité, la friction**
- **Constraint: contraintes appliquées aux corps**
- **Joint: containtes liant les corps entre eux**
- **Solver: le moteur qui met à jour les propriétés des objets toutes les x secondes en fonction des contacts, contraintes …**

BOX2D
THE WORLD

Initialisation:

```
// Définition du vecteur de gravité (dans le plan 2D)
var gravity = new Box2D.Common.Math.b2Vec2( 0, 10 );

// Augmente les performances
var doSleep = true;

// Initialisation du monde Box2D
var world = new Box2D.Dynamics.b2World( gravity, doSleep );
```

BOX2D FIXTURES

Définition d’une fixture avec une shape rectangulaire:

```
// Ne pas utiliser directement des pixels, prendre une échelle plus petite
var SCALE = 30, width = 400, height = 100;
// Propriétés de la fixture
var fixtureDef = new Box2D.Dynamics.b2FixtureDef;
fixtureDef.density = 1.0;
fixtureDef.friction = 0.5;
fixtureDef.restitution = 0.2;
// Forme de la fixture
fixtureDef.shape = new Box2D.Collision.Shapes.b2PolygonShape;
fixtureDef.shape.SetAsBox( width / SCALE, height / SCALE);
```

BOX2D
BODY

Création d’un corps avec la fixture précédente

```
var x = 200, y = 1000, angle = 0; // angle en radians
// Définition du body
var bodyDef = new Box2D.Dynamics.b2BodyDef;
bodyDef.type = Box2D.Dynamics.b2Body.b2_staticBody; // b2_dynamicBody
bodyDef.position.x = x / SCALE; // Position du centre de l’objet!
bodyDef.position.y = y / SCALE;
bodyDef.angle = angle;
// Création du body dans le monde
var body = world.CreateBody( bodyDef );
// Ajout de la fixture au body (donc de sa forme et ses caractéristiques)
body.CreateFixture( fixtureDef );
```

BOX2D
BODY METHODS

body.GetPosition(); // Retourne le centre de la forme dans le monde

body.GetAngle();

body.SetType

body.SetPositionAndAngle

Agir sur le corps:

body.ApplyForce(vecteur, centre) // Pousser un objet

body.ApplyImpulse (vecteur, centre) // Frapper un objet

BOX2D
RAFRAICHIR LE MONDE

Les éléments du monde sont recalculés toutes les x périodes.

Définition d'une période en Javascript:

```
var periode = 1000 / 60; // 60 fps
window.requestAnimFrame = (function(){
        return  window.requestAnimationFrame       ||
                window.webkitRequestAnimationFrame ||
                window.mozRequestAnimationFrame    ||
                window.oRequestAnimationFrame      ||
                window.msRequestAnimationFrame     ||
                function(/* function */ callback, /* DOMElement */ element){
                  window.setTimeout(callback, periode);
                };
   })();
```

BOX2D
RAFRAICHIR LE MONDE (SUITE)

Rafraichir le monde en utilisant la période précédente:

```
(function update() {
        world.Step(
        1 / 60,     // frame-rate
                10,         // velocity iterations
                10          // position iterations
        );
        // world.ClearForces();
        requestAnimFrame(update);
})();
```

BOX2D
DEBUG DRAW

ATTENTION: BOX2D N'EST PAS FAIT POUR DESSINER MAIS SEULEMENT POUR CALCULER DES INTERACTIONS !!!!

Cela-dit, le mode debug permet de dessiner le monde dans un canvas (utile pour démarrer sans avoir à coder les fonctions de dessins html5).

Initialisation:

```
var debugDraw = new Box2D.Dynamics.b2DebugDraw();
debugDraw.SetSprite( $(« #canvas »)[0].getContext("2d") );
debugDraw.SetDrawScale(SCALE);
debugDraw.SetFillAlpha(0.3);
debugDraw.SetLineThickness(1.0);
world.SetDebugDraw(debugDraw);
```

BOX2D
DEBUG DRAW (SUITE)

Ajouter après le world.Step:

```
world.DrawDebugData();


(function update() {
        world.Step(
                1 / 60,  // frame-rate
                10,      // velocity iterations
                10       // position iterations
        );
        world.DrawDebugData();
        // world.ClearForces();
        requestAnimFrame(update);
})();
```

BOX2D
POUR ALLER PLUS LOIN

http://code.google.com/p/box2dweb/

http://www.box2dflash.org/docs/2.1a/reference/

http://box2d.org/documentation/

CONCLUSION

Nous avons vu:

- **Comment utiliser Yeoman pour s'éviter les tâches répétitives et ainsi gagner du temps pour notre développement**
- **La libraire Javascript Backbone pour structurer et organiser notre code Javascript**
- **La librairie Box2D pour animer notre interface utilisateur avec un moteur physique en Javascript**

Y'a plus qu'à…

DOCUMENTATIONS

Yeoman

http://yeoman.io/index.html

https://github.com/yeoman/yeoman

Backbone.js

http://backbonejs.org/

http://underscorejs.org/

Box2D

http://code.google.com/p/box2dweb/

http://www.box2dflash.org/docs/2.1a/reference/

http://box2d.org/documentation/

A VOIR EGALEMENT

Sublime Text 2

http://www.sublimetext.com/

Git

http://git-scm.com/

http://git-scm.com/book/fr **(documentation en français)**

Github

https://github.com/

NOUS CONTACTER

Hugo Mallet

hugo.mallet@53js.org

@hugomallet

François Michaudon

francois.michaudon@53js.org

@_big26