

# Développement d'une interface utilisateur web

## Projet n°6 du parcours python d'openclassrooms

Bérenger Ossété Gombé

23 juillet 2022

# Sommaire

- 1 Introduction
- 2 Préparation du projet
- 3 Démonstration
- 4 Mise en place du *front-end*
  - HTML
  - CSS
  - Javascript
- 5 Conclusion

# Présentation

## Je me présente

- Béranger Ossété Gombé, 27 ans
- Mon parcours informatique
  - Baccalauréat scientifique (2013)
  - Master 1 en informatique à l'UFC, UFR-ST (2016-2017)
    - Spécialité génie logiciel.
  - Reconversion vers du développement web chez Openclassrooms (janvier 2022).

# Formation

## Contexte

Projet n°6 du parcours Développeur d'application python chez openclassrooms.

## Objectifs

- Développer la partie *front-end* d'une application web.
  - HTML
  - CSS
  - Javascript
- Interagir avec une API REST.

# Contexte fictif

## JustStreamIt

- Association de cinéphiles.
- Système de *newsletters*.
- Migration vers un site web.
- Équipe
  - Camille, la CEO de JustStreamIt.
  - Éric, notre contact technique.

## Notre travail

→ Nous sommes volontaire pour développer le *front-end* du nouveau site. Nous utiliserons leur API existante.

# Préparation du projet

## Étapes

- Collecte des exigences du client.
- Tests et documentation.
- Normes et qualité.

# Exigences fonctionnelles

## Une liste des exigences fonctionnelles

- Conception **conforme à la maquette** comprenant :
  - une barre de navigation,
  - une vignette présentant le meilleur film,
  - une catégorie des films les mieux notés,
  - trois catégories aux choix comprenant sept films.
- **Informations d'un film** *via* une fenêtre modale contenant :
  - l'image, le titre, le genre, la date de sortie,
  - le classement, le score IMDB,
  - le réalisateur, les acteurs,
  - la durée, la pays d'origine, le résultat au *box office* et
  - le résumé du film.

# Exigences non-fonctionnelles

## Une liste des exigences non-fonctionnelles

- **Portabilité** vers les navigateurs Chrome, Safari et Firefox.
- Utilisation de l'API **OCMovies-API** et accès *via* des requêtes AJAX.
- **Mise-à-jour automatique** des données.
- Utilisation de Javascript et de CSS *sans **framework** ni bibliothèque.*



# Tester le *front-end*

## Pourquoi tester le *front-end* ?

- Vérifier que le produit est conforme aux exigences.
- Mettre en place des vérifications pour accompagner une potentielle future mise à l'échelle.
- Documenter les bugs et éviter autant que possible les régressions.

# Tester le *front-end*

## Utilisation de Jasmine

Choix d'utiliser Jasmine car<sup>1</sup> :

- Possède peu de dépendances.
- Est un projet très stable.
- Une utilisation simple est possible.
- → Prend peu de place au sein du projet.

---

1. Voir

<https://npmcompare.com/compare/chai,jasmine,jest,karma,mocha>

# Documenter le *front-end*

## Pourquoi documenter le *front-end* ?

- Le projet peut être repris à l'avenir par quelqu'un d'autre.
- Le projet peut grossir et se complexifier.

## Utilisation de JSDoc

Choix d'utiliser JSDoc car :

- facile et rapide à installer,
- facile à utiliser,
- génère une documentation claire et concise.

# Normes et qualité

## Vérifier le code HTML

- Permet de détecter les erreurs dans le code HTML.
- Permet de garantir la conformité du code avec les recommandations de la W3C<sup>2</sup>.
- Disponible sous forme d'extension pour navigateurs<sup>3</sup>.

---

2. World Wide Web Consortium

3. Disponible au moins pour Chrome et Firefox.

# Normes et qualité








Type	Line	Column	HTML errors and warnings
 Info	0	0	0 errors / 23 warnings
 Warning	29	168	Warning: missing </div> before <a>
 Warning	32	4	Warning: discarding unexpected </a>
 Warning	32	14	Warning: discarding unexpected </div>
 Warning	58	184	Warning: missing </span> before <div>
 Warning	58	162	Warning: missing </span> before <div>
 Warning	58	363	Warning: inserting implicit <span>

Figure – HTML Validator : une extension firefox pour vérifier le code HTML

# Normes et qualité

## Vérifier le code Javascript

Utilisation de ESLint<sup>2</sup>, un *linter* pour Javascript.

- Permet de localiser les erreurs javascript.
- Permet également de localiser les erreurs de style.
- Permet de générer un rapport.

---

2. <https://eslint.org/>

# Normes et qualité

## ESLint Report

1 problem (1 error, 0 warnings) - Generated on Sun Jun 12 2022 17:15:30 GMT+0200 (heure d'été d'Europe centrale)

[-] /home/bog/Dev/Tutorials/JSESLint/app.js

1 problem (1 error, 0 warnings)

2:7 **Error** 'x' is assigned a value but never used.

[no-unused-vars](#)

Figure – Exemple de rapport généré par ESLint.

# Démonstration du *front-end*

## Fonctionnalités clefs

- Le menu principal.
- Le meilleur film.
- La catégorie des films les mieux notés.
- Les 3 catégories au choix.
- L'aperçu d'un film *via* une fenêtre modale.

## Fonctionnalités additionnelles

- Adaptation au mobile.



# Mise en place du *front-end*

## Les aspects du développement *front-end*

- La structure avec HTML.
- Le style avec CSS.
- Le comportement avec le Javascript.

# Structure HTML du projet

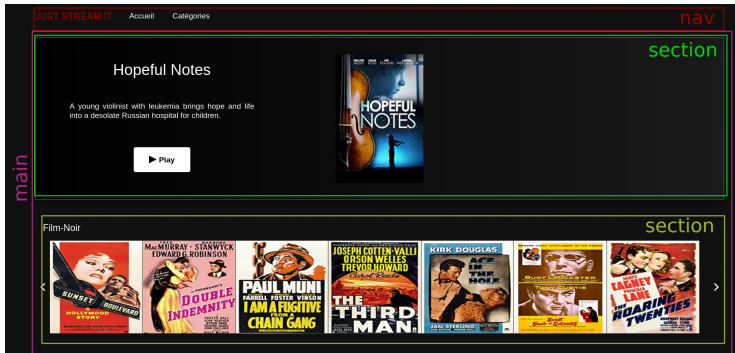


Figure – Éléments HTML de la page d'accueil

# Un CSS maintenable et *scalable* avec SASS

## Organisation

- Séparation du style sur plusieurs fichiers
- Application du *pattern* 7-1

```
@import 'base/reset';  
@import 'abstracts/mobile';  
@import 'themes/theme';  
@import 'layout/nav';  
@import 'pages/home';  
@import 'components/buttons';  
@import 'components/best_movie';  
@import 'components/movie_info';
```

# Un CSS *responsive*

aze

- Approche *Mobile first* lors du développement.
- Utilisation d'une mixin SCSS pour gérer les *media queries*.

```
@mixin for-mobile {  
  @media screen and (max-width: 599px) {  
    @content  
  }  
}
```

## Un CSS *responsive* : exemple

```
@include for-mobile {  
    position: fixed;  
    margin: 0;  
    top: 0;  
    left: 0;  
    width: 100%;  
    min-width: 0;  
    height: 100%;  
    max-height: 100%;  
}
```

# Un CSS organisé avec la méthode BEM

## La méthode BEM

- Permet d'organiser le CSS.
- Extensible au SCSS.
- Signifie *Block* - *Element* - *Modifier*.
- Propose un formatage dans le choix des noms de classe CSS d'un projet.

## Un exemple de formatage

`nom_bloc __ nom_element - - nom_modifier`

# Le BEM appliqué au SASS

## En SASS

```
.best_movie {  
  &__title {  
    &--blue {}  
  }  
}
```

## En CSS

```
.best_movie {}  
.best_movie best_movie__title {}  
.best_movie best_movie__title best_movie__title--blue {}
```

# Structure du code Javascript

src/js

- app.js
- movie.js
- carrousel.js
- category.js
- html.js
- api.js



# Utilisation de NPM

## Node Package Manager

- N'est pas obligatoire pour le bon fonctionnement du projet.
- Permet de gérer les dépendances vers des bibliothèques javascript.
- Nous l'utilisons uniquement pour le test, le linter et autres outils indépendants du domaine métier.

# Point d'entrée de l'application

## Fonctionnalité

- Initialise les différentes catégories.

```
async initAllCategories() {  
  await Promise.all([  
    this.initCategory('Biographies', 'Biography'),  
    this.initCategory('Film-Noir', 'Film-Noir'),  
    this.initCategory('Historique', 'History')  
  ]);  
}
```

# Initialisation des catégories

```
async initCategory(name, genre) {  
  let cat = new category.Category(name)  
  
  const fetcher = new api.MovieFetcher();  
  const movies = await fetcher.findByGenre(genre);  
  
  for (let movie of movies) {  
    cat.addMovie(movie);  
  }  
  
  this._categories.push(cat);  
}
```

# Représentation des films

## Champs d'un objet *Movie*

- title
- image\_url
- genres
- release\_date
- rated
- imdb\_score
- directors
- actors
- duration
- countries
- box\_office
- description
- long\_description

# Représentation des catégories

## Constructeur de l'objet *Category*

```
constructor(name) {  
  this._name = name;  
  this._movies = [];  
  this._carrousel = new carrousel.Carrousel();  
  this._root_element = null;  
}
```

# Mise en place du carrousel

## Représentation d'un carrousel

- Utilisation de la classe *Carousel*.
- On peut ajouter un film au carrousel.
- On peut déplacer le carrousel vers la gauche ou la droite.

# Animation du carrousel

## La classe *CarrouselAnimation*

```
slide(speed=512, count=1) {  
  const reference = this._list[0];  
  let start = reference.offsetLeft;  
  html.animationLoop(function (dt) {  
    for(let movie of this._list) {  
      html.moveElement(movie, speed * dt, 0);  
    }  
  
    return Math.abs(reference.offsetLeft - start)  
    < this._movie_width * count;  
  }).bind(this));  
}
```

# Manipulation du code HTML

## Fonctionnalités de *html.js*

```
export default {  
  CategoryHTMLBuilder: CategoryHTMLBuilder,  
  BestMovieHTMLBuilder: BestMovieHTMLBuilder,  
  Modal: Modal,  
  moveElement: moveElement,  
  setElementPosition: setElementPosition,  
  animationLoop: animationLoop  
};
```



# Génération du code HTML

## Fonctionnalités de *html.js*

```
build(root_id) {  
  let root = document.getElementById(root_id);  
  let section = this.buildSection();  
  root.appendChild(section);  
  
  return section;  
}
```

# Génération du code HTML

## Fonctionnalités de *html.js*

```
buildSection() {  
  let section = document.createElement('section');  
  section.classList.add('category');  
  let h2 = document.createElement('h2');  
  h2.innerText = this._title;  
  section.appendChild(h2);  
  section.appendChild(this.buildCategoryContent());  
  return section;  
}
```

# Génération du code HTML

## Fonctionnalités de *html.js*

```
buildCategoryContent() {  
  let content = document.createElement('div');  
  content.classList.add('category__content');  
  
  let left_arrow = this.buildArrow('left');  
  left_arrow.addEventListener('click', this._left_callback);  
  content.appendChild(left_arrow);  
  
  let ul = document.createElement('ul');  
  
  for (let movie of this._movies) {  
    ul.appendChild(this.buildMovie(movie));  
  }  
  
  // ...  
}
```

# Récupération des informations *via* l'API

## La classe *MovieFetcher*

- Permet de faire une recherche par genre, par ID ou par score IMDB.
- Permet de créer un objet de type *Movie* à partir d'un format JSON.

# Recherche par genre

```
async findByGenre(name, count=7, url = null, results = []) {  
  if (url === null) {  
    url = `http://localhost:8000/api/v1/titles/?genre=${name}&sort_by=-imdb_score`;  
  }  
  
  const data = await fetch(url);  
  const json_movie = await data.json();  
  
  for (let result of json_movie.results) {  
    const movie_result = await this.findByID(result.id);  
    results.push(this.createMovieFromJSON(movie_result));  
  }  
  
  if (results.length < count) {  
    await this.findByGenre(name, count, json_movie.next, results);  
  }  
  
  return results.slice(0, count);  
}
```

## Création d'un objet *Movie*

```
createMovieFromJSON(json_movie) {  
    return new movie.Movie(json_movie.title,  
    new URL(json_movie.image_url),  
    json_movie.genres,  
    new Date(json_movie.date_published),  
    json_movie.rated,  
    json_movie.imdb_score,  
    json_movie.directors,  
    json_movie.actors,  
    json_movie.duration,  
    json_movie.countries,  
    json_movie.reviews_from_critics,  
    json_movie.description,  
    json_movie.long_description);  
}
```

# Gestion de la modale

## La classe *Modal*

- Permet d'afficher ou de cacher la fenêtre modale.
- Permet de mettre à jour les informations montrées.

# HTML de la modale

## Entête de la modale

```
<div class="movie_info">
  <div class="movie_info__header">
    <span class="movie_info__header__title">Titre</span>
    <span class="movie_info__header__close">&times;</span>
  </div>
```

## Contenu de la modale

```
<div class="movie_info__content">
  </img>

  <table>
    <tr>
      <th>Genres</th>
      <td id="movie_info__content__genres"><ul></ul></td>
    </tr>
    <tr>
      <th>Sortie</th>
      <td id="movie_info__content__release_date"></td>
    </tr>
  <!-- ... -->
```



# Visibilité de la modale

## Montrer

```
show(movie) {  
  if (this._is_visible === false) {  
    this.update(movie);  
    this._is_visible = true;  
    this._root.style.display = 'inline-block';  
  }  
}
```

## Cacher

```
hide() {  
  if (this._is_visible === true) {  
    this._is_visible = false;  
    this._root.style.display = 'none';  
  }  
}
```

# Mise à jour de la modale

## Récupération des éléments du DOM de la modale (extrait)

```
update(movie) {  
  const prefix = '#movie_info__content__';  
  
  const elements = {  
    'title': document.querySelector('.movie_info__header__title'),  
    'img': document.querySelector('.movie_info__content img'),  
    'genres': document.querySelector(prefix + 'genres ul'),  
    'release': document.querySelector(prefix + 'release_date'),  
    'rated': document.querySelector(prefix + 'rated'),  
    'imdb': document.querySelector(prefix + 'imdb'),  
    'directors': document.querySelector(prefix + 'directors ul'),  
    'actors': document.querySelector(prefix + 'actors ul'),  
    'duration': document.querySelector(prefix + 'duration'),  
    'countries': document.querySelector(prefix + 'countries ul'),  
    'box-office': document.querySelector(prefix + 'box_office'),  
    'summary': document.querySelector(prefix + 'summary')  
  };  
  
  // ...  
}
```

# Mise à jour de la modale

## Modification du HTML (extrait)

```
elements['title'].innerText = movie.title;  
elements['img'].setAttribute('src', movie.image_url);  
  
this.buildList(elements['genres'], movie.genres);  
elements['release'].innerText = movie.release_date.toDateString();  
elements['rated'].innerText = movie.rated;  
elements['imdb'].innerText = movie.imdb_score;  
this.buildList(elements['directors'], movie.directors);  
this.buildList(elements['actors'], movie.actors);  
  
// ...
```

# Conclusion

## Pour conclure

- Pour aller plus loin.
- Travail effectué.

## Pour aller plus loin

- Mise en place de plusieurs thèmes (light/dark theme).
- Utilisation d'un serveur d'intégration continue comme Jenkins ou Gitlab CI.
-

# Travail effectué

- Mise en place d'un environnement de développement *front-end*.
- Utilisation de SASS pour mieux organiser les feuilles de styles CSS.
- Mise en application des fondamentaux du javascript ES6.
- Interaction avec une API REST avec l'API fetch.

# Merci pour votre attention

- 1 Introduction
- 2 Préparation du projet
- 3 Démonstration
- 4 Mise en place du *front-end*
  - HTML
  - CSS
  - Javascript
- 5 Conclusion