JQuery

Matthieu Nicolas
Licence Pro CIASIE
Slides par Christophe Bouthier

But de jQuery

- Accès aux éléments du DOM
- Modification des éléments (ajout classes, ajout éléments)
- Gestions des events
- Effets et animations
- Code utilitaire (each…)

Plan

- Principes généraux
- Méthode ‘jquery’
- Sélecteurs
- Manipulation du DOM
- Events
- Effets

Principes généraux

JQuery #01

Historique

- 2006 - John Resig
- Yet Another Javascript Library
  - Prototype, Scriptaculous, Dojo, MooTools, YUI…
- Oui mais…
  - Ne cherche pas à modifier les paradigmes de JS
    - Au contraire, s’insère en plein dedans
  - Evite d’écrire du code « boilerplate »
  - Très bon support cross-browser

Principe 1 : API « fluent »

- chaque appel retourne une valeur
  - le + souvent la valeur modifiée par l'appel
  - possible d'avoir la valeur 'avant' l'appel
- possibilité de chainer les appels
  - e.doThis().doThat()

Principe 2 : Itérations automatiques

- Retour de valeurs :
  - soit élément unique
  - soit tableau d'éléments
- Méthodes s'appliquent automatiquement sur les tableaux
  - itération sur les éléments du tableau
  - appel de la méthode sur chaque élément

Principe 3 : Polymorphisme

- Une même méthode peut faire plusieurs choses
  - en fonction du nombre de paramètre
  - en fonction du type des paramètres
  - mix-up des deux
- Notamment :
  - sans argument : getter
  - avec argument : setter

Principe 4 : objet « options »

- Pas d'arguments nommés en JS
  - seule la place de l'argument compte
- Plus lisible si l'argument a un nom
  - indépendant de la place
- Idée : passer un 'hash' (un objet)
  - chaque propriété = un argument
- -> 1 seul arg : Objet « options »

Utilisation

- Appel local
  - méthode sur un élément du DOM
  - effet « local » (sur l'élément ou ses fils)
  - méthode ajoutées par JQuery
- méthode « jQuery »
  - aliasée sur '$()'
  - appel et effet « global »
    - tout ce qui n'a pas un contexte local

Méthode ‘jQuery’

JQuery #01

Méthode ‘jQuery’

- Aliasée à $()
- 4 actions (suivant params) :
  - $(callback)
  - $(dom_element)
  - $(« css selector »)
  - $(« html »)

$(callback)

- Identique à $(document).ready(callback)
- Appelle le callback une fois que le dom est ‘prêt’ :
  - tous les éléments du DOM sont présents, même s’ils n’ont pas tous été téléchargés (images…)
- Permet d’éviter des « race conditions »
  - pas d’ordre garantie de chargement
- Bonne pratique

Exemple

```
1   $(function·(){
2   ····//·Ce·code·sera·exécuté·une·fois·le·DOM·prêt
3   })
4
```

Objets JQuery

- méthodes sur éléments du DOM
  - méthodes ajoutées par jQuery
- nouveau objet avec ces méthodes
  - objets « jQuery »
  - encapsule les objets DOM
  - ajoute des méthodes
- Convention :
  - nom de variable commençant par $

$(dom_element)

- Encapsule un élément du DOM
  - retourne un objet « jQuery »
- Utilisable sur tout élément du DOM
  - $(document)
  - $(this)
- A utiliser à chaque fois qu’une valeur n’est pas retournée par une méthode de jQuery

$(« css_selector »)

- Retourne l'élément ou la liste d'éléments qui « matche » le sélecteur
- Ancêtre de « querySelector() » et « querySelectorAll() » d'ECMAScript 5
  - sauf que gère automatiquement 1 ou plusieurs
- Pseudo-classes en plus de CSS

$(« html »)

- Parse le code HTML passé en paramètre
- Crée l’élément DOM correspondant
- Encapsule l’élément dans un objet jQuery
- Retourne l’objet jQuery

Exemple

```html
<html>
    <body>
        <div id="menu"></div>

        <script type="text/javascript" src="lib/jquery-3.3.1/jquery-3.3.1.js"></script>
        <script type="text/javascript">
            // Déclenche ce code une fois le DOM prêt
            $(function () {
                const $menu = $("#menu")    // Récupère l'élément avec id="menu"

                // Génère un header
                const $header = $("<h1>My Application</h1>")

                // Ajoute l'header au menu
                $menu.append($header)
            })
        </script>

    </body>
</html>
```

String interpolation

- 3ème méthode pour déclarer une chaîne de caractère

```
let a = 42
let str = `la valeur est ${a}` // "la valeur est 42"

str = `On peut
en plus déclarer
des chaînes sur
plusieurs lignes`
```

- Attention ! Utilise des « backquote »

Sélecteurs

JQuery #01

But

- Pouvoir accéder à un élément à partir d'un sélecteur CSS
- Premier but de jQuery
  - d'où le nom
- Retourne un objet jQuery, ou un ensemble d'objets jQuery

Comme CSS

- $(« div »)
- $(« .class »)
- $(« #id »)
- $(« #id > tag »)
- $(« a[href^=‘mailto:’] »)
- $(« tr:nth-child(1) »)

Custom selectors

- Négation
  - $(« li:not(.class) »)
- odd, even
  - $(« tr:nth-child(odd) »)
- Accès à un élément parmi un set
  - $(« div:eq(1) »)
- Texte
  - $(« p:contains(mytexte) »)

Sets d’objets

- Une fois que l’on a un set/élément
  - soit action directement dessus
  - soit raffinement du set/de l’élément
    - filtrage
    - navigation
- Méthodes
  - sur set ou element
  - chainable

.filter()

- $(« div »).filter(callback)
- Permet de construire un set
- Appelle le callback pour chaque élément de l’ensemble
- Regarde la valeur de retour
  - true -> l’élément est gardé dans le set
  - false -> l’élément est rejeté du set

Navigation

- .next()
- .nextAll()
- .prev()
- .prevAll()
- .siblings()
- .parent()
- .children()

- .end()

revient à l'élément précédent

- .find(« sel »)

refait une recherche

- .eq(index)

accès à l'élément index du set

Exemple

```
$("div.myclass")·····//·Retourne·tous·les·div·ayant·la·classe·"myclass"
····.parent()·····//·Pour·chaque·div,·prend·son·parent
····.find("p:eq(1)")····//·À·partir·du·parent,·prend·le·second·<p>
····.hide()·····//·Le·cache
····.end()·····//·Revient·au·parent
····.show()·····//·L'affiche
```

Manipulation du DOM

JQuery #01

Classes CSS

- addClass(), removeClass()
  - gère directement l’ajout et le retrait
  - même si plusieurs classes sur l’élément
- .toggleClass()
  - ajoute si la classe n’y est pas, retire si elle y est
  - teste automatiquement la présence de la classe

Accès CSS direct

- .css(« property »)
  - getter
  - retourne la valeur de la propriété
- .css(« property », « value »)
  - setter
  - met la propriété à value
- S’applique sur un élément

Accès multiples

- .css([« property_1 », « property_2 »])
  - getter multiple
  - retourne un objet
- .css({prop1: value1, prop2: value1})
  - setter multiple
  - permet de mettre plusieurs propriétés en un seul appel

Subtilités

- Attentions aux unités !!!
  - valeur de retour: **String**
    - « 10px »
      - parseFloat() ne parse que la partie numérique
  - valeur d’entrée: **String**
    - ne pas oublier de concaténer l’unité !
- Gère automatiquement les préfixes CSS
  - -webkit-, -moz-, …

Accès aux attributs

- .attr()
- Utilisation comme .css()
  - getter, setter
  - accès simple, accès multiples
- .attr(attributeName, callback) :
  - fonction appelée pour chaque valeur
  - valeur = valeur de retour de la fonction

Insertion d’éléments

- .insertBefore()
  - dehors et avant l’élément
- .prependTo()
  - dedans et avant l’élément
- .prependAfter()
  - dedans et après l’élément
- .insertAfter()
  - dehors et après l’élément

Wrapping

- .wrapAll(« <tag></tag> »)
  - entoure tout un ensemble d'éléments avec un seul tag (passé en paramètre)
- .wrap(« <tag></tag> »)
  - entoure chaque élément avec le tag passé en paramètre

Events

JQuery #01

.on() / .off()

- méthode générique
- s’applique sur un élément
- prend deux paramètres :
  - event-name : le nom de l’event à capturer (click…)
  - callback : le callback à appeler
- toujours en bubbling

Utilisation

- En fait, ajoute un listener
  - possibilité d'avoir plusieurs listeners sur le même élément pour le même event
  - appelé dans l'ordre d'enregistrement
- Callback
  - event passé en premier param
  - this = élément
    - $(this) pour récupérer un objet jQuery

Event namespaces

- .off(<event-name>) retire TOUS les listeners
- Création de namespace : .name après le nom de l’event
- .on(« click.monNom »)
- .off(« click.monNom »)
  - ne retire que le listener mis avec .on(« click.monNom »)

Subtilités

- Prévu pour la délégation
  - .on(<event-name>, <selector>, <callback>)
  - ne se déclenche que si event.target.is(<selector>)
- alias avec le nom de l'event
  - .click(callback)
- Simulation d'event
  - .trigger(<event-name>)
  - alias, comme .on()
    - .click() -> simule un event click

Effets

JQuery #01

Effets

- Animations « pré-cablées »
- S’applique aussi bien à 1 élément qu’à un ensemble d’éléments
- Pas d’argument -> effet immédiat
- Argument **String** -> animation
  - « slow » (600ms), « fast » (200ms), « autre » (400ms)
- Argument **entier** (en milliseconds)

.hide() / .show()

- Afficher / cacher
  - basé sur la propriété « display »
  - .hide() sauve la propriété de « display » avant de la mettre à « none »
    - .show() remet l’ancienne valeur
- .toggle() pour passer automatiquement de l’un à l’autre
  - test de la propriété « display » automatique

Autres effets

- .fadeIn() / .fadeOut()
  - opacity
- .slideDown() / .slideUp()
  - height
- Switch automatique
  - fadeToggle()
  - slideToggle()

Résumé

- Méthode ‘jquery’
- Sélecteurs
- Manipulation du DOM
- Events
- Effets

Projet

JQuery #01

Planning

- Séance d’aujourd’hui
  - Présentation du projet + affectation des sujets
- Séance du 20 novembre
  - Cours + TP de prévu
  - Ne comptez pas trop dessus pour avancer sur le projet
- Séance du 27 novembre
  - Soutenances

Principe

- Par groupe de 4

- Chaque groupe choisit un thème différent d'une liste

- Développe une appli sur ce thème

- Donnera lieu à une démo de l'appli (~5min) et à une présentation de la techno (~10min)

Objectifs & attentes

- Rechercher
  - Se documenter, filtrer les informations
- Etudier
  - Comprendre le fonctionnement de la techno
  - Comment se place-t-elle par rapport aux concurrentes ?
- Communiquer
  - Présenter la techno, illustrer son fonctionnement

Thèmes

- Electron
- Cordova
- WebSocket
- WebRTC
- WebSpeechAPI
- Cypress.io
- Howler.JS
- Chart.js

TP

JQuery #01

Conception d'une médiathèque

- Veut une appli pour gérer ma collection
- Plusieurs types de médias
  - Albums de musique (**Album**)
  - Films (**Movie**)
  - Jeux (**Game**)
  - …

Fonctionnalités principales

- Afficher une liste de médias
- Ajouter/supprimer un média
- Filtrer l’affichage par type de média
  - Afficher seulement les films par exemple

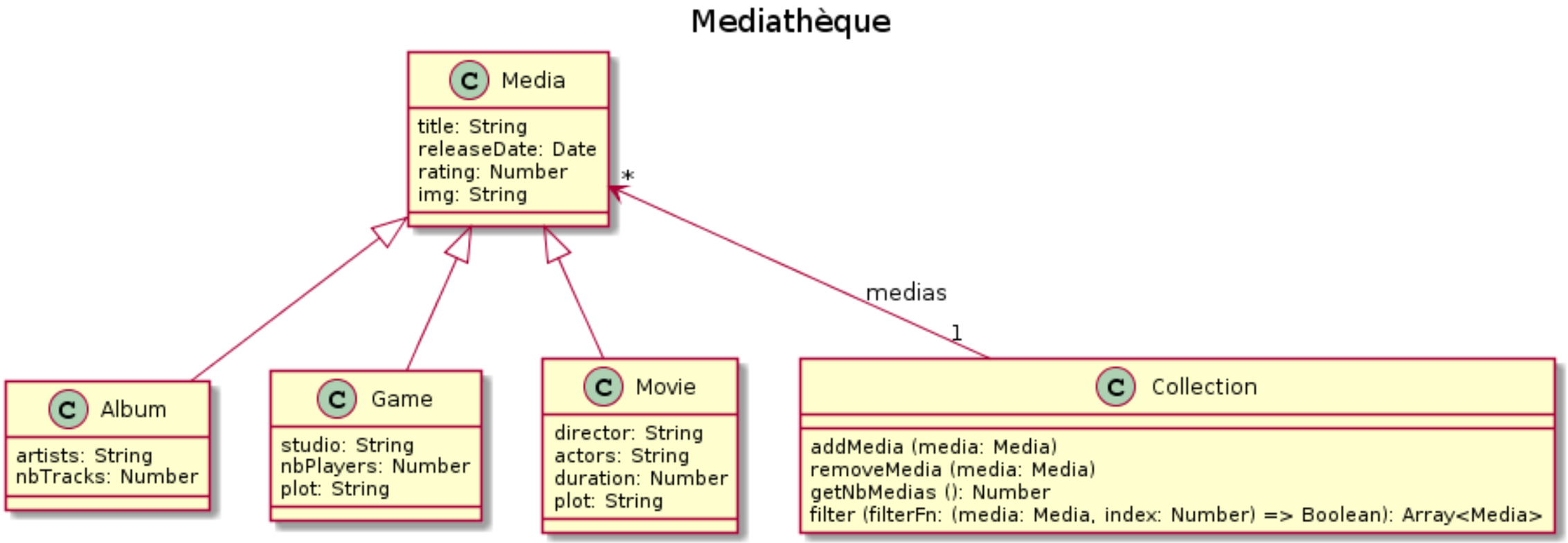
Fonctionnalités bonus

- Trier la collection
  - Par titre, date de sortie…
- Enregistrer la collection en local
  - Pour la récupérer à la prochaine session
  - Voir l'API **localStorage**
- Importer un média à l'aide d'une API
  - Créer un film à partir de sa fiche IMDB par ex.
  - Voir http://www.omdbapi.com/

Objectifs

- Écrire une **application** web
  - Faire le lien entre les objets qu'on manipule et l'interface
- **Manipuler** le DOM en JS
  - Ajouter dynamiquement des éléments, lire la valeur d'éléments
- **Réagir** aux actions de l'utilisateur
  - Gestion des évènements

Conception



- Reste à concevoir le “main” qui lie notre **Collection** au DOM

Consignes

- Pas de frameworks JS/CSS lourd
  - Pas d'Angular, React, Vue…
  - Vous pouvez utiliser Foundation, Bootstrap…

- Utilisez le design pattern **Module**
  - Seulement le module **MyMediatek** doit être global

Rappel Module

```
const Module = (function () {

    const obj = {}
    obj.prop = 1
    obj.method = function () { return 2 }

    return obj
})()

Module.prop     // 1

Module.method() // 2

```

Module MyMediatek

- On va regrouper l'ensemble des constructeurs dans le module **MyMediatek**
  - *MyMediatek.models* va regrouper les constructeurs des différents types de données
- On pourra ajouter d'autres propriétés à **MyMediatek**

```
1   MyMediatek.models.Media·····//·Le·constructeur·de·Media
2   MyMediatek.models.Album·····//·Le·constructeur·de·Album
3   MyMediatek.models.Game·····//·Le·constructeur·de·Game
4   MyMediatek.models.Movie·····//·Le·constructeur·de·Movie
5
6   MyMediatek.App·····//·Le·constructeur·du·main
7
```

Cheminement conseillé

1. Implémenter les modèles de données
2. Afficher un média en dur (pur HTML)
3. Créer et afficher le média dynamiquement
4. Ajout du formulaire pour créer un nouveau média
5. …

Évaluation

- Le TP sera évalué
- Date limite: 26/10, 23:59
- Seront pris en compte
  - Les fonctionnalités implémentées
  - L’utilisation de git
  - La qualité du code
  - L’interface

Let’s go

https://classroom.github.com/a/mIYzKq_y
(lien dispo sur Arche)