

Bachelier en Informatique de Gestion

Web : principes de base Projet de Développement Web

Enseignement supérieur économique de type court

Code FWB : 7534 29 U32 D1, 7534 30 U32 D3

Code ISFCE : 4IWPB, 4IPW3



Table des matières

Généralités

- 01. Introduction au web
- 03. Outils
- 05. Format XML
- 06. Format JSON

Front-End

- 12. Structure HTML
- 13. Formulaire HTML
- 14. Mise en forme CSS
- 15. Adaptabilité
- 17. Javascript
- 18. Bibliothèque jQuery
- 19. Composant Vue.js

Back-End

- 21. Middleware PHP
- 22. Traitement du formulaire
- 23. Architecture MVC
- 24. Données SQL
- 25. Données NoSQL
- 27. Requête asynchrone





18. Framework jQuery

Installation

Syntaxe

Selector

Modification d'un
contenu

Modification des
propriétés

Données



= BUROTIX ()



Prologue



= BUROTIX ()

Propos liminaire : Pourquoi JQuery ?

- JQuery est la bibliothèque JavaScript utilisée par 75% des sites web au niveau mondial.
 - Bootstrap : 20%
 - React : 4%
 - Vue.js : 1%
 - Angular : <1%
- Source
 - W3TECHS, septembre 2024

Principe

- "Write less, do more"
- Bibliothèque créée au dessus de HTML, CSS, JavaScript et AJAX
 - Manipuler des éléments HTML mis en forme en CSS
 - Utiliser des instructions donnant simplement accès aux immenses possibilités de JavaScript et d'AJAX.
- Avantages de jQuery sur JavaScript
 - Syntaxe plus robuste
 - Moins d'erreur de syntaxe
 - Syntaxe moins verbeuse
 - Moins de lignes de code
 - Intemporel
 - Pas de problème de version de Javascript



Références

- OpenClassRooms
 - <https://openclassrooms.com/fr/courses/1631636-simplifiez-vos-developpements-javascript-avec-jquery/>
- La documentation officielle de jQuery
 - <http://docs.jquery.com/>
- Le forum de discussion consacré à jQuery
 - <https://forum.jquery.com/>



Prérequis

- HTML
 - Balise
 - Attribut
 - Block vs inline
- CSS
 - Sélecteur CSS
- Javascript
 - DOM
- POO
 - Fonction
 - Méthode
 - Objet





Syntaxe de base



= BUROTIX ()

Installation

- Local ou remote
 - 2025 : Version stable 3.7.1
- Intégrer le code suivant dans ses propres pages web (sous balise **<head>**).
- Cf <https://releases.jquery.com/jquery/>
- Il existe aussi du code similaire fourni par Google.

<script

src="https://code.jquery.com/jquery-3.6.3.min.js"

integrity="sha256-pvPw+upLPUjgMXy0G+800xUf+/Im1MZjXxxgOcBQBXU="

crossorigin="anonymous"></script>>



= BUROTIX 0

Emplacement du code jQuery

- Principe : On met le code JS/jQuery là on est sûr que le DOM est complet, càd après le chargement de la page
 - Soit sous `<head><script>...`
 - Recommandé
 - Soit en fin de `<body>`
 - De préférence comme fichier séparé
`<script src="mon-script.js"></script>`
 - Hint: attribut `defer`
 - pour différer l'exécution du Javascript
`<script defer src="mon-script.js"></script>`



Architecture du code jQuery

- Déclaration jQuery

```
<script src="https://code.jquery.com/jquery-3.4.1.js" ...
```

- Fonctions

```
<script>  
    function ma_fonction()  
    {  
        ///  
    }  
</script>
```

- Fonction "DOM ready", à exécuter juste après le chargement de la page

```
<script>  
    $(function()  
    {  
        ///  
    });  
</script>
```



DOM-ready

- Code exécuté après le chargement du document
 - en pratique tout votre code
- Trois formules équivalentes !
- Emplacement libre
 - Dans **<head>** de préférence ...

```
jQuery(document).ready(function() {  
    // Ici, DOM entièrement défini  
});
```

```
$(document).ready(function() {  
    // Ici, DOM entièrement défini  
});
```

```
$( function() {  
    // Ici, DOM entièrement défini  
});
```



Première commande jQuery : exo01

```
$("#exo1601").html("Hello World !");
```

- `$("#exo1601")` : sélecteur
- `.` : lien entre sélecteur et action
- `.html(...)` : méthode appliquée
aux éléments sélectionnés
- `"Hello World !"` : argument de la méthode
- `;` : fin de commande jQuery





Commandes essentielles



= BUROTIX ()

Créer un objet : JSON

- Créer un objet de toute pièce

```
var obj = {  
  prop1: 'prop1Value',  
  prop2: 'prop2Value',  
  child: {  
    childProp1: 'childProp1Value'  
  }  
}
```

- Exemple

```
var my_image_position = {  
  top : 100,  
  left : 100,  
};
```



Lire un objet : plusieurs façons

1. `console.log(my_image_position)`

- Debugging purpose

2. `alert(JSON.stringify(my_image_position)) ;`

- Debugging purpose

3. `$.each(my_image_position, function(k, v) {
 alert(k + ":" + v) ;
}) ;`

- Process purpose





Selector



= BUROTIX ()

Sélection d'éléments

`$ ('ul.bleu')`

- Éléments `` de classe `bleu`

`$ ('ul li[class="pair"]')`

- Éléments ``, contenus dans un ``, avec attribut `class` de valeur `pair`

`$ ('li[class]')`

- Éléments `` avec attribut `class`

`$ ('[width="40"]')`

- Éléments ayant attribut `width` de valeur `40`

`$ ('*')`

- Tous les éléments du document



Sélection d'éléments

- Bien réviser son CSS ! 😊
- Retour d'un sélecteur :
 - Toujours un "objet jQuery"
 - Similaire à un tableau
 - Propriété **length**
 - Index
 - \$ ('ul.bleu') [3]**
retourne la 4^{ème} balise **** de classe **bleu**.



Sélecteur et Méthode

\$ (selector) .method(params)

- **\$ (selector)** : sélectionner des éléments
- **méthod** : effectuer un traitement sur la sélection

▪ Exemple

\$ ('span#resultat') .html ('blabla') ;

- Écrire un message dans une balise **** d'identifiant **resultat**





Méthodes

Sélecteur et Méthode

■ Getters

- Lire une valeur
- Souvent **un** paramètre
- Aucune valeur en paramètre
- Retour : zéro, un ou plusieurs éléments

■ Setters

- Écrire une valeur
- Souvent **deux** paramètres
- Une valeur mise en paramètre

```
$('h2').css('font-size', '2em');
```

```
const fs = $('h2').css('font-size');
```



Méthodes : contenu des éléments

`.text()`

- Accéder à la valeur textuelle stockée dans l'élément.

`.html()`

- Accéder au code HTML stocké dans l'élément.

`$(this)`

- Accéder à l'élément sélectionné



Méthodes : insérer du contenu dans un élément

.append()

- Insérer du contenu à la fin de la sélection

.prepend()

- Insérer du contenu au début de la sélection

.before()

- Insérer du contenu avant la sélection

.after()

- Insérer du contenu après la sélection

.replaceWith()

- Remplacer la sélection (la sélection elle-même et non son contenu)



Méthodes : insérer des éléments dans le DOM

- **`eai.appendTo(cible)`**
 - Insérer un élément à la fin de la cible
 - **`eai`** : élément à insérer (sélecteur jQuery, nom d'élément, ...)
 - **`cible`** : élément dans lequel se fera l'insertion (sélecteur jQuery, nom d'élément, ...)
- **`eai.prependTo(cible)`**
 - Insérer un élément au début de la cible
- **`eai.insertBefore(cible)`**
 - Insérer un élément avant la cible
- **`eai.insertAfter(cible)`**
 - Insérer un élément après la cible
- **`.wrap()` , `.wrapAll()`**
 - Entourer un élément par un ou plusieurs autres éléments créés à la volée.
- **`.remove()`**
 - Supprimer les éléments sélectionnés

Méthode : manipuler les attributs (getter)

- **.attr()** : Capturer la valeur d'un attribut
- Ce getter ne renvoie qu'une seule valeur, celle du premier élément
- Exemple: Dans

```
<a href="http://api.jquery.com">  
    API jQuery  
</a>
```

```
<a href="http://docs.jquery.com">  
    Documentation jQuery  
</a>
```

... l'expression
`$('a').attr('href')`
renvoie seulement **http://api.jquery.com**



Méthode : manipuler les attributs (setter)

- Définir la valeur d'un attribut

```
$('#logo').attr('src', 'logo.gif');
```

- Définir la valeur d'une série d'attributs

```
$('#logo').attr( { src: 'logo.gif',  
                  alt: 'Logo de la société' } );
```

- Définir la valeur d'un attribut par fonction (algorithme)

```
$('a').attr('target', function() {  
    if(this.host == location.host) return '_self'  
    else return '_blank'  
});
```

- Supprimer un attribut

- `.removeAttr()`



Méthodes : CSS

.css ()

- Accéder aux propriétés CSS

.addClass ()

- Ajouter une classe

.removeClass ()

- Supprimer une classe

.hasClass ()

- Tester si l'élément est d'une certaine classe



Méthodes : effets spéciaux

.hide()

- Masquer un élément

.show()

- Afficher un élément

.fadeIn()

- Afficher un élément progressivement

.fadeOut()

- Masquer un élément progressivement

.toggle()

- Afficher ou masquer un élément

- Tous les effets spéciaux



Méthode : données : \$.data()

- Associer des données à un élément du DOM.
 - Données textuelles, complémentaires, quelconques.
 - Indépendant du contenu HTML.
- Setter : **\$.data(el, 'nom', json_data);**
 - **el** : nom de l'élément concerné, sans apostrophes
 - **'nom'** : nom dans lequel sera stockée la donnée
 - **json_data : {nom_don1:val_don1, nom_don2:val_don2, etc.}**
 - nom_don1, nom_don2, etc. : noms associés aux données
 - val_don1, val_don2, etc. : données quelconques
- Getter : **var uneVariable = \$.data(el, 'nom').nom_don;**
 - **uneVariable** : variable quelconque
 - **el** : nom de l'élément auquel une donnée a été associée
 - **'nom'** : nom dans lequel a été stockée la donnée
 - **nom_don** : nom de la donnée à retrouver.
- **\$.removeData()**
 - Supprimer les données associées à un élément



Méthode : données : \$.data() : exemple

```
// élément dans lequel on va stocker des données  
var my_div = $('div')[0];
```

```
// données en question  
var grandfather_json = {  
    name : "PeulaFenetre",  
    firstname : "Firmin",  
    age : 75,  
};
```

```
// on stocke les données dans l'élément  
$.data(my_div, 'grandfather', grandfather_json );
```

```
// on récupère les données de l'élément pour les afficher  
var val3 = $.data(div, 'grandfather').age;  
$('#sp3').text(val3);
```





Évènements : souris, clavier, focus, timer



= BUROTIX ()

Évènements

```
$(sel).on( "mge", function() {  
    // instructions jQuery  
    // gérant l'événement  
    // ! code de "callback" !  
    // càd exécuté plus tard  
})
```

- *sel* : sélecteur jQuery
- *mge* : méthode de gestion événementielle (cf slide suivant)



Évènements multiples

```
$(sel) .on ( {  
    mge1 : function () {  
        // instructions évén. 1  
    },  
    mge2 : function () {  
        // instructions évén. 2  
    },  
}) ;
```

- *sel* : sélecteur jQuery
- *mge1*, *mge2* : méthode de gestion événementielle



Évènements (*deprecated*)

```
$(sel) .mge( function() {  
    // instructions jQuery  
    // gérant l'événement  
    // ! code de "callback" !  
    // càd exécuté plus tard  
})
```

- *sel* : sélecteur jQuery
- *mge* : méthode de gestion événementielle (cf slide suivant)



Méthode événementielle : souris

.click()

- Clic gauche

.mouseover()

- Début de survol de l'élément

.mouseout()

- Arrêt de survol de l'élément

- Tous les événements souris



Méthode événementielle : clavier

- **.keydown ()**
 - Appui sur une touche du clavier
- **.keyup ()**
 - Relâchement d'une touche du clavier
- **.keypress ()**
 - Maintien d'une touche **textuelle** du clavier enfoncée
- **.change ()**
 - À chaque modification du contenu d'un **input**, **textarea** ou **select**.
- Tous les évènements clavier

Sur l'évènement lui-même :

- **event.which**
 - Déterminer la touche du clavier pressée
 - Valeur renvoyée : code de la touche
- **event.type**
 - Déterminer le type d'évènement
 - Valeur renvoyée
 - keydown
 - keypress
 - keyup



Méthode événementielle : focus

- **.focus()**
 - À la réception du focus
- **.blur()**
 - À la perte du focus
- **.focusin()** , **.focusout()** , **.resize()**
 - Autres méthodes potentiellement intéressantes



Évènement souris : exo 31

```


$("#target").on( "click", function() {
    // code exécuté au moment de l'évènement !
    var mip = {
        top : Math.floor(Math.random()*480),
        left : Math.floor(Math.random()*640)
    };
    $('#target').offset(mip);
    // fin du code évènementiel
});
```

Question :
Que fait ce
code jQuery ?

Évènement souris : exo 32

Fichier de départ : **exo32_image_bouton_start.html**

1. Positionner l'image en **top:100** et **left:milieu** de la page. Tuyaux :
 - **`$(window).width()`**
 - **`.offset()`**
2. Que l'image bouge de 10 pixels vers le bas à chaque clic sur le bouton. Tuyaux :
 - **`e.which`**
 - **`.click()`**
 - **`.offset()`**
3. Que le **span id="my_index"** (dans le bouton) affiche le nombre de clics. Tuyaux :
 - compteur en variable globale
 - **`.html()`**



Évènement clavier : exo 33

Question :
Que fait ce
code jQuery ?

```
<div id="lumiere">  
    <textarea id="target"></textarea>  
</div>
```

```
...  
$(function() {  
    $('#target').keydown(function() {  
        $('#lumiere').css('background-color', 'green');  
    });  
    $('#target').keyup(function() {  
        $('#lumiere').css('background-color', 'white');  
    });  
});
```



Évènement clavier : exo 35

Fichier de départ : **exo35_image_clavier_start.html**

1. Positionner l'image au milieu de la page
 - **`$(window).width()`**
 - **`.offset()`**
2. Afficher le code de la touche pressée dans le span d'id "unelettre"
 - **`e.which`**
 - **`.text()`**
3. Que l'image bouge de 10 pixels vers le haut/bas/G/D, en fonction de la touche de flèche pressée : flèche haut/bas/G/D
 - **`e.which`**
 - **`.offset()`**



Événement : timer : exo 37

- Évènement déclenché automatiquement par le navigateur à intervalle de temps régulier
 - Fonction principale : **setInterval(*function*, *délai*) ;**
 - **function** : code à exécuter à chaque intervalle de temps
 - **délai** : intervalle de temps, en millisecondes
 - Exemple : Une horloge sur la page web est rafraichie chaque seconde.
 - Application de **setInterval()**
 - La fonction **Horloge()** est exécutée chaque seconde.
- ```
$ (function() {
 setInterval(Horloge, 1000) ;
}) ;
```

**Horloge**

21:18:52



# Formulaire



= BUROTIX ()

# Formulaire : Méthode : `.val()`

- Accéder à la valeur d'un élément
  - `input`
  - `radio`
  - `checkbox`
  - `select/option`
- Exemples
  - `$('#nom').val()`
    - Lit le nom de l'utilisateur.
  - `$(':radio#H:checked').val()`
    - Lit l'état du bouton `radioH` et renvoie `true` si le bouton est sélectionné, sinon `false`.

## Code HTML

Nom d'utilisateur

```
<input type="text"
 id="nom">
```

Sexe

```
H <input type="radio"
 id="H"
 name="sexe"
 value="H">
```

```
F <input type="radio"
 id="F"
 name="sexe"
 value="F">
```



# Formulaire : Méthode : `.val()`

- Accéder à la valeur d'un élément
  - `input`
  - `radio`
  - `checkbox`
  - `select/option`
- Exemples
  - `$('select#fonction').val()`
    - Lit l'élément sélectionné dans la liste déroulante.
  - `$('#fonction').val('retraite')`
    - Sélectionne **Retraité** dans la liste déroulante.

## Code HTML

### Fonction

```
<select id="fonction">
 <option value="etudiant">
 Etudiant
 </option>
 <option value="ingenieur">
 Ingénieur
 </option>
 <option value="enseignant">
 Enseignant
 </option>
 <option value="retraite">
 Retraité
 </option>
</select>
```

# Formulaire : exo 41

- Écrire dans un champ input  
`$('input#nom').val('Michel');`
- Écrire dans un champ password  
`$('input#pass').val('CeciEstMonMotDePasse');`
- Imposer une sélection dans une liste  
`$('#fonction').val('retraite');`
- Imposer une sélection dans une radio  
`$('input:radio').val(['H']);`
- Lire une sélection dans une radio  
`alert("H selected ? " + $('input:radio#H:checked').val());`  
`alert("F selected ? " + $('input:radio#F:checked').val());`





# Formulaire : exo 42

- Manipulation d'une table HTML à l'aide du clavier
  - Navigation : **left, down, right, up**
  - Édition : **F2**
- Fichier : **exo42\_navigate\_in\_table.html**
  - Comprenez d'abord le code
    - `.index()` ? `nb_cols` ?
  - Exercice : Si l'utilisateur a le focus sur le premier **TD** du premier **TR**, alors que **left** déplace le focus sur le dernier **TD** du dernier **TR**. Et vice-versa.
  - Exercice : Si l'utilisateur a le focus sur un **TD** du premier **TR**, alors que **up** déplace le focus sur le **TD** correspondant du dernier **TR**. Et vice-versa.