

# Les bases de jQuery

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Les sélecteurs</b>	<b>3</b>
<b>III. Exercice : Appliquez la notion</b>	<b>6</b>
<b>IV. Interagir avec un élément</b>	<b>7</b>
<b>V. Exercice : Appliquez la notion</b>	<b>10</b>
<b>VI. Ajouter, supprimer un élément</b>	<b>11</b>
<b>VII. Exercice : Appliquez la notion</b>	<b>13</b>
<b>VIII. Les événements</b>	<b>14</b>
<b>IX. Exercice : Appliquez la notion</b>	<b>16</b>
<b>X. Délégation d'événement</b>	<b>16</b>
<b>XI. Exercice : Appliquez la notion</b>	<b>18</b>
<b>XII. Les animations</b>	<b>19</b>
<b>XIII. Exercice : Appliquez la notion</b>	<b>21</b>
<b>XIV. Auto-évaluation</b>	<b>22</b>
A. Exercice final .....	22
B. Exercice : Défi .....	24
<b>Solutions des exercices</b>	<b>27</b>

## I. Contexte

**Durée :** 1 h

**Environnement de travail :** Visual Studio Code

**Pré-requis :** HTML, CSS, JavaScript, présentation de la librairie jQuery

### Contexte

Dans la présentation de jQuery, nous avons vu comment l'inclure dans nos projets web, ainsi que son fonctionnement de base.

Nous allons entrer dans le détail et voir techniquement comment interagir avec les éléments du DOM existants, comment ajouter ou supprimer de nouveaux éléments, puis nous évoquerons la programmation événementielle avec l'utilisation des événements de base de JavaScript, la délégation d'événements, ainsi que la mise en place d'animations.

## II. Les sélecteurs

### Objectifs

- Sélectionner un élément
- Sélectionner des éléments en profondeur
- Sélectionner l'élément courant

### Mise en situation

Avant toute interaction avec le DOM, il faut être capable de sélectionner n'importe quel élément, qu'importe où il se trouve sur une page web. Nous allons entrer dans le détail des différents principes de sélection.

### Rappel Syntaxe de base

Avec jQuery, la sélection d'éléments se fait de la même façon qu'avec des sélecteurs CSS.

Pour rappel, la syntaxe est la suivante : `$ ( 'selecteur' )`. `$` étant l'alias de `jQuery()`.

Une méthode est ensuite appliquée à l'élément sélectionné : `$ ( 'selecteur' ) .methode ()`.

### Les sélecteurs

Un sélecteur par identifiant commence par `"#"` et un sélecteur par classe commence par `"."`, suivi du nom du sélecteur.

Il est possible de sélectionner un élément par :

- sa balise : `$ ( 'div' )`
- son identifiant : `$ ( '#bases' )`
- sa classe : `$ ( '.bases' )`

### Exemple

```
1 <div id="bases">Bases de jQuery</div>
1 $(' #bases') // Sélectionne <div id="bases">
```

### Complément La sélection avancée

Pour aller plus loin, il est possible de sélectionner en profondeur dans le DOM, grâce à la notion de parent, frère et enfant. Pour cela, il sera possible d'appliquer deux techniques :

1. L'enchaînement des sélecteurs
2. Les pseudo-classes CSS

Pour rappel, une pseudo-classe est un mot-clé ajouté à la suite d'un sélecteur. Elle permet d'inclure l'état d'un élément dans la sélection et facilite certaines écritures.

### Exemple Enchaînement des sélecteurs

```
1 <div id="content" class="content-div">
2   <p>Présentation de jQuery</p>
3 </div>
1 // Sélection du paragraphe contenu dans la div.
2 $('div p') // Sélection de la balise div puis de la balise p.
3 $(' #content p') // Sélection de l'élément portant l'id #content puis de la balise p.
```

### Exemple Avec les pseudo-classes

```
1 <div id="content" class="content-div">
2   <p>Bases de jQuery</p>
3 </div>
1 $(' #content:first-child') // Sélectionnera également le paragraphe, 1er enfant de la div #content.
```

### Complément

Voici quelques-unes des pseudo-classes CSS les plus célèbres :

- `:first-child` : le premier enfant
- `:last-child` : le dernier enfant
- `:last-of-type` : le dernier élément d'un type
- `:hover` : au survol de l'élément
- `:before` : avant l'élément
- `:after` : après l'élément
- `nth-child(x)` : enfant n° x

Il en existe beaucoup d'autres, toutes indiquées dans la documentation officielle de jQuery.

### Méthode Sélection multiple

Parfois, des méthodes devront être appliquées à des éléments différents, avec des identifiants et des classes divers.

Au lieu de réécrire le code pour chaque élément, il est possible de lister les sélecteurs en les séparant par une virgule.

**Exemple**

```
1 $('h1, h2, h3, .titre') // Sélectionnera les balises h1, h2 et h3 ainsi que les éléments ayant la classe .titre
```

**Le sélecteur this**

Ce sélecteur permet d'agir sur l'objet courant. Soit le code suivant :

```
1 $('#content p').text('Bonjour tout le monde !')
```

Ici, nous écrivons "Bonjour tout le monde !" dans le paragraphe de notre `div`. Nous aurions pu écrire l'équivalent :

```
1 $('#content p').html(function() {  
2     $(this).text('Bonjour tout le monde !')  
3 })
```

Présentation de jQuery  
Bonjour tout le monde !

L'utilisation de `this` n'est possible qu'à partir d'une fonction anonyme, que nous passons en paramètre de la méthode utilisée.

`this` est passé en tant qu'objet à jQuery et non comme une chaîne de caractères : `$(this)` et non pas `$('this')`.

**Attention**

Côté performance, nous préférons les sélecteurs par identifiant, car ce dernier étant unique, il sera plus facilement ciblé dans le DOM.

```
1 $('#content')
```

Évitons, par exemple, les recherches trop générales sur des classes :

```
1 $('.content')
```

Utilisons plutôt la notion parent/enfant afin de cibler au mieux les éléments :

```
1 $('div .content')
```

**Syntaxe**   **À retenir**

- Un sélecteur s'écrit avec l'alias `$` suivi du nom du sélecteur entre parenthèses : `$('#mon-sélecteur')` ou encore `$('balise .mon-sélecteur')`.
- Une sélection peut être multiple : `$('selecteur1, selecteur2, selecteur3')`.
- `this` permet de cibler l'élément courant : `$(this)`.

**Complément**

Documentation de jQuery<sup>1</sup>

Pseudo-classes<sup>2</sup>

<sup>1</sup> <https://api.jquery.com/>

<sup>2</sup> <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes>

### III. Exercice : Appliquez la notion

Vous disposez du code HTML suivant :

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width">
6   <title>Les bases de jQuery</title>
7 </head>
8 <body>
9 <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.4.1.min.js"></script>
10 <script src="script.js"></script>
11
12 <section>
13   <article>
14     <h1>Les bases de jQuery</h1>
15
16     <h2>Syntaxe générale</h2>
17     <p class="content">
18       Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
19       incididunt
20       ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
21       exercitation ullamco
22       laboris nisi ut aliquip ex ea commodo consequat
23     </p>
24     <h2>Les sélecteurs</h2>
25     <p class="content">
26       Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
27       incididunt
28       ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
29       exercitation ullamco
30       laboris nisi ut aliquip ex ea commodo consequat
31     </p>
32     <h2>La sélection avancée</h2>
33     <p class="content">
34       Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
35       incididunt
36       ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
37       exercitation ullamco
38       laboris nisi ut aliquip ex ea commodo consequat
39     </p>
40     <h2>Conclusion</h2>
41   </article>
42 </section>
43 </body>
44 </html>

```

#### Question 1

[solution n°1 p.29]

À l'aide d'un seul sélecteur, appliquez un texte vert à tous les éléments h1, h2 ou de classe content.

#### Indice :

css('key', 'value') permet d'ajouter ou modifier la valeur d'un attribut css.

#### Question 2

[solution n°2 p.29]

À l'aide d'un sélecteur, appliquez un texte en italique au dernier paragraphe .content de la page.

## IV. Interagir avec un élément

### Objectifs

- Récupérer le code HTML
- Modifier le CSS
- Parcourir une liste d'éléments
- Associer des données à un élément

### Mise en situation

jQuery offre une multitude de méthodes permettant de créer de l'interaction. Nous allons voir les prémices de la programmation événementielle, à savoir comment agir sur le code HTML et CSS, comment parcourir des listes d'éléments et utiliser des données propres à jQuery.

#### Méthode Récupérer le code HTML

La méthode `html()` permet de récupérer le code HTML d'un élément. Il s'agit de la réécriture d'`innerHTML` de JavaScript.

#### Exemple

```
1 <div id="content">
2   <h3>Présentation de jQuery</h3>
3   <p>Enfant de #content</p>
4 </div>

1 $(document).ready(() => {
2   console.log($('#content').html())
3 })

1 // Affichera :
2 <h3>Présentation de jQuery</h3>
3 <p>Enfant de #content</p>
```

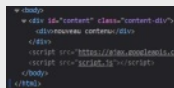
#### Attention

Attention, si on lui donne un argument, cette méthode servira alors à remplacer du contenu HTML.

#### Exemple

```
1 $('#content').html('<div>Nouveau contenu</div>')
```

Les éléments `h3` et `p` seront remplacés par cette nouvelle div.



```
<div id="content" class="content-div">
  <div>Nouveau contenu</div>
</div>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="script.js"></script>
</body>
</html>
```

#### Méthode Modifier le CSS

La méthode `css()` permet d'ajouter ou modifier les propriétés CSS d'un élément. Elle prend comme paramètres le nom de la propriété CSS et sa valeur.

### Exemple

```
1 <div id="content" class="content-div">
2   <h3>Bonjour tout le monde !</h3>
3 </div>
```

```
1 $(document).ready(() => {
2   $('h3').css('color', 'blue')
3 })
```

Ici, la propriété `color`, avec `blue` comme valeur, est ajoutée à l'élément HTML `h3`.

Bonjour tout le monde !

### Méthode Parcourir une liste d'éléments

Si une sélection retourne plusieurs éléments, la méthode `each()` permet de les parcourir.

L'intérêt est de pouvoir récupérer les éléments un à un pour leur appliquer une ou différentes méthode(s) selon les besoins.

Elle prend en paramètre une fonction anonyme prenant elle-même deux paramètres non obligatoires :

- `index` : position de l'élément dans la liste,
- `element` : l'élément courant de la liste (équivalent à `this`).

### Exemple

```
1 <div id="content">
2   <ul id="list">
3     <li>PHP</li>
4     <li>HTML</li>
5     <li>CSS</li>
6     <li>jQuery</li>
7   </ul>
8 </div>
```

```
1 $(document).ready(() => {
2   $('#list li').each(function() {
3     console.log($(this).text())
4   })
5 })
```

```
1 // Affichera :
2 PHP
3 HTML
4 CSS
5 jQuery
```

Dans cet exemple, `this` aurait pu être remplacé par `element` si ce dernier était passé en paramètre.



**Méthode** Associer des données à un élément

La méthode `data()` permet de créer des variables utilisables seulement par jQuery. Elles ne seront en aucun cas incluses dans le DOM. Elle prend comme paramètres :

- `element` : l'élément pour lequel associer la variable,
- `key` : le nom de la variable,
- `value` : la valeur de la variable (peut être une simple donnée de type entier, une chaîne de caractères ou bien un tableau ou un objet JavaScript).

**Exemple**

```
1 <div id="content">
2   <ul id="list">
3     <li></li>
4     <li></li>
5     <li></li>
6     <li></li>
7   </ul>
8 </div>
```

Cette fois, les éléments de la liste sont vides.

Création par tableau :

```
1 $(document).ready(() => {
2   let element = $( "#content" );
3
4   $.data(element, "languages", ['PHP', 'HTML', 'CSS', 'jQuery']);
5
6   $('#list li').each(function(index) {
7     $(this).text($.data(element, "languages")[index]);
8   })
9 });
```

Création par objet :

```
1 $(document).ready(() => {
2   let element = $("#content");
3
4   $.data(element, "languages", {
5     0: "PHP",
6     1: "HTML",
7     2: "CSS",
8     3: "jQuery"
9   });
10
11   $('#list li').each(function(index) {
12     $(this).text($.data(element, "languages")[index]);
13   })
14 });
```

- PHP
- HTML
- CSS
- jQuery

Dans cet exemple, la donnée `languages` est associée à l'élément ayant l'id `#content`, ce qui signifie que `languages` ne sera disponible que pour cet élément.

Nous l'utilisons ensuite pour remplir les éléments de la liste.

### Méthode Récupérer la valeur d'un élément

La méthode `.val()` permet de récupérer la valeur d'un élément d'un formulaire tels qu'un input, un select ou encore un textarea. Cette méthode ne prend pas de paramètres.

### Exemple

Ici on récupère la valeur d'un select pour l'option "foo"

```
1 $( "select#foo" ).val();
```

### Syntaxe À retenir

- `html()` permet de récupérer ou d'ajouter du code HTML.
- `css()` permet d'ajouter et modifier des propriétés CSS.
- `each()` permet de parcourir une liste d'éléments.
- `data()` permet d'associer des données à un élément.
- `val()` permet de récupérer la valeur d'un élément.

### Complément

`html()`<sup>1</sup>

`css()`<sup>2</sup>

`each()`<sup>3</sup>

`data()`<sup>4</sup>

`val()`<sup>5</sup>

## V. Exercice : Appliquez la notion

Vous disposez du code HTML suivant :

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width">
6   <title>Les bases de jQuery</title>
7 </head>
8 <body>
9 <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.4.1.min.js"></script>
10 <script src="script.js"></script>
11
12 <section>
13   <article>
14     <h1>Les bases de jQuery</h1>
15
```

1 <https://api.jquery.com/html/#html>

2 <https://api.jquery.com/css/#css2>

3 <https://api.jquery.com/each/#each-function>

4 <https://api.jquery.com/jQuery.data/#jQuery-data-element-key-value>

5 <https://api.jquery.com/val/>

```

16     <h2>Objectifs</h2>
17     <ul id="goals">
18         <li>Objectif 1</li>
19         <li>Objectif 2</li>
20         <li>Objectif 3</li>
21         <li>Objectif 4</li>
22     </ul>
23
24     <div id="introduction">
25         <h2>Introduction</h2>
26         <p class="content">
27             Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
28             incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
29             exercitation ullamco
30             laboris nisi ut aliquip ex ea commodo consequat
31         </p>
32     </div>
33
34     <div id="introduction-copy">
35
36     </div>
37 </article>
38 </body>
39 </html>
40

```

**Question 1**

[solution n°3 p.29]

Dupliquez le contenu de `#introduction` pour l'insérer dans `#introduction-copy`.

**Indice :**

Pour dupliquer du contenu, peut-être pourriez-vous lire le contenu dans un premier temps, et insérer celui-ci dans un second temps ?

**Question 2**

[solution n°4 p.29]

Initialisez la liste d'éléments `#goals` grâce aux valeurs suivantes :

- Récupérer le code HTML
- Modifier le CSS
- Parcourir une liste d'éléments
- Associer des données à un élément

**VI. Ajouter, supprimer un élément****Objectifs**

- Ajouter un élément
- Supprimer un élément

**Mise en situation**

Il existe des méthodes permettant de modifier le DOM en ajoutant de nouveaux éléments, en copiant des éléments existants ou en supprimant des éléments.

### Méthode Ajouter un élément

jQuery permet l'ajout d'éléments dans le DOM grâce aux méthodes `append()` et `prepend()`. Elles prennent en paramètre le nouvel élément à ajouter.

`append` permet d'ajouter un élément en tant que **dernier enfant** de l'élément cible, `prepend` en tant que **premier enfant**.

### Exemple

```
1 <div id="content">
2   <ul id="list">
3     <li>PHP</li>
4   </ul>
5 </div>

1 $(document).ready(() => {
2   let list = $('ul')
3
4   list.prepend('<li>jQuery</li>')
5   list.append('<li>HTML</li>')
6 });

1 // Affichera :
2 . jQuery // résultat de prepend.
3 . PHP
4 . HTML // résultat de append.
```

### Complément

Il est possible d'ajouter un élément déjà existant à un autre endroit de la page. Pour cela, il suffit de passer en paramètre un sélecteur à la méthode utilisée.

```
1 $('ul #other-list').append($('#list:first-child'))
2 // Ajoute le premier élément de la liste à la fin d'une autre liste.
```

### Méthode Supprimer un élément

jQuery permet la suppression d'éléments dans le DOM grâce à la méthode `remove()`. Elle prend en paramètre optionnel un élément.

### Remarque

Les enfants de l'élément, ainsi que les événements et données associées, seront aussi supprimés.

### Exemple

```
1 <div id="content">
2   <div id="add">
3     <p>Ajouter un élément au DOM.</p>
4   </div>
5   <div id="remove">
6     <p>Supprimer un élément du DOM.</p>
7   </div>
8 </div>

1 $(document).ready(() => {
2   $('#remove').remove()
3 });
```

```

<div id="test">
  <div id="test">
    <div id="test"></div>
    <div id="test"></div>
  </div>
</div>

```

La div avec l'id #remove a été supprimée du DOM, ainsi que ses éléments enfants.

Nous aurions aussi pu écrire : `$('div').remove('#remove')`.

### Syntaxe À retenir

- Les méthodes `append()` et `prepend()` servent à ajouter des éléments dans le DOM.
- `remove()` sert à supprimer des éléments.

### Complément

`append()`<sup>1</sup>

`prepend()`<sup>2</sup>

`remove()`<sup>3</sup>

## VII. Exercice : Appliquez la notion

Vous disposez du code HTML suivant :

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width">
6   <title>Les bases de jQuery</title>
7 </head>
8 <body>
9 <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.4.1.min.js"></script>
10 <script src="script.js"></script>
11
12 <section>
13   <article>
14     <h1>Les bases de jQuery</h1>
15
16     <h2>Objectifs</h2>
17     <ul id="goals">
18       <li>Objectif 1</li>
19       <li>Objectif 2</li>
20       <li>Objectif 3</li>
21       <li>Objectif 4</li>
22     </ul>
23
24     <div id="introduction">
25       <h2>Introduction</h2>
26       <p class="content">
27         Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

```

incididunt

1 <https://api.jquery.com/append/>

2 <https://api.jquery.com/prepend/>

3 <https://api.jquery.com/remove/>

```

28         ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco
29         laboris nisi ut aliquip ex ea commodo consequat
30     </p>
31 </div>
32 </article>
33 </section>
34 </body>
35 </html>
36

```

### Question 1

[solution n°5 p.29]

Supprimez le paragraphe #introduction.

### Question 2

[solution n°6 p.30]

Ajoutez deux nouveaux éléments à la liste #goals :

- Un élément "Objectif 0" en début de liste
- Un élément "Objectif 5" en fin de liste

## VIII. Les événements

### Objectifs

- Ajouter un écouteur d'événement
- Simuler un écouteur d'événement

### Mise en situation

Comme en JavaScript pur, jQuery permet l'ajout d'écouteurs d'événements qui seront déclenchés par le navigateur ou l'utilisateur. Il peut s'agir d'un chargement de page, d'un clic sur un bouton ou encore de la manipulation des champs d'un formulaire.

#### Remarque

Sans le savoir, nous avons déjà utilisé un écouteur d'événement au cours des précédentes étapes de l'initiation à jQuery.

Lorsque nous écrivons : `$(document).ready(function() {})`, l'événement **ready** est écouté. Il signifie, quand le DOM est chargé, d'exécuter le script jQuery correspondant.

#### Méthode Écouter un événement

Créer un écouteur d'événement avec jQuery est très simple : il suffit d'appliquer la méthode correspondant à l'écouteur souhaité sur un élément.

Parmi les écouteurs les plus utilisés :

- `click()` : cliquer sur un bouton de la souris,
- `mousedown()` : presser un bouton de la souris,
- `keydown()`, `keyup()` : enfoncer, relâcher un bouton du clavier,
- `focus()` : focaliser sur un champ de formulaire,
- `submit()` : soumettre un formulaire.

**Exemple** Événement formulaire

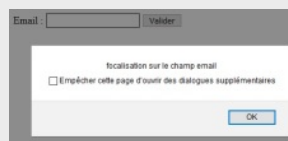
```

1 <div id="content">
2   <form id="form">
3     <label for="email">Email :</label>
4     <input type="text" id="email" name="email" value="" />
5     <button id="btn-submit">Valider</button>
6   </form>
7 </div>

1 $(document).ready(() => {
2   $('#email').focus(function() {
3     alert('focalisation sur le champ email')
4   })
5 });

```

Ici, l'événement **focus** est écouté sur le champ `email` du formulaire. En cliquant dans le champ pour saisir, le code correspondant sera exécuté.

**Complément** Simuler un événement

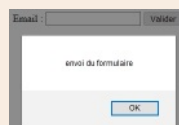
Il est possible de déclencher un événement sans attendre une action de l'utilisateur grâce à la méthode `trigger()`. Elle prend en paramètre le nom de l'événement (**click**, **change**, **submit**).

```

1 $(document).ready(() => {
2   $('#form').submit(function(e) {
3     e.preventDefault()
4     alert('envoi du formulaire')
5   })
6
7   $('#form').trigger('submit')
8 });

```

Ici, l'écouteur **submit** est simulé dès le chargement de la page, ce qui aura pour effet d'envoyer le formulaire sans cliquer sur le bouton.

**Complément** Arrêter la propagation

Comme en JavaScript pur, `e.preventDefault()` peut être utilisé pour stopper le comportement par défaut de l'événement.

**Syntaxe** À retenir

- Un écouteur d'événement s'écrit : `$('#selecteur').click()`.
- Un événement peut être déclenché : `$('#selecteur').trigger('click')`.

**Complément**

trigger()<sup>1</sup>  
e.preventDefault()<sup>2</sup>

## IX. Exercice : Appliquez la notion

Vous disposez du code HTML suivant :

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width">
6     <title>Les bases de jQuery</title>
7 </head>
8 <body>
9 <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.4.1.min.js"></script>
10 <script src="script.js"></script>
11
12 <section>
13     <div id="content">
14         <form id="form">
15             <label for="comment">Commentaire :</label>
16             <textarea id="comment" name="comment" cols="30"></textarea>
17         </form>
18     </div>
19     <div>
20         Commentaire saisi :
21         <p id="commentValue">Pas de commentaire</p>
22     </div>
23 </section>
24 </body>
25 </html>
```

### Question

[solution n°7 p.30]

Lorsqu'un utilisateur saisit un commentaire, faites en sorte que la valeur de la zone de texte se retrouve dupliquée dans le paragraphe #commentValue au fil de la saisie de l'utilisateur.

### Indice :

Vous pourriez utiliser la fonction keyup<sup>3</sup>.

## X. Délégation d'événement

### Objectifs

- Ajouter un écouteur d'événement avec on ( )
- Déléguer un événement

1 <https://api.jquery.com/trigger/>

2 <https://api.jquery.com/event.preventDefault/#event-preventDefault>

3 <https://api.jquery.com/keyup/>



## Mise en situation

La programmation événementielle réserve quelques surprises. Imaginons créer un écouteur d'événement **click** sur les boutons de notre site. Si nous rajoutons dynamiquement d'autres boutons, ceux-ci ne seront pas reliés à notre écouteur.

Le rôle de la délégation d'événement est précisément de résoudre cette problématique.

### Méthode Méthode on()

La délégation avec jQuery implique l'utilisation d'une autre méthode d'ajout d'écouteur d'événement que celle vue dans le chapitre précédent. Nous utiliserons `on()` avec deux paramètres :

- le nom de l'événement,
- une fonction anonyme contenant le code à exécuter.

Reprenons le formulaire créé précédemment.

Email:  Valider

```
1 $(document).ready(() => {
2     $('#email').on('focus', function() {
3         alert('focalisation sur le champ email')
4     })
5 });
```



De la même manière, le message d'alerte s'affichera quand le focus se fera sur le champ `email`.

### Méthode Délégation

Pour déléguer un écouteur d'événement, la méthode `on()` accepte un nouveau paramètre : l'élément sur lequel déléguer, qui se placera en seconde position.

Considérons l'écouteur **focus** placé sur les éléments `input` sans délégation.

Dans un premier temps, seul le champ `email` est présent dans le formulaire.

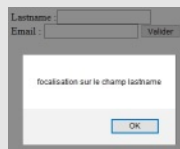
```
1 $(document).ready(() => {
2     let form = $('#form')
3
4     $('input').on('focus', function() {
5         alert(`focalisation sur le champ ${$(this).attr('name')}`)
6     })
7
8     form.prepend('<input type="text" id="lastname" name="lastname" value="" />')
9     form.prepend('<label for="lastname">Lastname :</label>')
10 });
```

Ajoutons dynamiquement le champ `lastname` grâce à `prepend`. Dans ce cas, le message d'alerte s'affichera au focus sur le champ `email`, mais pas sur le champ `lastname`.

Lastname:   
Email:  Valider

Avec la délégation :

```
1 $(document).ready(() => {
2     let form = $('#form')
3
4     form.on('focus', 'input', function() {
5         alert(`focalisation sur le champ ${$(this).attr('name')}`)
6     })
7
8     form.prepend('<input type="text" id="lastname" name="lastname" value="" />')
9     form.prepend('<label for="lastname">Lastname :</label>')
10 });
```



Le message d'alerte s'affiche bien au focus sur le champ lastname.

### Fondamental

Pour que la délégation fonctionne, l'écouteur a été placé sur le parent direct des éléments input. `$('#input').on('focus', function() {})` est devenu : `$('#form').on('focus', 'input', function() {})`.

Autrement dit, l'écouteur est placé sur le parent et sera délégué par celui-ci aux éléments input.

### Conseil

Pour optimiser l'exécution du script, la délégation doit se faire sur le parent le plus proche non créé dynamiquement.

### Syntaxe À retenir

- Pour déléguer un écouteur d'événement, la méthode `on()` doit être utilisée : `on('event', 'element', function() {})`.

### Complément

`on()`<sup>1</sup>

## XI. Exercice : Appliquez la notion

Vous disposez du code HTML suivant :

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width">
6     <title>Les bases de jQuery</title>
```

<sup>1</sup> <https://api.jquery.com/on/>

```

7 </head>
8 <body>
9 <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.4.1.min.js"></script>
10 <script src="script.js"></script>
11
12 <section>
13   <div id="content">
14     <form id="form">
15       <label for="comment">Commentaire :</label>
16       <textarea id="comment" name="comment" class="element" cols="30"></textarea>
17     </form>
18   </div>
19   <div>
20     Valeur de l'élément modifié :
21     <p id="elementValue">Rien à afficher</p>
22   </div>
23 </section>
24 </body>
25 </html>

```

```

1 $(document).ready(() => {
2   $('#element').keyup(function () {
3     $('#elementValue').html($(this).val())
4   })
5
6   let form = $('#form')
7
8   form.prepend('<input type="text" id="lastname" name="lastname" class="element" value=""
9 />')
9   form.prepend('<label for="lastname">Lastname :</label>')
10 });

```

### Question

[solution n°8 p.30]

Adaptez le script afin que la valeur de n'importe quel élément `.element` ajouté dynamiquement ou non soit affichée dans `#elementValue` lorsqu'elle est modifiée.

## XII. Les animations

### Objectif

- Animer le contenu d'une page web

### Mise en situation

Par définition, une interaction provoque un effet de changement. Jusqu'à maintenant, nous avons manipulé des éléments fixes, mais jQuery permet également de provoquer des effets de mouvement avec les animations.

## Syntaxe Animations natives

Certaines animations sont prédéfinies par jQuery. Elles sont utilisables en appliquant à un élément la méthode correspondant à l'animation souhaitée.

Nous noterons parmi les plus connues :

- `hide()` : cache un élément en réduisant sa taille et son opacité
- `show()` : montre un élément en augmentant sa taille et son opacité
- `fadeIn()` : montre un élément en augmentant son opacité
- `fadeOut()` : cache un élément en réduisant son opacité

## Méthode

Ces méthodes peuvent prendre en paramètre une durée en millisecondes. Il s'agira de la durée de l'animation : `$('element').hide(2000)`.

Il existe des durées prédéfinies : **fast = 200ms, normal = 400ms et slow = 600ms.**

On utilisera alors la syntaxe `$('element').hide('slow')`.

## Remarque

Un délai peut être appliqué avant une animation grâce à la méthode `delay`, qui prend en paramètre une durée en millisecondes : `$('element').delay(2000).show('fast');`

## Exemple

```
1 <div id="content" style="width: 200px; height: 200px; background-color: aqua;">
2 </div>

1 $(document).ready(() => {
2     $('#content').hide(2000).delay(1000).show(2000)
3 });
```

## Méthode Animations personnalisées

La méthode `animate()` permet de créer ses propres animations en jouant sur les propriétés CSS de l'élément. Elle peut prendre jusqu'à quatre paramètres :

- `properties` : un objet contenant les propriétés CSS et leurs valeurs (obligatoire)
- `duration` : une durée (optionnel, 400 ms par défaut)
- `easing` : le style d'animation (optionnel : *swing* ou *linear*)
- `complete` : une fonction à exécuter en fin d'animation (optionnel)

## Exemple

```
1 $(document).ready(() => {
2     $('#content').animate({width: '500px'}, 2000).delay(1000).animate({height: '500px'}, 2000)
3 });
```

Dans ce cas, nous avons pu décomposer la modification de largeur et hauteur en deux animations.

**Complément**

La méthode `animate` peut aussi s'écrire : `$('element').animate({css}, {options})`. D'abord, les propriétés CSS et leurs valeurs, ensuite un objet d'options générales, comprenant entre autres la durée et le style d'animation.

**Attention**

Côté performance, un abus d'utilisation de la méthode `animate` pourrait entraîner des ralentissements significatifs. Il faut la préférer pour de petites animations. Une alternative peut être la librairie **Velocity.js**, plus optimisée pour les animations lourdes.

**Syntaxe**    **À retenir**

Il existe deux syntaxes pour faire appel à `animate()` :

- `$('element').animate({css}, duration, easing, complete)`
- `$('element').animate({css}, {options})`

La première est plus courante et suffira dans la majorité des cas. La seconde permet de personnaliser encore plus les animations pour des besoins spécifiques.

**Complément**

`animate()`<sup>1</sup>

`velocity.js`<sup>2</sup>

### XIII. Exercice : Appliquez la notion

Vous disposez du code HTML suivant :

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width">
6     <title>Les bases de jQuery</title>
7 </head>
8 <body>
9 <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.4.1.min.js"></script>
10 <script src="script.js"></script>
11
12 <section>
13     <article>
14         <h1>Les bases de jQuery</h1>
15
16         <h2>Syntaxe générale</h2>
17         <p class="content">
18             Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
19             incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
20             exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat

```

<sup>1</sup> <https://api.jquery.com/animate/>

<sup>2</sup> <http://velocityjs.org/>

```

21     </p>
22     <h2>Les sélecteurs</h2>
23     <p class="content">
24         Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
25         incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
26         exercitation ullamco
27         laboris nisi ut aliquip ex ea commodo consequat
28     </p>
29     <h2>La sélection avancée</h2>
30     <p class="content">
31         Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
32         incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
33         exercitation ullamco
34         laboris nisi ut aliquip ex ea commodo consequat
35     </p>
36     <h2>Conclusion</h2>
37 </article>
38 </section>
39 </body>
40 <style>
41     h1, h2 {
42         display: none;
43     }
44 </style>
45 </html>

```

## Question

[solution n°9 p.30]

Affichez progressivement et lentement les différents titres et masquez progressivement les éléments de la classe `.content`.

## XIV. Auto-évaluation

### A. Exercice final

#### Exercice 1

[solution n°10 p.30]

#### Exercice

Quel sélecteur faut-il utiliser si l'on souhaite cibler tous les éléments `div` et paragraphes d'une page, indépendamment de sa structure ?

- ☐ `$('div p');`
- ☐ `$('div, p')`
- ☐ `$('div + p')`

#### Exercice

Selon la déclaration suivante, à quoi correspondra `$(this)` ?

```

1 $(' .element').each(function () {
2     console.log($(this).value())
3 })

```

- ☐ À la liste de tous les éléments disposant de la classe `.element`
- ☐ À l'élément courant parmi tous les éléments disposant de la classe `.element`

## Exercice

Parmi ces syntaxes, laquelle permet de récupérer le contenu HTML de `#content` ?

- ☐ `$(document).html('#content')`
- ☐ `$('#content').html()`
- ☐ `$('#content').text()`
- ☐ `$(document).text('#content')`

## Exercice

Quelle syntaxe doit-on utiliser si l'on souhaite afficher du texte brut dans un élément ?

- ☐ `$(element).html(content)`
- ☐ `$(element).text(content)`
- ☐ `$(content).html(element)`
- ☐ `$(content).text(element)`

## Exercice

On souhaite ajouter un nouvel élément à la fin d'une liste `ul`, quelle méthode faut-il utiliser ?

- ☐ `$('#ul').prepend('<li>jQuery</li>')`
- ☐ `$('#ul').append('<li>jQuery</li>')`
- ☐ `$('#ul').insertBefore('<li>jQuery</li>')`
- ☐ `$('#ul').insertAfter('<li>jQuery</li>')`

## Exercice

Quelle méthode peut-on utiliser pour supprimer définitivement un élément du DOM ?

## Exercice

Qu'effectuera le code suivant ?

```
1 $('#email').focus(function() {  
2     //do stuff  
3 })
```

- ☐ Il placera le focus sur le champ `#email` et effectuera une action
- ☐ Il effectuera une action lors du focus sur le champ `#email`

## Exercice

Grâce à quelle méthode est-il possible de simuler le déclenchement d'un événement ?

## Exercice

Quelle syntaxe faut-il respecter pour déléguer un écouteur d'événement ?

- ☐ on('event', 'element', function({}))
- ☐ on(element, function({}), event)
- ☐ on(function({}), event, element)

#### Exercice

Parmi ces méthodes, laquelle faut-il utiliser si l'on souhaite cacher un élément en réduisant son opacité sans toucher à sa taille ?

- ☐ hide
- ☐ show
- ☐ fadeOut
- ☐ fadeIn

### B. Exercice : Défi

Vous êtes chargé d'améliorer le formulaire de création de compte utilisateur, dont vous trouverez le code ci-dessous :

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="utf-8" />
5 </head>
6 <body>
7   <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.4.1.min.js"></script>
8   <script src="script.js"></script>
9   <h1>Accès utilisateur</h1>
10  <form id="form">
11    <fieldset>
12      <legend>Création de votre compte</legend>
13      <label for="name">Nom d'utilisateur</label>
14      <input type="text" name="name" id="name" class="js-hasHelpText" required
15      placeholder="Saisissez votre nom">
16      <label for="email">Adresse e-mail</label>
17      <input type="text" name="email" id="email" class="js-hasHelpText" required
18      placeholder="Saisissez votre email">
19      <label for="password">Votre mot de passe</label>
20      <input type="password" name="password" id="password" class="js-hasHelpText" required
21      placeholder="Saisissez votre mot de passe">
22      <label for="birthDate">Votre date de naissance</label>
23      <input type="date" name="birthDate" id="birthDate" class="js-hasHelpText" required
24      placeholder="Saisissez votre date de naissance">
25      <label for="comment">Commentaire</label>
26      <textarea name="comment" id="comment" class="js-hasHelpText" cols="30" rows="10">
27    </textarea>
28      <button type="submit">Soumettre le formulaire</button>
29    </fieldset>
30  </form>
31 </body>
32 <style>

```



```
33  /**
34  Attention, ce CSS est là uniquement pour rendre le formulaire "agréable" à la lecture
35  sans que vous n'ayez
36  à récupérer deux fichiers distincts.
37  Dans un cas d'usage "réel", ces éléments doivent être externalisés
38  */
39  body {
40      font-family: Calibri, serif;
41  }
42  form {
43      max-width: 75%;
44  }
45  form label {
46      display: block;
47      font-weight: bold;
48      margin-bottom: 10px;
49  }
50  }
51  fieldset {
52      border: 1px solid lightgray;
53      background-color: rgba(225, 233, 255, 0.25);
54  }
55  }
56  legend {
57      font-style: italic;
58      font-size: 1.1em;
59      padding: 5px;
60  }
61  }
62  form input, form textarea {
63      display: inline-block;
64      margin-bottom: 10px;
65      padding: 10px;
66      width: 80%;
67  }
68  }
69  button[type=submit] {
70      padding: 10px;
71      margin-top: 15px;
72      width: auto;
73      display: block;
74  }
75  }
76  .help-text {
77      color: #2b2b2b;
78      font-size: 12px;
79      display: none;
80  }
81  }
82 </style>
83 </html>
84
```

```

1 $(document).ready(() => {
2     let constraintsText = {
3         name: 'Le nom d\'un utilisateur ne peut contenir plus de 20 caractères et ne doit
4             contenir que des lettres.',
5         email: 'Il doit s\'agir d\'un e-mail valide',
6         password: 'Le mot de passe ne peut contenir que des chiffres et des lettres. Il doit
7             disposer d\'au moins 6 caractères',
8         birthDate: 'L\'utilisateur doit être une personne majeure',
9         comment: 'Le commentaire ne peut excéder plus de 200 caractères'
10    }
11
12    $('input').each(function () {
13        console.log($(this).attr('name'))
14    })
15 });

```

### Question 1

[solution n°11 p.32]

Vous disposez d'un certain nombre de messages de contraintes propres à chaque élément du formulaire.

Dans un premier temps, associez ces contraintes à chacun des champs du formulaire. Vous pourriez vous aider du **console.log** présent dans le script de base.

Par défaut, ces contraintes doivent être masquées. La contrainte concernée s'affichera uniquement lorsque l'utilisateur placera le focus sur ce champ, et disparaîtra s'il quitte le champ.

Vous prendrez soin de les faire apparaître lentement.

**Attention, vous n'êtes pas chargé de valider les données du formulaire, seulement de gérer l'affichage.**

#### Accès utilisateur

Création de votre compte

Nom d'utilisateur

Le nom d'un utilisateur ne peut contenir plus de 20 caractères et ne doit contenir que des lettres.

Adresse e-mail

Votre mot de passe

Votre date de naissance

Commentaire

Soumettre le formulaire

#### Indice :

La fonction `next`<sup>1</sup> pourrait vous être utile.

#### Indice :

La fonction `attr`<sup>2</sup> pourrait vous être utile.

### Question 2

[solution n°12 p.33]

Rendez ce formulaire plus agréable à la lecture en modifiant les éléments suivants :

- Au chargement, les éléments `h1` / `legend` seront masqués, puis lentement affichés sur 1000 ms.
- Sur 2000 ms, la largeur du formulaire passera à 50 %.

1 <https://api.jquery.com/next/>

2 <https://api.jquery.com/attr/>

## **Solutions des exercices**



**p. 6 Solution n°1**

```
1 $(document).ready(() => {  
2   $('h1, h2, .content').css('color', 'green');  
3 });  
4
```

**p. 6 Solution n°2**

```
1 $(document).ready(() => {  
2   $('p.content:last-of-type').css('font-style', 'italic');  
3 });  
4
```

**p. 11 Solution n°3**

```
1 $(document).ready(() => {  
2   // On insère dans #introduction-copy le contenu lu dans #introduction  
3   $('#introduction-copy').html($('#introduction').html());  
4 });  
5
```

**p. 11 Solution n°4**

```
1 $(document).ready(() => {  
2   let goals = $( "#goals" )  
3  
4   $.data( goals, "goals", {  
5     0 : "Récupérer le code HTML",  
6     1 : "Modifier le CSS",  
7     2 : "Parcourir une liste d'éléments",  
8     3 : "Associer des données à un élément"  
9   })  
10  
11   $('#goals li').each(function(index) {  
12     $(this).text($.data( goals, "goals" )[index])  
13   })  
14 });  
15
```

**p. 14 Solution n°5**

```
1 $(document).ready(() => {  
2   $('#introduction').remove()  
3 });  
4
```

p. 14 Solution n°6

```
1 $(document).ready(() => {
2     let list = $('#goals')
3
4     list.prepend('<li>Objectif 0</li>')
5     list.append('<li>Objectif 5</li>')
6 });
```

p. 16 Solution n°7

```
1 $(document).ready(() => {
2     $('#comment').keyup(function () {
3         $('#commentValue').html($(this).val())
4     })
5 });
```

p. 19 Solution n°8

```
1 $(document).ready(() => {
2     let form = $('#form')
3
4     form.on('keyup', '.element', function () {
5         $('#elementValue').html($(this).val())
6     })
7
8     form.prepend('<input type="text" id="lastname" name="lastname" class="element" value=""
9 />')
10    form.prepend('<label for="lastname">Lastname :</label>')
11 });
```

p. 22 Solution n°9

```
1 $(document).ready(() => {
2     $('h1, h2').fadeIn('slow')
3     $('.content').fadeOut('slow')
4 });
```

Exercice p. 22 Solution n°10

Exercice

Quel sélecteur faut-il utiliser si l'on souhaite cibler tous les éléments div et paragraphes d'une page, indépendamment de sa structure ?

- ☐ \$('div p');
- ☒ \$('div, p')
- ☐ \$('div + p')

Exercice

Selon la déclaration suivante, à quoi correspondra \$(this) ?

```
1 $(' .element').each(function () {
2     console.log($(this).value())
3 })
```

- ☐ À la liste de tous les éléments disposant de la classe `.element`
- ☒ À l'élément courant parmi tous les éléments disposant de la classe `.element`

### Exercice

Parmi ces syntaxes, laquelle permet de récupérer le contenu HTML de `#content` ?

- ☐ `$(document).html('#content')`
- ☒ `$('#content').html()`
- ☐ `$('#content').text()`
- ☐ `$(document).text('#content')`

### Exercice

Quelle syntaxe doit-on utiliser si l'on souhaite afficher du texte brut dans un élément ?

- ☐ `$(element).html(content)`
- ☒ `$(element).text(content)`
- ☐ `$(content).html(element)`
- ☐ `$(content).text(element)`

### Exercice

On souhaite ajouter un nouvel élément à la fin d'une liste `ul`, quelle méthode faut-il utiliser ?

- ☐ `$('#ul').prepend('<li>jQuery</li>')`  
*Cette méthode l'ajoute en tant que premier enfant.*
- ☒ `$('#ul').append('<li>jQuery</li>')`
- ☐ `$('#ul').insertBefore('<li>jQuery</li>')`  
*Cette méthode l'ajouterait avant l'élément `ul`.*
- ☐ `$('#ul').insertAfter('<li>jQuery</li>')`  
*Cette méthode l'ajouterait après l'élément `ul`.*

### Exercice

Quelle méthode peut-on utiliser pour supprimer définitivement un élément du DOM ?

`remove`

### Exercice

Qu'effectuera le code suivant ?

```
1 $('#email').focus(function() {
2     //do stuff
3 })
```

- ☐ Il placera le focus sur le champ `#email` et effectuera une action
- ☒ Il effectuera une action lors du focus sur le champ `#email`

### Exercice

Grâce à quelle méthode est-il possible de simuler le déclenchement d'un événement ?

trigger

### Exercice

Quelle syntaxe faut-il respecter pour déléguer un écouteur d'événement ?

- ☒ on('event', 'element', function({}))
- ☐ on(element, function({}), event)
- ☐ on(function({}), event, element)

### Exercice

Parmi ces méthodes, laquelle faut-il utiliser si l'on souhaite cacher un élément en réduisant son opacité sans toucher à sa taille ?

- ☐ hide  
Cache un élément en réduisant sa taille et son opacité.
- ☐ show  
Montre un élément en augmentant sa taille et son opacité.
- ☒ fadeOut  
Cache un élément en réduisant son opacité.
- ☐ fadeIn  
Montre un élément en augmentant son opacité.

### p. 26 Solution n°11

```

1 $(document).ready(() => {
2     let constraintsText = {
3         name: 'Le nom d\'un utilisateur ne peut contenir plus de 20 caractères et ne doit
4         contenir que des lettres.',
5         email: 'Il doit s\'agir d\'un e-mail valide',
6         password: 'Le mot de passe ne peut contenir que des chiffres et des lettres. Il doit
7         disposer d\'au moins 6 caractères',
8         birthDate: 'L\'utilisateur doit être une personne majeure',
9         comment: 'Le commentaire ne peut excéder plus de 200 caractères'
10    }
11
12    $('.js-hasHelpText').each(function () {
13        let helpText = '<p class="help-text">' + constraintsText[$(this).attr('name')] +
14        '</p>'
15        $(this).after(helpText)
16    })
17
18    let form = $('#form')
19
20    form.on('focus', 'input, textarea', function () {
21        $(this).next('.help-text').show('slow')
22    })
23
24    form.on('blur', 'input, textarea', function () {
25        $('.help-text').hide()
26    })
27 })

```



```

23     })
24 });

```

**p. 26 Solution n°12**

```

1 $(document).ready(() => {
2     let constraintsText = {
3         name: 'Le nom d\'un utilisateur ne peut contenir plus de 20 caractères et ne doit
4             contenir que des lettres.',
5         email: 'Il doit s\'agir d\'un e-mail valide',
6         password: 'Le mot de passe ne peut contenir que des chiffres et des lettres. Il doit
7             disposer d\'au moins 6 caractères',
8         birthDate: 'L\'utilisateur doit être une personne majeure',
9         comment: 'Le commentaire ne peut excéder plus de 200 caractères'
10    }
11    $('.js-hasHelpText').each(function () {
12        let helpText = '<p class="help-text">' + constraintsText[$(this).attr('name')] +
13        '</p>'
14        $(this).after(helpText)
15    })
16    let form = $('#form')
17    form.animate({'width': '50%'}, 2000)
18    $('h1, legend').hide().show(1000)
19    form.on('focus', 'input, textarea', function () {
20        $(this).next('.help-text').show('slow')
21    })
22    form.on('blur', 'input, textarea', function () {
23        $(this).next('.help-text').hide()
24    })
25 });

```