

LES SÉLECTEURS

SÉLECTEURS D'ÉLÉMENTS

Les sélecteurs d'éléments permettent de sélectionner un ou plusieurs éléments du **DOM** par leur nom de balise.

- Exemple de sélecteur d'élément avec le nom de balise :

```
const paragraphe = document.querySelector('p'); // Sélectionne le premier élément p (paragraphe) dans la page
const tds = document.querySelectorAll('td'); // Sélectionne tous les éléments td (cellules de tableau) dans la page
```

SYNTAXE

```
$( "balise" )
```

EXEMPLES D'UTILISATION

```
$( "div" ) // Sélectionne tous les éléments div  
$( "p" ) // Sélectionne tous les éléments p (paragraphe)
```

SÉLECTEURS D'ATTRIBUTS

Les **sélecteurs d'attributs** permettent de sélectionner des éléments en fonction des **attributs HTML** et de leurs valeurs. Par exemple :

```
const elementsAvecAttribut = document.querySelectorAll("[data-custom-attribute]");
const elementsAvecAttributEtValeur = document.querySelectorAll("[data-custom-attribute='valeur']");
```

Exemples de sélecteurs d'attributs :

- [href] : sélectionne tous les éléments avec un attribut href
- [src="image.jpg"] : sélectionne tous les éléments avec un attribut src ayant pour valeur "image.jpg"

SYNTAXE

```
$( "element[attribute='value']" )
```

EXEMPLES D'UTILISATION

```
$(“input[type='text’]”) // Sélectionne les éléments input de type texte  
$(“a[href^='https://’]”) // Sélectionne les liens commençant par "https://"
```

SÉLECTEURS DE CLASSE

Les sélecteurs de classe permettent de sélectionner des éléments en fonction de leur **classe CSS**.

- Syntaxe : .classname

Exemple :

```
const elements = document.getElementsByClassName("example");
```

SYNTAXE

```
$( ".classname" )
```

EXEMPLES D'UTILISATION

```
$(".important") // Sélectionne tous les éléments avec la classe "important"  
$("p.important") // Sélectionne les éléments p (paragraphe) avec la classe "important"
```

SÉLECTEURS D'ID

Les **sélecteurs d'ID** permettent de sélectionner un élément unique par son **identifiant** (attribut "id").

```
document.getElementById("monID");
```

Attribut Utilisation

id	Sélectionner un élément
----	-------------------------

SYNTAXE

```
$ ("#identifier")
```

EXEMPLES D'UTILISATION

```
$( "#navigation" ) // Sélectionne l'élément avec l'ID "navigation"
```

SÉLECTEURS AVANCÉS

Les **sélecteurs avancés** permettent de sélectionner des éléments en fonction de leur **position**, de leur **relation** avec leurs voisins ou en fonction de **conditions plus complexes**.

Sélecteur	Description
:nth-child(n)	Sélectionne les éléments en position n
:first-child	Sélectionne le premier enfant
:last-child	Sélectionne le dernier enfant
:only-child	Sélectionne les éléments sans frères
:not (selector)	Sélectionne les éléments ne correspondant pas au sélecteur

SYNTAXE

```
$ ("element:first")  
$ ("element:last")  
$ ("element:even")  
$ ("element:odd")
```

EXEMPLES D'UTILISATION

```
$( "li:first" ) // Sélectionne le premier élément li  
$( "tr:even" ) // Sélectionne les éléments tr (ligne de tableau) en position paire
```

JQUERY - LES ÉVÉNEMENTS

GESTION DES ÉVÉNEMENTS

SYNTAXE

Pour attacher un **événement** à un élément avec **jQuery**, utilisez la syntaxe suivante :

```
$( "selector" ).event( function() {  
    // code à exécuter  
});
```

EXEMPLES D'UTILISATION

CLICK

```
$( "button" ).click(function() {  
    alert("Bouton cliqué");  
});
```

DOUBLE CLICK

```
$( "p" ).dblclick(function() {  
    alert("Paragraphe double cliqué");  
});
```

MÉTHODES COURANTES

Evénement	Syntaxe jQuery	Description
Click	<code>\$ ("selector").click (func)</code>	Exécuté lors d'un clic gauche
Double click	<code>\$ ("selector").dblclick (func)</code>	Exécuté lors d'un double clic gauche
mouseenter	<code>\$ ("selector").mouseenter (func)</code>	Exécuté lors de l'entrée du pointeur sur l'élément
mouseleave	<code>\$ ("selector").mouseleave (func)</code>	Exécuté lors de la sortie du pointeur de l'élément

DÉLÉGATION D'ÉVÉNEMENTS

SYNTAXE

Pour attacher un **événement délégué** avec **jQuery**, utilisez la syntaxe suivante :

```
$( "selector" ).on( "event", "delegated-selector", function() {  
    // code à exécuter  
});
```

EXEMPLES D'UTILISATION

```
$( "ul" ).on( "click", "li", function() {  
    alert("Élément de liste cliqué");  
});
```

AVANTAGES ET INCONVÉNIENTS

Avantages :

- Gère les **événements** pour les éléments ajoutés dynamiquement
- Améliore les **performances** en utilisant moins de gestionnaires d'événements

Inconvénients :

- Ne fonctionne pas pour les **événements natifs** de JavaScript

CRÉER ET DÉCLENCHER DES ÉVÉNEMENTS PERSONNALISÉS

SYNTAXE

Pour créer un **événement personnalisé**, utilisez `$.Event ()`. Pour déclencher un événement personnalisé, utilisez `.trigger ()` :

```
var event = $.Event("custom-event");
$("selector").trigger(event);
```

EXEMPLES D'UTILISATION

```
var customEvent = $.Event("customEvent");
$("button").click(function() {
  $("p").trigger(customEvent);
});
```

LES ÉVÉNEMENTS DE LA SOURIS

SYNTAXE

```
$( "selector" ) .event (function() {  
    // code à exécuter  
} );
```

EXEMPLES D'UTILISATION

MOUSEOVER

```
$( "p" ).mouseover(function() {  
    alert("Pointer sur paragraphe");  
});
```

Explication :

- \$ représente le sélecteur en jQuery
- "p" sélectionne tous les éléments <p>
- .mouseover est un événement qui se déclenche lorsqu'on passe le curseur sur l'élément
- La fonction anonyme déclenche une alerte lorsqu'on passe le curseur sur un paragraphe

MOUSEOUT

```
$( "p" ) .mouseout (function() {  
    alert ("Pointer hors du paragraphe");  
}) ;
```

LES ÉVÉNEMENTS DU CLAVIER

SYNTAXE

```
$( "selector" ) .event (function (e) {  
    // code à exécuter  
} );
```

EXEMPLES D'UTILISATION

KEYDOWN

```
$( "input" ).keydown(function() {  
    alert("Touche enfoncée");  
});
```

KEYUP

```
$( "input" ).keyup(function() {  
    alert("Touche relâchée");  
});
```

LES ÉVÉNEMENTS DE FORMULAIRE

SYNTAXE

```
$( "selector" ) .event (function() {  
    // code à exécuter  
} );
```

EXEMPLES D'UTILISATION

SUBMIT

```
$( "form" ).submit(function(e) {  
    e.preventDefault();  
    alert("Formulaire soumis");  
});
```

CHANGE

```
$( "select" ).change(function() {  
    alert("Option sélectionnée modifiée");  
});
```

LES ANIMATIONS

EFFETS DE BASE

FONCTIONS `fadeIn`, `fadeOut`, `fadeTo`

`fadeIn ()`: Affiche progressivement un élément en augmentant son opacité.

`fadeOut ()`: Cache progressivement un élément en diminuant son opacité.

`fadeTo ()`: Modifie l'opacité d'un élément à une valeur spécifiée.

FONCTIONS `slideUp`, `slideDown`, `slideToggle`

`slideUp ()`: Cache un élément en réduisant sa **hauteur**.

`slideDown ()`: Affiche un élément en augmentant sa **hauteur**.

`slideToggle ()`: Alterne entre `slideUp ()` et `slideDown ()`.

EXEMPLES D'UTILISATION

```
$( "#div1" ).fadeIn()
$( "#div2" ).fadeOut()
$( "#div3" ).fadeTo("slow", 0.5)

$( "#div1" ).slideUp()
$( "#div2" ).slideDown()
$( "#div3" ).slideToggle()
```

EFFETS PERSONNALISÉS

SYNTAXE DE LA FONCTION `animate()`

La fonction `animate()` permet de créer des **animations personnalisées** avec plusieurs propriétés **CSS**.

```
$(selector).animate({propriétés}, durée, easing, callback)
```

EXEMPLES D'UTILISATION

```
$( "#div1" ).animate({left: "100px"}) // Déplace vers la droite  
$( "#div2" ).animate({height: "toggle"}) // Alterne la hauteur
```

PROPRIÉTÉS CSS ANIMABLES

Propriété	Description
left, top	Positionnement
width, height	Dimensions
borderWidth	Bordure
marginLeft	Marge gauche
marginRight	Marge droite
fontSize	Taille de la police
lineHeight	Hauteur de ligne

GESTION DES ANIMATIONS

MÉTHODES `stop`, `delay`, `finish`

`stop ()`: Arrête l'**animation** en cours.

`delay ()`: Retarde l'exécution d'une **animation**.

`finish ()`: Arrête **toutes** les animations en cours et les termine immédiatement.

EXEMPLES D'UTILISATION

```
$( "#div1" ).stop();
$( "#div2" ).delay(1000).fadeIn();
$( "#div3" ).finish();
```

- **stop()** : arrête l'animation en cours sur l'élément sélectionné
- **delay()** : retardé l'exécution d'une fonction ou d'une animation
- **fadeIn()** : affiche progressivement un élément en augmentant son opacité
- **finish()** : arrête toutes les animations en cours et les termine immédiatement

CALLBACKS ET CHAÎNAGE

SYNTAXE

Les **callbacks** sont des fonctions qui s'exécutent après la fin d'une animation. Elles permettent d'**enchaîner les animations**.

```
$(selector).animate({propriétés}, durée, easing, callback)
```

EXEMPLES D'UTILISATION

```
$( "#div1" ).fadeIn(1000, function() {
    $( "#div2" ).fadeOut(1000)
})
```

AVANTAGES

- Permet d'**organiser** et de **synchroniser** les animations
- Facilite la **maintenance** et la compréhension du code

