# jQuery

write less, do more.

# jQuery, c'est quoi ?

- Un framework JavaScript
  - sorte d'énorme bibliothèque qui ajoute de nombreuses fonctionnalités au langage et change véritablement la manière de coder ;
  - JQuery est probablement le framework le plus utilisé de JavaScript
- jQuery permet entre autre de :
  - manipuler le DOM (HTML + CSS) ;
  - gérer les événements (clavier, souris, chargement de fichier…) ;
  - gérer l'AJAX (chargement asynchrone  de contenu HTML ou autre) ;
  - éviter les problèmes de compatibilité entre navigateurs.

# Plan

- Le DOM HTML/CSS de JavaScript
- Fonctionnement général de jQuery et sélection des éléments du DOM
- Gestion des événements
- Manipulation du DOM HTML/CSS

# Plan

- Le DOM HTML/CSS de JavaScript
- Fonctionnement général de jQuery et sélection des éléments du DOM
- Gestion des événements
- Manipulation du DOM HTML/CSS

# Le DOM de JavaScript, c'est quoi ?

- Document Object Model => Modèle à objets d'un document (HTML ou XML)
- DOM représente un document HTML ou XML dans l'environnement JavaScript
  - Objets JavaScript reliés selon une structure en arbre
- Le DOM est un « pont » entre l'univers HTML et l'univers JavaScript :
  - Fournit une interface de programmation (normalisée par W3C) pour :
    - examiner le document présent dans le navigateur ;
    - modifier le document présent dans le navigateur ;

# Structure en arbre du DOM

- Chaque objet de l'arbre est appelé **nœud** ;
- Un nœud peut avoir en dessous de lui un ou plusieurs autres nœuds « enfant » *(child)* ;
- Un nœud a au dessus de lui un seul autre nœud « parent » *(parent)* ;
- Les nœuds de mêmes parents sont « frères » *(sibling)*.

# Structure en arbre du DOM

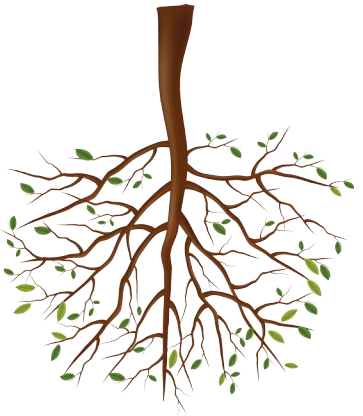
- Dans la représentation du DOM, il y a 4 types de nœuds :
  - Le nœud de type « *Document* » : représente tout le document ;
  - Les nœuds de type « *Element* » : représentent les balises et leurs attributs ;
  - Les nœuds de type « *#text* » : représentent le texte entre les balises ;
  - Les nœuds de type « *#comment* » : représentent les commentaires.

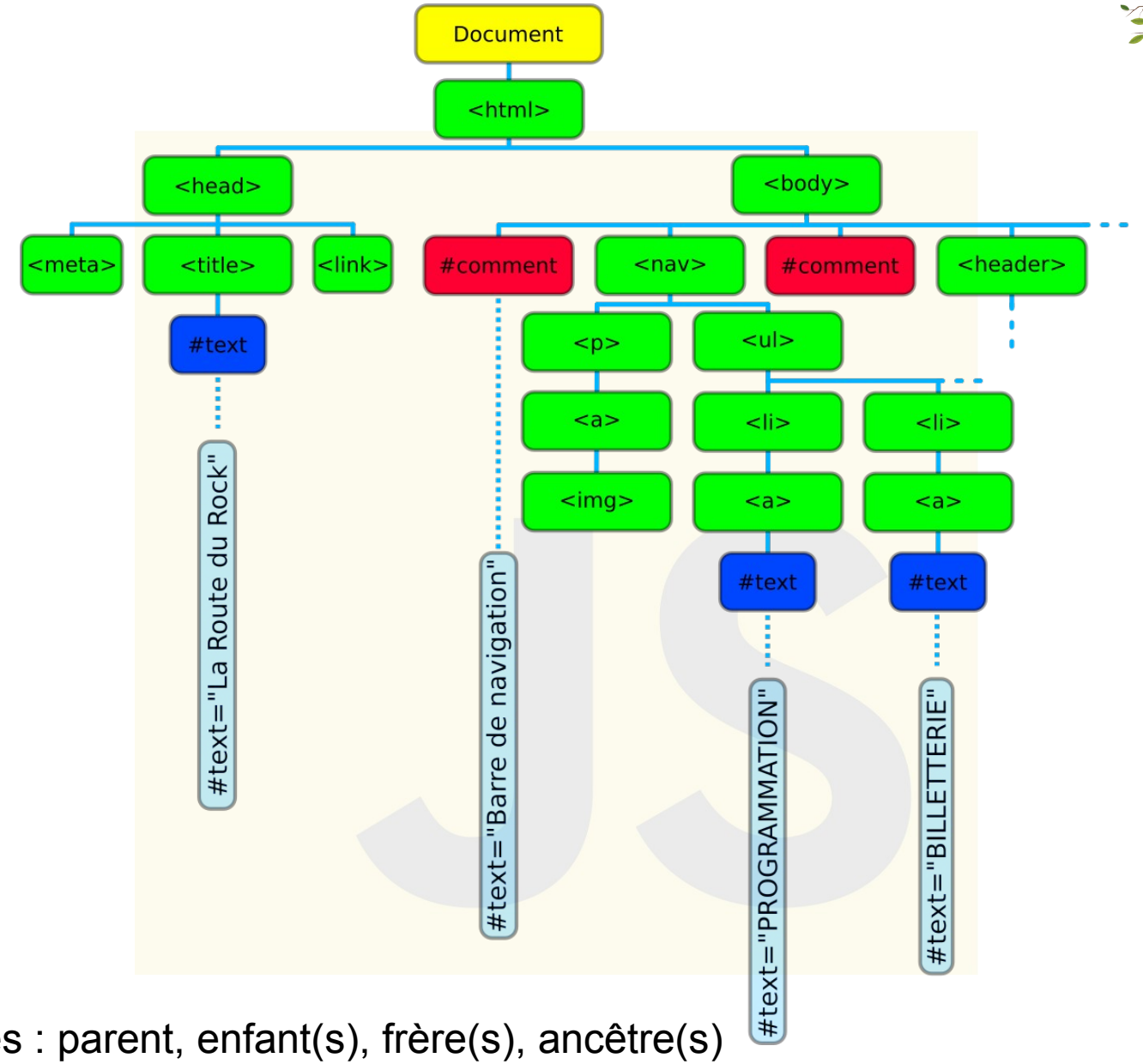
    *Les attributs des balises sont aussi des nœuds (de type « attribut »), mais ne sont pas représentés dans l'arbre DOM.*

# Structure en arbre du DOM

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8" />
        <title>La Route du Rock</title>
        <link rel="stylesheet" href="styles.css" type="text/css" />
    </head>
    <body>
        <!-- Barre de navigation -->
        <nav>
            <p><a href="index.html"><img src="images/logoBlanc.png" alt="Logo de la route du rock"/></a></p>
            <ul>
                <li><a href="#programmation">PROGRAMMATION</a></li>
                <li><a href="#reserver">BILLETTERIE</a></li>
                <li><a href="informationspratiques.html">INFORMATIONS PRATIQUES</a></li>
                <li><a href="presse.html">PRESSE/PROS</a></li>
                <li><a href="shop.html">SHOP</a></li>
            </ul>
        </nav>
        <!-- En-tête -->
        <header class="texteBlancCentre">
            <div>
                <p><img src="images/header.svg" alt="Logo de la route du rock 3 jours"/></p>
            </div>
```

# Structure en arbre du DOM






Exemples : parent, enfant(s), frère(s), ancêtre(s)

# Mise en œuvre du DOM

- Chaque navigateur web met à disposition le DOM ;
  - Objets, méthodes (~fonctions sur les objets), attributs… pour manipuler les éléments HTML avec JavaScript
- Principales catégories d'objets JavaScript disponibles dans le DOM:
  - « `document` » : informations et fonctions générales sur le document HTML ;
  - « `element` » : représente une balise HTML et comporte une propriété « `attributes` » avec l'ensemble des attributs ;

    chaque « `element` » comporte des méthodes permettant de naviguer dans le DOM (atteindre le parent, les enfants, etc.) ;
  - « `text` » : contient le texte à l'intérieur des balises.

# Que peut-on faire sur le DOM avec JavaScript et jQuery ?

- JavaScript et jQuery permettent tous les deux d'accéder à un (ou plusieurs) nœud(s) du DOM et de les utiliser ou les manipuler
  - Lire ou  changer le contenu HTML d'un élément
  - Lire ou changer une propriété CSS d'un élément
  - Créer et insérer de nouveaux éléments
  - Attacher des événements (ex : click de souris) à certains éléments pour déclencher des actions/interactions...
- Mais la syntaxe est plus simple et légère en jQuery
  - Exemples : écriture du contenu HTML d'un élément d'identifiant "titre"
    - En JavaScript :

```
document.getElementById("titre").innerHTML = "Texte de l'élément" ;
```

    - En jQuery :

```
$('#titre').html("Texte de l'élément") ;
```

# Attendre la disponibilité du DOM

- Avant de faire quoi que ce soit au niveau du HTML ou du CSS il faut attendre que l'arbre HTML (DOM) soit entièrement lu
- Cela se fait en utilisant une fonction particulière de jQuery qui va contenir tout le code jQuery/JavaScript :

```
$(function(){
    // Ici, le DOM est entièrement défini
    // On peut y mettre le code de traitement
}) ;
```

- *Remarque : il s'agit en fait d'un raccourci d'écriture pour la fonction :*

```
jQuery(document).ready(function() { ... })
```

# Plan

- Le DOM HTML/CSS de JavaScript
- **Fonctionnement général de jQuery et sélection des éléments du DOM**
- Gestion des événements
- Manipulation du DOM HTML/CSS

# Comment mettre en place jQuery ?

- Pour que jQuery puisse être utilisé dans le navigateur, il faut l'importer dans la page HTML
  - Import local (fichier téléchargé dans le repertoire *js/*)

```
<script src="js/jquery.js"></script>
<script src="js/mon_script.js"></script>
```

  - Import en ligne

```
<script src="http://code.jquery.com/jquery.min.js"></script>
<script src="js/mon_script.js"></script>
```

# Utiliser le DOM avec jQuery

- Principes de base

1. Sélectionner un ou plusieurs éléments du DOM
    - Cette étape renvoie des objets du DOM qui représentent les éléments sélectionnés
2. Utiliser des méthodes sur les objets sélectionnés pour :
    - récupérer/modifier des caractéristiques (attributs, contenu, propriétés CSS, …)
    - insérer/supprimer des éléments dans le DOM
    - Attacher des événements à certains éléments

# Sélectionner des éléments du DOM

- Cela se fait avec la syntaxe :

  `$('sélecteur')`

  - sélecteur reprend la **même syntaxe de sélection que les sélecteurs CSS**
  - Exemples :

    `$('*')` sélectionne tous les éléments ;

    `$('h1')` sélectionne tous les éléments <h1> ;

    `$('#titre')` sélectionne l'élément d'id « titre » ;

    `$('#galerie .vignette')` sélectionne tous les éléments de classe «vignette » contenus dans l'élément d'id « galerie » ;

    `$('h1[title]')` sélectionne tous les éléments <h1> ayant un attribut « title » ;

    `$('img[src = ''images/facebook.svg'']')` sélectionne tous les éléments <img> ayant un attribut « src » qui vaut « images/facebook.svg » ;

    etc...

# Sélectionner des éléments du DOM

- Remarques :
  - L'étape de sélection peut renvoyer **plusieurs éléments**

    Exemple : **$('nav li')** renvoie tous les éléments li contenus dans nav
  - Dans ce cas :
    - les méthodes jQuery de manipulation du HTML et du CSS agissent sur tous les éléments de la sélection
    - les éléments sélectionnés peuvent être récupérés dans un tableau

# Sélectionner des éléments du DOM

- On peut continuer à chercher à l'intérieur d'une sélection avec **.find()** :

  ```
  $('sélection1').find('sélection2')
  ```

  - Cherche les éléments ciblés par *sélection2* **à l'intérieur** des éléments ciblés par *sélection1*

- On peut exclure certains éléments d'une sélection avec **.not()** :

  ```
  $('sélection1').not('sélection2')
  ```

  - **Exclue** les éléments ciblés par *sélection2* de l'ensemble des éléments ciblés par *sélection1*

# Sélectionner des éléments du DOM

- On peut aussi filtrer la liste des éléments renvoyés en fonction de leur place dans la sélection :

  $('nav li:first') cible le premier élément de la liste ;

  $('nav li:last') cible le dernier élément de la liste ;

  $('nav li:eq(N)') cible le N$^{ème}$ élément de la liste (en commençant à l'indice 0);

# Sélectionner des éléments du DOM

- Ou encore parcourir l'arbre DOM :

  Syntaxe :

  $('sélecteur CSS').children() : cible l'ensemble des enfants de l'élément ciblé par le sélecteur CSS ;

  $('sélecteur CSS').parent() : cible le parent de l'élément ciblé par le sélecteur CSS ;

  - Remarque : on peut mettre un autre sélecteur CSS entre les parenthèses de *.children()* et *.parent()* pour ne sélectionner qu'un type d'enfants ou de parent, par exemple *.children('p')* pour sélectionner les éléments enfants de type paragraphe

  $('sélecteur1').closest('sélecteur2') : cible l'ancêtre le plus proche (i.e. en remontant l'arbre DOM) correspondant à *sélecteur2*, du ou des éléments sélectionnés par sélecteur1

  etc.

- Voir la doc en ligne pour les autres méthodes de navigation dans le DOM

# Plan

- Le DOM HTML/CSS de JavaScript
- Fonctionnement général de jQuery et sélection des éléments du DOM
- **Gestion des événements**
- Manipulation du DOM HTML/CSS

# Qu'est-ce qu'un événement ?

- Un événement peut être
  - une action de l'utilisateur.
    - Survol par la souris d'une zone de la page web ;
    - Clique sur un élément, appui sur une touche…
  - la fin d'une action asynchrone
    - Chargement d'un fichier ou  de données en ligne
- Le JavaScript est un langage dit événementiel.
  - Cela permet d'exécuter du code lors d'un événement précis (clic sur un élément, chargement d'un fichier, réponse d'un serveur, etc.) qui survient à un moment non déterminé à l'avance

# Liste des événements (souris)

| Nom de l'événement | Action pour le déclencher |
|---|---|
| click | Cliquer (appuyer puis relâcher) sur l’élément |
| dblclick | Double-cliquer sur l’élément |
| mouseenter | Faire entrer le curseur dans l'élément |
| mouseleave | Faire sortir le curseur de l'élément |
| mousedown | Appuyer (sans relâcher) sur le bouton gauche de la souris sur l’élément |
| mouseup | Relâcher le bouton gauche de la souris sur l’élément |
| mousemove | Faire déplacer le curseur sur l’élément |

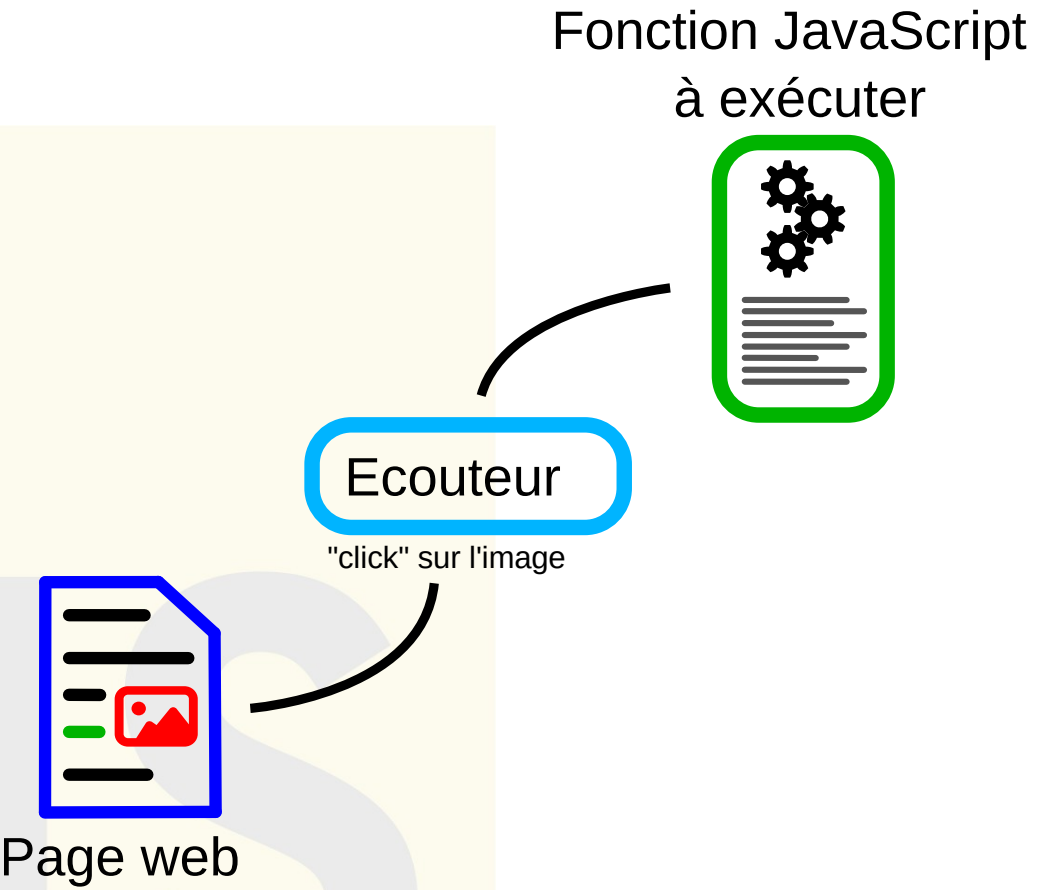
# Liste des événements (clavier)

| Nom de l'événement | Action pour le déclencher |
|---|---|
| keydown | Appuyer (sans relâcher) sur une touche de clavier sur l'élément |
| keyup | Relâcher une touche de clavier sur l'élément |
| keypress | Frapper (appuyer puis relâcher) une touche de clavier sur l'élément |

- Pour retrouver tous les événements qui peuvent être ajoutés au DOM :
  - https://en.wikipedia.org/wiki/DOM_events
  - https://www.w3schools.com/jsref/dom_obj_event.asp

# Gestionnaire d'événement

1. Disposer un écouteur (listener) sur une balise HTML ;
2. Préciser le type d'événement (event) à écouter ;
3. Écrire une fonction JavaScript à exécuter lorsque l'événement est réalisé.



# Les événements en jQuery

- Les écouteurs s'attachent avec la méthode on()
  - On donne
    - en premier paramètre de on(), le **nom de l'événement** à "écouter" (i.e. surveiller)
    - en deuxième paramètre le **nom de la fonction déclenchée** lorsque l'événement est détecté (ou directement le code d'une **fonction anonyme**, cf exemple ci-après)
  - Exemples :

```
$('nav img').on('mouseenter', rotation);

$('nav img').on('mouseleave', annuleRotation);
```

- On peut associer plusieurs événements au même écouteur ainsi que plusieurs éléments
  - Exemple :

```
$('nav img, #reserver img, #reseauxSociaux img').on({'mouseenter': rotation,
'mouseleave': annuleRotation});
```

# Les événements en jQuery

- On peut associer plusieurs écouteurs au même événement
  - Exemple :

```
// avec le nom d'une fonction (à définir ailleurs dans le code)
function retournement(){
...
}
$('#reserver img').on('click', retournement);
// avec le code d'une fonction anonyme (i.e. sans nom)
$('#reserver img').on('click', function(){
    ...
});
```

# Les événements en jQuery

- Les écouteurs se détachent avec la méthode off()
  - Utilisation :

```
$(…).off(); //Retire tout écouteur de l'élément sélectionné

$(…).off('click'); //Retire tout écouteur du click

$(…).off('click',maFonction); //Retire l'écouteur du click qui exécute maFonction
```

  - Exemples :

```
$('#reserver img').on('click', function(){
        $('#reserver img').off('mouseover mouseleave');
});
```

# Variantes de syntaxe des événements en jQuery

- Vous rencontrerez (peut-être) d'autres méthodes générales pour attacher/détacher des écouteurs en jQuery (ces méthodes s'appliquent dans certains contextes) :
  - bind()/unbind
  - live()/die()
  - delegate()/undelegate()
- on() et off() permettent de regrouper l'ensemble des méthodes ci-dessus (plus pratique) ;
- Il est recommandé depuis jQuery 1.7 d'utiliser on() et off()

# Variantes de syntaxe des événements en jQuery

- Il existe plein de raccourcis pour des types d'événement spécifiques :
  - $(…).click();
  - $(…).mouseover();
  - $(…).keypress();
  - $(…).error();
  - …
- Liste complète :
  - https://api.jquery.com/category/events/

# Que peut-on faire dans la fonction déclenchée par un événement ?

- Sélectionner d'autres éléments et agir dessus
  - Sélection avec jQuery : **$('*sélecteur*')**
- Sélectionner l'élément qui a capté l'événement et agir dessus
  - Cet élément est représenté par **$(*this*)** dans la fonction
- Utiliser l'objet "Event"
  - Cet objet peut être récupéré dans la fonction déclenchée et fournit de nombreuses informations sur l'événement déclenché (cf ci-après)

# À quoi sert l'objet événement "Event" ?

- Fournit beaucoup d'informations sur l'événement déclenché ;
- Exemples :
  - Touche actuellement enfoncée ;
  - Coordonnées du curseur ;
  - Élément qui a déclenché l'événement ;
  - …

# Généralités sur l'objet Event

- Accessible uniquement si un événement est déclenché ;
- En général il est noté « e » (mais peut prendre n'importe quel nom de variable) et doit être passé en paramètre de la fonction déclenchée :
  - Exemple :

```
$('#reserver img').on('click', function(e){
    alert('Objet event : ' + e) ;
}) ;
```

- « e » représente l'objet Event de l'événement déclenché.

# Les fonctionnalités de l'objet Event

- Cet objet Event « e » permet :
  - D'accéder au type d'événement avec la propriété

    **e.type**
  - D'accéder à l'élément ayant **émis** l'événement au plus bas niveau du DOM

    **e.target**
    - Ce peut un élément enfant de l'élément qui a **capté** l'événement
  - D'empêcher l'action par défaut (par exemple empêcher un lien de changer de page au clic)

    **e.preventDefault()**

# Les fonctionnalités de l'objet Event

- Mais aussi :
  - D'accéder aux coordonnées du pointeur

    **e.pageX** et **e.pageY** (par rapport à la page)

    **e.screenX** et **e.screenY** (par rapport à l'écran)
  - D'accéder au code de la touche clavier pressée

    **e.which**
  - ...

# Exemple d'affichage d'informations de l'objet Event

- On souhaite afficher les coordonnées du curseur dans la console lorsque la souris bouge sur le « header ».

```
function afficherCoordonnees(e){
    console.log("Coordonnées du pointeur : ("+e.pageX+","+e.pageY+")");
}
$('header').on('mousemove', afficherCoordonnees);
```

# Plan

- Le DOM HTML/CSS de JavaScript
- Fonctionnement général de jQuery et sélection des éléments du DOM
- Gestion des événements
- **Manipulation du DOM HTML/CSS**

# Manipuler le HTML

- Quelques méthodes de base pour récupérer ou ajouter du contenu HTML :
  - **.html()** : récupère ou modifie le contenu HTML de l'élément ciblé.

    Syntaxe :

    ```
    var contenu = $('sélecteur CSS').html() ;   // en lecture
    $('sélecteur CSS').html('Texte à modifié avec <em>du texte un peu en valeur<em>') ;  // en écriture
    ```

  - **.text()** : récupère ou modifie le contenu textuel de l'élément ciblé.

    Syntaxe :

    ```
    var contenu = $('sélecteur CSS').text() ;
    $('sélecteur CSS').text('Texte à modifié sans balise HTML') ;
    ```

# Manipuler le HTML

- **.attr(...)** : récupère ou modifie l'attribut d'un élément.

  Syntaxe :

```
var valeur = $('sélecteur CSS').attr('href') ;  // en lecture
$('sélecteur CSS').attr('href', 'prog3jours.html') ;  // en écriture
```

- **.append()** : Ajoute du contenu HTML ou textuel **à la fin** de l'élément.

  Syntaxe :

```
$('sélecteur CSS').append('Ce contenu HTML est ajouté à la <strong>fin</strong> de la balise');
```

- **.prepend()** : Ajoute du contenu HTML ou textuel **au début** de l'élément.

  Syntaxe :

```
$('sélecteur CSS').prepend('Ce contenu HTML est ajouté au <strong>début</strong> de la balise');
```

# Manipuler le HTML

- **.after()** : Ajoute du contenu HTML ou textuel **après** l'élément sélectionné (au même niveau).

  Syntaxe :

  ```
  $("sélecteur CSS").after("<p>Ce paragraphe est ajouté <strong>après</strong> la balise</p>");
  ```

- **.before()** : Ajoute du contenu HTML ou textuel **avant** l'élément sélectionné (au même niveau).

  Syntaxe :

  ```
  $("sélecteur CSS").before("<p>Ce paragraphe est ajouté <strong>avant</strong> la balise</p>");
  ```

# Manipuler le HTML

- **Pour créer de nouveaux éléments et supprimer des éléments existants**
  - $('<p id="intro">') : renvoie un nouvel élément paragraphe avec l'identifiant «intro»
    - En général on récupère l'élément créé dans une variable pour pouvoir le manipuler avant de l'insérer

      Ex : `var new_p = $('<p id="intro">') ;`
    - Attention, par défaut ce nouvel élément **n'est pas placé dans l'arbre HTML** , il faudra l'ajouter soi-même (avec append(), prepend(), after() ou before() )
  - **.clone()** : duplique un élément existant **sans le placer dans le DOM**.

    Syntaxe : `var mon_clone = $('sélecteur CSS').clone();`
  - **.remove()** : Supprime un élément.

    Syntaxe : `$('sélecteur CSS').remove();`

# Manipuler le HTML

- Quelques méthodes spéciales pour les classes
  - **.hasClass(**'une-classe'**)** : permet de tester si un élément sélectionné possède une classe donnée (renvoie true s'il la possède, false sinon)
  - **.addClass(**'une-classe'**)** : permet d'ajouter une classe à un élément sélectionné
  - **.removeClass(**'une-classe'**)** : permet de retirer une classe à un élément sélectionné
  - **.toggleClass(**'une-classe'**)** : permet d'ajouter une classe à un élément sélectionné s'il ne la possède pas ou de la retirer s'il la possède

# Manipuler le CSS

- Une méthode de base pour modifier des propriétés CSS
  - **.css()** : récupère ou modifie une ou plusieurs propriétés CSS

    Syntaxe :

```
var valeur = $('sélecteur CSS').css('propriété CSS');  // en lecture
// ci-dessous, en écriture :
$('sélecteur CSS').css('propriété CSS', 'valeur de la propriété');
$('sélecteur CSS').css({'propriété 1': 'valeur 1', 'propriété 2': 'valeur 2'});
```

    ***Remarque :*** *sur ce dernier exemple, utilisation d'un objet avec des paires propriété/valeur pour définir un ensemble de propriétés CSS*

# Manipuler le CSS

- Quelques méthodes spéciales
  - **.width()** et **.height()** : récupère ou modifie la largeur/la hauteur en pixels d'un élément, sous forme d'un nombre (utile pour des calculs)
  - **.hide()** et **.show()** : permet de cacher/afficher un élément
    - On peut préciser le temps de la transition en millisecondes
      - Ex : **.hide(**1000**)**
        - Cache un élément en le faisant disparaître (diminution de taille et d'opacité) en 1 seconde

# Sélection multiple : parcours et manipulation des éléments

- Dans le cas où la sélection renvoie plusieurs éléments :
  - Les modifications par les méthodes précédentes sont appliquées à **tous ces éléments** ;
  - On peut aussi appliquer un traitement individuel à chaque éléments de la sélection grâce à la méthode each(). Il faut définir dans each() une **fonction** qui agira sur chaque élément individuellement.

    Syntaxe :

```
$('sélecteur CSS').each( function(index){
        console.log("Traitement de l'élément "+index) ;
        $(this).methodeSurElement();
    });
```

    Dans cette fonction, $(this) permet d'accéder à chaque élément sélectionné (objet jQuery) un par un, et index correspond à son indice (~rang) dans la sélection.

# Mémento des principales fonctions jQuery

- https://openclassrooms.com/courses/un-site-web-dynamique-avec-jquery/memento-des-principales-fonctions