**Preliminaire** : c’est quoi jquery ?

**Intégration de jq dans une page web** : chercher le lien du CDN sur google

<https://code.jquery.com/jquery-3.6.1.min.js>

🡺 le techarger dans le dossier js 🡺 l’intégrer dans la page web

**Le symbole de jQuery** : $

Le symbole de jq est suivi d’un selecteur ou d’un objet

Exp :

console.log($(".para1")); // selecteur

console.log($(document)); // var

**Les sélecteurs de jQuery**

<https://www.w3schools.com/jquery/jquery_ref_selectors.asp>

**$(document).ready ()**

Syntaxe :

$(document).ready (cb) ;

Exp :

$(document).ready(

      function () {

console.log("execution apres chargement du DOM");

       }

 ) ;

**Chainage de fonctions**

Rajouter la ligne

$(".para1").text("cours jquery").css("color", "skyblue").fadeIn("slow");

**la methode css() :** ajoute du style en ligne à un element

Syntaxe : variante 1

$(selecteur).css("prop", "valeur");

$("p").css("background-color", "skyblue");

Variante 2 (avec objet json)

$(selecteur).css({"*propriete*":"*valeur*"," *propriete* ":" *valeur* ",...});

$("p").css({"background-color": "yellow", "font-size": "200%"});

Exo : changer le style d’un element avec la methode css()

**Parcours du DOM**

**La fonction find()** : retourne tous les descendants d’un selecteur

syntaxe

$(selecteur).find(selecteur).fonction ();

Exemple :

<ul>

       <li>

          <span>span petit fils</span>

       </li>

</ul>

**$("ul").find("span").css({"color": "red", "border": "2px solid red"});**

**La fonction children()** : retourne les enfants DIRECTS d’un selecteur

syntaxe

$(selecteur).children (selecteur).fonction ();

Exemple :

$("ul").children("li").css({"color": "blue", "border": "2px solid greenyellow"});

**La fonction parent()** : retourne le parent DIRECT d’un selecteur

syntaxe

$(selecteur).parent().fonction ();

Exemple :

 $("span").parent().css({"color": "red", "border": "2px solid red"});

**La fonction closest()** : retourne le parent le PLUS PROCHE d’un selecteur

syntaxe

$(selecteur).closest(selecteur).fonction ();

Exemple :

$("span").closest("ul").css({"color": "black", "border": "2px solid black"});

**La fonction next()** : retourne le FRERE SUIVANT d’un selecteur

syntaxe

$(selecteur).next().fonction ();

Exemple :

$("li:nth-child(2)").next().css({ color: "green", border: "2px solid green" });

**La fonction prev()** : retourne le FRERE PRECEDENT d’un selecteur

syntaxe

$(selecteur).prev().fonction ();

Exemple :

$("li:nth-child(2)").prev().css({ color: "blue", border: "2px solid blue" });

**La fonction first()** : retourne la 1ERE OCCURENCE d’un selecteur

syntaxe

$(selecteur).first().fonction ();

Exemple :

$("li").first().css({ color: "red", border: "2px solid red" });

**La fonction last()** : retourne la derniere occurence d’un selecteur

syntaxe

$(selecteur).last().fonction ();

Exemple :

$("li").last().css({ color: "yellow", border: "2px solid yellow" });

**La fonction siblings()** : retourne tous les freres d’un selecteur (le selecteur en lui-même non compris)

syntaxe

$(selecteur).siblings().fonction ();

Exemple :

$("li:nth-child(2)").siblings().css({ color: "gray", border: "2px solid gray" });

Liste de toutes les fonctions du parcours du DOM

<https://www.w3schools.com/jquery/jquery_ref_traversing.asp>

**Manipulation du DOM**

Il s’agit d’ajouter, supprimer des elements ou modifier leur contenu

**La fonction text() :** Elle retourne ou modifie le contenu textuel d’un selecteur

recupération

console.log($(".para1").text());

modifier le contenu

$(".para1").text("texte para 1 modifié");

console.log(`texte apres modif : ${$(".para1").text()}`);

**La fonction html()**

Pareil que la fonction text(), mais elle recupère ou modifie le contenu html d’un element

recupération

console.log($("ul").html());

modifier le contenu

$("ul").html(`

<li><a href="https://www.google.com/" target="\_blank">lien vers Google</a></li>

    `);

Expliquer la gestion des evenements + la fonction val()

**Appliquer un evenement à un element**

Syntaxe

$(selecteur).evenement(cb);

NB : la cb admet un paramètre qui represente l’evenement.

Exp :

$(".bc-btn").click(function (e) {

      console.log(" element bc-btn cliqué");

      console.log(e); *// e  est l'evenement : click*

    });

Liste complète des évenements

<https://www.w3schools.com/jsref/dom_obj_event.asp>

**Récupérer et modifier les valeurs des inputs**

Récupération

$(selecteur).val();

Exp :

$("#nom").val()

Modification

$(selecteur).val(nvValeur);

Exp :

$("#nom").val("HIDRI Jr")

**Ex :** créer un input qui demande la capitale de la France, apres deux div ayant les classes reponse et resultat et un bouton submit que lorsqu’on clique dessus teste si la valeur tapée est ‘’paris’’, mettre

‘’Votre réponse est correcte’’ dans la div reponse sinon ‘’Réponse erronée !’’ avec une ecriture et une bordure rouge et dans la zone reponse.

Enfin, on affiche le score 10/10 ou 0/10 dans la div resultat.

$(document).ready(function () {

    $(".dom input[type='submit']").click(function () {

        if ($("#capitale").val().toLowerCase() === "paris") {

            $(".reponse").html(`<span style="color:green">Bonne réponse </span>`);

            $(".resultat").html("10/10");

        } else {

            $(".reponse").html(`<span style="color:red">Mauvaise réponse </span>`);

            $(".resultat").html("0/10");

        }

        $("#capitale").val("");

    });

});

**Ajout d’elements au DOM**

**La fonction prepend :** ajoute du contenu au debut du selecteur (à l’intérieur et en premier lieu)

<p class="extensible">para extensible</p>

$(".extensible").prepend("contenu d'extension au debut<br/>");

**La fonction append()** : ajoute du contenu à la fin du selecteur (à l’intérieur et en dernier lieu)

$(".extensible").append("<br/>contenu d'extension à la fin");

**La fonction before()** : ajoute du contenu avant le selecteur (à l’extérieur et avant )

$(".extensible").before("<p>para externe avant</p>");

**La fonction after()** : ajoute du contenu apres le selecteur (à l’extérieur et apres)

$(".extensible").after("<p>para externe apres</p>");

**Suppression des elements**

**La fonction remove()** : supprime un element du DOM

<p class="supprimer">Lorem ipsum dolor quidem Aut.</p>

$(".supprimer").remove();

**Acces et modification des attributs**

Pour accéder et pour modifier les attributs d’un element, on utilise la fonction attr()

Pour recupérer la valeur de l’attribut, on le lui passe en param

<a href="https://www.google.com/">lien vers google</a>

console.log($("a").attr("href"));

pour modifier sa valeur, on lui passe deux param, le premier est le nom de l’attribut, et le 2eme la nouvelle valeur.

$("a").attr("href", "https://www.w3schools.com/jquery/default.asp")

console.log($("a").attr("href"));

**NB :** il est possible d’ajouter des attributs en utilisant la meme syntaxe (si l’attribut n’existe pas, on l’ajoute).

$("a").attr("target", "\_blank")

console.log($("a").attr("target"));

**la fonction data() :** elle permet de récupérer ou de modifier les attributs data-

$(selecteur).data("suffixe");*// lecture*

$(selecteur).data("suffixe", "nvValeur");*// modification*

Exp :

<div class='div-data' data-role='user' data-info='{"nom":"HIDRI", "prenom":"Ryan"}' ></div>

console.log($(".div-data").data("role"));

console.log($(".div-data").data("info").nom);

**NB :** la fonction attr est valable aussi pour les attributs data, par contre, il faut mentionner le nom complet de l’attribut

console.log($(".div-data").data("role"));

console.log($(".div-data").attr("data-role"));

**NB :** dans le cas où on utilise un objet json comme valeur pour un attribut data, on privilégie la fonction data (l’utilisation de attr est possible, mais elle le voit comme une chaine de caractères, par conséquent, on ne récupère pas la structure json)

console.log($(".div-data").attr("data-info")); *// chaine*

console.log($(".div-data").data("info")); *// objet*

Ex : changer le role de user à admin, puis afficher la nouvelle valeur

$(".div-data").data("role", "admin")

Ou bien

$(".div-data").attr("data-role", "admin")

console.log($(".div-data").data("role"));

console.log($(".div-data"). attr ("data-role "));

Ex : changer le prenom de Ryan à Ryan Junior

let info = $(".div-data").data("info");

info.prenom = "Ryan junior";

$(".div-data").data("info", info);

console.log($(".div-data").data("info").prenom);

**Rq :** privilégier data au lieu de attr pour que l’attribut soit considéré comme objet lors de la récupération.

**Autres fonctions des attributs**

**removeAttr ()** : supprime un attribut

Dans le html, ajouter au lien une classe

class="lien-supp"

$(".lien-supp").removeAttr("class")

console.log($(".lien-supp "));

// commenter la ligne de removeAttr

**hasClass()** : retourne true si l’element a la classe passée en param, false sinon

$(selecteur).hasClass("classe");

Exp :

$(".btn-style").prev().hasClass("para-style"); *// true*

$(".div-eleve").hasClass("recherche"); *// false*

ex : tester si le lien a la classe  "lien-supp"

tester si le lien a la classe  "lien-autre"

console.log( $("body > a").hasClass("lien-supp") )  *// true*

console.log( $("body > a").hasClass("lien-autre") ) *// false*

**addClass** : ajoute une classe à un element

ajouter la classe "lien-autre"

$("a").addClass("lien-autre") ;

console.log($("a"));

**removeClass()** : supprime une classe

supprimer la classe "lien-autre"

$("a").removeClass("lien-autre");

console.log($("a"));

**toggleClass()** : alterne entre l’ajout et la suppression d’une classe

créer un bouton

<button>Toggle class</button>

Créer une classe CSS "para-rouge"

$(".toggleBtn").click(function () {

   $("p").toggleClass("para-rouge");

});

Ex : créer un paragraphe qui a comme texte ‘’para non cliqué’’ et qui a comme style : couleur et bordure bleu ciel.

Créer un bouton que lorsqu’on clique dessus, il change le texte du paragraphe en ‘’para cliqué’’ change le style en couleur et bordure grise.

Et si on clique dessus encore une fois, il reprend son style initial.

Appliquer les styles avec des classes dans le html

<p class="para-style">para non cliqué</p>

<button class="btn-style">style du paragraphe</button>

.para-style {

   color: skyblue;

   border: 1px solid skyblue;

}

.para-style-clique {

color: gray;

       border: 1px solid gray;

}

$(".btn-style").click(function () {

  $(".para-style").text("para cliqué").toggleClass("para-style-clique ");

});

**Parcours des elements (tableau) :** permet d’exécuter une cb sur une collection d’elements qui répondent à un selecteur (un par un, un a chaque fois). La cb admet 2 param : l’index et la valeur de l’element (à l’intérieur de la fonction cb, on utilise $(this) pour référencer le param element).

$(selector).each(cb);

Exp :

$("p").each(

  function (index, element) {

    let elemClass = $(element).attr("class");

    console.log(elemClass);

    console.log("index", index);

  }

);

Ex : créer un bouton que lorsqu’on clique dessus, il ajoute la classe ‘’bg-bc’’ aux li (bg-bc donne un style de couleur et arriere plan bleu ciel), ensuite il met dans l’element en question ‘’each : li--xx’’ où xx est le  numéro de l’element.

.bg-bc {

                background: skyblue;

                color: white;

                border: 1px solid skyblue;

            }

<button class="bc-btn">bc-btn</button>

$(".bc-btn").click(function () {

  $("li").addClass("bg-bc");

  $("li").each(

    function (index, element) {

      $(element).text(`each : li--${index + 1}`)

    }

  );

})

**Afficher et masquer des elements :** show() et hide()

<button class="masquer-info-btn">masquer info</button>

<button class="afficher-info-btn">afficher info</button>

$(".masquer-info-btn").click(function () {

$(".div-data").hide();

});

$(".afficher-info-btn").click(function () {

$(".div-data").show();

});

**NB :** Les fonctions show et hide admettent des param optionnels

$(*selecteur*).hide(*vitesse,cb*);

Vitesse : vitesse en ms

Cb : fonction callback

**La fonction toggle** : alterne entre afficher et masquer l’element sélectionné

$(*selecteur*).toggle(*vitesse,cb*) ;

**Autres effets de jq**

<https://www.w3schools.com/jquery/jquery_fade.asp>

<https://www.w3schools.com/jquery/jquery_slide.asp>

**La gestion des evenements :** exécuter une action lorsqu’un evenement se produit sur un element

$(selecteur).evenement(cb);

Liste complete des evenements jq

<https://www.w3schools.com/jquery/jquery_ref_events.asp>

**Exécuter un evenement une seule fois**

$("ul").one("mouseenter", function () {

    console.log("message une seule fois");

});

Variante 2 : la cb s’exécute une seule fois pour CHACUN des evenements

$("ul").one("click mouseleave", function () {

    console.log("message une seule fois");

});

**Empecher le comportement par defaut d’un element**

$(".no-link").click(function (e) {

    e.preventDefault();

    console.log("pas de redirection");

})

**Declencher un evenement (artificiel)**

<a class="trigger-link" href="https://www.google.com/"> redirection artifielle</a>

<button class="trigger-btn">trigger bouton</button>

$(".trigger-btn").click(function () {

    console.log("avant");

    $(".trigger-link").trigger("click");

    console.log("apres");

});

$(".trigger-link").click(function (e) {

    console.log("lien cliqué");

    console.log(e.type);

});

**la methode on() :** Avec la methode on(), on peut rattacher à un seul element un ou plusieurs evenements et une seule action, ou bien une action par evenement.

$(selecteur).on("evenement", cb);

Exp 1:

$(".maj-form-btn").on("click", function () {

    console.log("evenement click seul");

})

Exp 2 :

$("ul").on("mouseenter mouseleave click", function () {

console.log("survolé ou quitté la ul");

});

exp 3 : la mathode on() admet comme param un json.

$("ul").on(

{

        mouseenter: function () {

          console.log("survolé  la ul");

        },

        mouseleave: function () {

          console.log(" quitté la ul");

         },

  click: function () {

      console.log("clic sur la ul")

    }

}

);

**Les formulaires :**

val() : Récupère ou met à jour la valeur d’un champ de formulaire

$(selecteur).val(); *// recupere la valeur d’un element de form*

$(selecteur).val(nvValeur); *// met à jour la valeur d’un element de form*

Exp :

<form action="test.php" method="get">

<input type="text" name="nom" id="nom" />

<input type="text" name="prenom" id="prenom" />

       <input type="submit" value="Valider" />

</form>

$("body>form").submit(function (e) {

  e.preventDefault();

*// let nom = $("#nom").val();*

*// let prenom = $("#prenom").val();*

*// console.log(`nom : ${nom}, prenom : ${prenom}`);*

  console.log(`nom : ${   $("#nom").val()   }, prenom : ${  $("#prenom").val()  }`);

});

Ex : mettre à jour les valeurs du nom et du prenom suite à un click sur le bouton ‘’maj-form-btn’’

<button class="maj-form-btn">maj champ form</button>

$(".maj-form-btn").click(function () {

    $("#prenom").val("Ryan");

$("#nom").val("HIDRI");

})

EX : generateur de mots clés pour le referencement

**Ajax**

Il est important de noter que l’utilisation d’ajax connait plusieurs methodes (XMLHttpRequest, avec jQuery, la bib axios …)

Squelette de base pour une requete ajax :

let request = $.ajax({

                type: "POST",

                url: "controller.php",

                data: Datas,

                dataType: "json",

                timeout: 120000, *//2 Minutes*

                cache: false,

                contentType: false,

                processData: false,

                beforeSend: function () {

*//Code à jouer avant l'appel ajax en lui même*

                },

            });

            request.done(function (output\_success) {

*//Code à jouer en cas d'éxécution sans erreur du script du PHP*

                alert(output\_success.output);

            });

            request.fail(function (http\_error) {

*//Code à jouer en cas d'éxécution en erreur du script du PHP*

                let server\_msg = http\_error.responseText;

                let code = http\_error.status;

                let code\_label = http\_error.statusText;

                alert("Erreur " + code + " (" + code\_label + ") : " + server\_msg);

            });

            request.always(function () {

*//Code à jouer après done OU fail dans tous les cas*

            });

Type : type de la requete (verbe HTTP)

url : cible de la requete

data : donnés à envoyer

dataType : type de données qu’on reçoit du serveur

timeout : temps d’exécution

contentType : type de données qu’on envoie au serveur

processData : gestion de la serialisation (Lorsque data est un objet, jQuery génère la chaîne de données à partir des paires clé/valeur de l'objet, sauf si l'option processData est définie sur false. Par exemple, { a: 'bc', d: 'e,f' } est converti en la chaîne 'a=bc&d=e%2Cf'.)

cette option est pratiquement non utilisée, on garde le comportement par defaut.

**Ex :** les todos via l’api de placeholder

**API Rest avec json server et jQuery**

Créer un dossier api 🡺 dans api, on crée deux dossiers back et front

Creation du back

Aller à la page du depot de json server sur GitHub

<https://github.com/typicode/json-server>

cliquer sur Getting started 🡺 exécuter la commande suivante pour installer json server via NPM

npm install -g json-server

on se positionne dans le dossier back 🡺 on crée un fichier db.json 🡺 dans ce dernier, on copie le contenu json sur la page de GitHub 🡺 lancer la console 🡺 se positionner dans le dossier back 🡺 lancer le serveur node

json-server --watch db.json

tester le serveur en lançant cette URL sur le navigateur

<http://localhost:3000/posts>

**NB :** il ne faut pas fermer la fenetre de la console

🡺 supprimer tt le contenu de db.json 🡺 definir la structure de nos objets et mettre des enregistrements de test

**Projet gestion de contacts**

Créer un nv projet 🡺 télécharger la template de dashboard via le lien suivant

<https://github.com/twbs/bootstrap/releases/download/v4.6.1/bootstrap-4.6.1-examples.zip>

faire les ajustements nécessaires au niveau du design 🡺