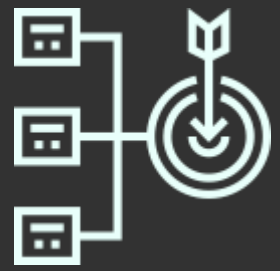
JAVASCRIPT
PRÉSENTATION, JQUERY, AJAX – **PARTIE 1**

Jérôme GRASSET

OBJECTIFS DU COURS

- Voir les bases de l'utilisation de javascript dans un contexte de développement web où il est le langage « côté client »
- Utiliser JQuery, en particulier pour accéder au DOM et réaliser des requêtes asynchrones



Comme précédemment : la présentation part du principe que vous savez déjà programmer dans un langage à syntaxe similaire à Java.

INTRODUCTION

C'EST QUOI ?

- Pour nous : un langage utilisé côté client, interprété par le navigateur web.
- D'une façon plus générale :
  - à l'origine, un langage de script
  - a évolué pour devenir un langage à part entière
  - existe en versions « hors navigateur », par exemple pour utilisation côté serveur (node JS)

PETIT HISTORIQUE

- Créé en 1995 par Brendan Eich, pour la société éditrice du navigateur Netscape
- Standardisé en 1997 : ECMAScript, ECMA-262
- Evolutions régulières, amplifiées depuis 2008 et l'arrivée de Chrome
- Différentes implémentations du standard, chaque navigateur propose sa version

Remarque : malgré le nom, aucun rapport direct entre JavaScript et Java.

ARCHITECTURE

Pour nous c'est un langage « côté client » :

- Le code est stocké côté serveur, comme tous nos fichiers
- Le serveur web (apache) envoie le code source au client sans y toucher
- Le navigateur le reçoit et l'interprète (l'exécute)

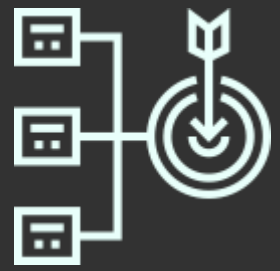
Remarque :

- on peut donc exécuter sans passer par le serveur, en les demandant à l'OS (Windows par exemple)
- mais pour des scripts qui doivent communiquer avec des serveurs il vaut mieux demander les fichiers au serveur Apache (ou autre)

BASES DU LANGAGE

POUR QUOI FAIRE ?

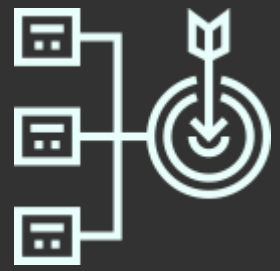
- Ajouter de l'interactivité sur une page web
- Assurer des traitements côté client :
  - calculs
  - contrôles
  - communications avec le serveur « en arrière plan »
  - …



OU ÉCRIRE LE CODE

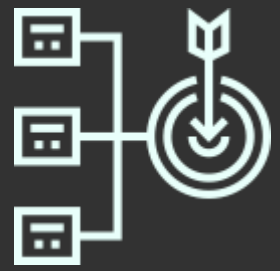
- Dans un document HTML : balise <script>, souvent dans body
- Dans un fichier indépendant avec extension js par convention, puis inclusion dans un document HTML en écrivant (dans head par exemple) :

```
<script src='monScript.js'></script>
```



OÙ EST LE POINT D'ENTRÉE DU CODE ?

- En programmation « classique » (Java, C#, C…) :
  - une fonction est le point d'entrée dans le programme (ex : *Main*)
  - on écrit le code « à partir de là »
- En JS :
  - on peut écrire des scripts sans créer de fonction (comme en php)
  - on attend souvent que le document html soit chargé pour démarrer l'exécution: cela assure que les éléments du DOM existent avant d'être manipulés.



EXEMPLE MINIMALISTE

JS_01_exemple_mini.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>exemple JS 01</title>
    </head>
    <body>
        <h1>Exemple JS...</h1>
        <p> Ouvrez la console pour voir le résultat.</p>

        <script>
            console.log("Yop yop JS.");
        </script>

    </body>
</html>
```

EXEMPLE MINIMALISTE AVEC ATTENTE DE FIN DE CHARGEMENT, V1

Ajout d'une attente de fin de chargement :

Version 1, utilisation de window.onload et d'une fonction anonyme

JS_02_attente_mini_V1.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>exemple JS 01</title>
    </head>
    <body>
        <h1>Exemple JS...</h1>
        <p> Ouvrez la console pour voir le résultat.</p>

        <script>
            window.onload = function()
            {
                console.log("Yop yop JS.");
            }
        </script>

    </body>
</html>
```

EXEMPLE MINIMALISTE AVEC ATTENTE DE FIN DE CHARGEMENT, V2

Ajout d'une attente de fin de chargement :

Version 2, utilisation de l'évènement HTML onload et d'une fonction

JS_03_attente_mini_V2.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>exemple JS 01</title>
    </head>

    <body  onload="init()"  >


        <h1>Exemple JS...</h1>
        <p> Ouvrez la console pour voir le résultat.</p>

        <script>
            function init()
            {
                console.log("Yop yop JS.");
            }
        </script>

    </body>
</html>
```

SYNTAXE GÉNÉRALE : COMME JAVA, C, PHP, ETC.

- On retrouve, par exemple :
  - Le point-virgule à la fin de chaque instruction
  - Les boucles classiques : `for`, `do...while`, et `while`
  - Les conditionnelles : `if... else...`
  - Les commentaires avec `/* ... */` et `//...`
  - La sensibilité à la casse (majuscules et minuscules ne sont pas interchangeables)

SYNTAXE GÉNÉRALE : QUELQUES PARTICULARITÉS

- On peut utiliser les caractères unicodes (dont les accents par exemple) : mieux de s'en tenir aux caractères non accentués anglais.
- Les points virgules ne sont pas nécessaires s'il n'y a qu'une instruction sur une ligne : mieux de les mettre.
- On peut définir des fonctions anonymes
- On peut créer une fonction à l'intérieur d'une fonction

VARIABLES ET TYPES

- Les noms de variables sont créés de façon similaires à Java
- Javascript est faiblement typé :
  - pas d'obligation de déclaration explicite de variables
  - possibilité d'affecter un contenu d'un certain type dans une variable qui contenait un contenu d'un autre type
- Déclaration de variables :
  - mot clé var : variable locale ou globale selon le lieu de définition
  - mot clé let (depuis 2015): variable dont la portée est le bloc courant
  - mot clé const : constante dont la portée est le bloc courant
  - pas de mot clé, affection : variable globale (mauvaise pratique)

TYPES DE DONNÉES

- Types existants :
  - numérique (entier ou décimal) : `number`
  - chaine de caractères : `string`
  - booléen : `boolean`
  - `null` est une « valeur » particulière
  - `undefined` est l'état d'une variable déclarée mais non initialisée

  - objet : `object` (ressemble à une instance de classe, mais n'en est pas une…)

TYPES NUMÉRIQUES : OPÉRATEURS

- Comme en php.
- Opérateurs arithmétiques comme en Java ou C : `+`, `*`, `/`, `-`, `%`
- a puissance b : `a ** b`
- Opérateurs abrégés : `++`, `--`, `+=`, `-=`, `*=`, `/=`

- Opérateurs de comparaison comme en Java ou C : `<`, `>`, `>=`, `<=`, `==`, `!=`
- Opérateurs de comparaison avec vérification du type :
  - `===` : valeurs égales et même types
  - `!==` : valeurs différentes ou types différents

TYPE CHAINE DE CARACTÈRES

- Notation entre simples ou doubles quotes :
  - Équivalent
  - Mais : en JSON ce sont des doubles quotes (on y reviendra plus tard)
- Opérateur de concaténation : **+**

TYPE BOOLÉEN ET OPÉRATEURS LOGIQUES

- Similaire à ce que vous connaissez en Java !

OBJET ARRAY

- Pas un type tableau à strictement parler : un objet.
- Des méthodes et des propriétés pour l'utiliser, par exemple :
  - `monTab.length`
  - `monTab.foreach(function(elemCourant){...})` pour les parcours
  - `monTab.sort()`
  - etc…
- A consulter : 06_array.js

LIEN AVEC LE DOCUMENT HTML

JavaScript et JQuery

EVENEMENTS HTML

- Le document émet des évènements, qui permettent de lancer des traitements en JavaScript.
- Un évènement rencontré précédemment : onload
- Autres (non exhaustif) :
  - onclick
  - onchange
  - onmouseover

  - Liste exhaustive : https://www.w3schools.com/jsref/dom_obj_event.asp

DOM : DOCUMENT OBJECT MODEL

- JavaScript permet de manipuler le DOM :
  - modifier les éléments HTML : contenu, attributs, styles…
  - récupérer des informations dans un élément HTML
  - créer ou supprimer des éléments HTML

- Le DOM est accessible via l'objet `document` qui dispose de propriétés et de méthodes.

ACCÈS À UN ÉLÉMENT

Exemples d'accès à un élément ou à une liste d'éléments :

- `document.getElementById(unId)`
- `document.getElementsByClassName(`*`uneClasse`*`)`

- `innerHTML` : propriété qui représente le contenu d'un élément HTML

```
<script>
    document.getElementById("message").innerHTML = "Coincoin fait le chat.";
</script>
```

ACCÈS À UN ÉLÉMENT : JQUERY

- Une bibliothèque très répandue pour faciliter différentes tâches dont l'accès au DOM : JQuery
- Pour inclure la librairie :
  - Téléchargement et stockage dans votre site (jquery-3.4.1.min.js)
  - Ou lien vers un CDN
- A inclure avant les scripts qui l'utilisent

```
<head>
    <meta charset="utf-8">
    <title>exemple JS générique</title>
    <script src='js/jquery-3.4.1.min.js'></script>
    <script src='js/07_jquery_base.js'></script>
</head>
```

Extrait de :
JS_07_cadre_jquery.html

ACCÈS À UN ÉLÉMENT : JQUERY

- Syntaxe générale : `$(unSelecteur).uneMéthode()`

- `$` : fonction jQuery (aucun rapport avec PHP)
- sélecteur : idem CSS

- Exemples :
  - `$("#message").html("<strong>YOUPI !</strong>, j'ai cours de Dev Web...");`
  - `$(".elemMasquable").hide();`

- Voir plus loin dans le support pour un complément.

JAVASCRIPT ET JQUERY

- Pour approfondir :
  - Compléments en TP
  - w3schools : https://www.w3schools.com/jquery/default.asp
  - Doc officielle : https://api.jquery.com/

- En cours : focalisation sur ajax

REQUÊTES ASYNCHRONES - AJAX

REQUÊTE ASYNCHRONE

- Objectif : adresser une requête HTTP au serveur web via javascript, pour communiquer sans recharger la page
- Principe :
  - JS adresse une requête au serveur
  - Le serveur reçoit et traite la requête : il adresse une réponse
  - JS reçoit la réponse et l'utilise

  - La page web reste totalement « active » pendant toute la durée du processus : c'est l'aspect asynchrone

AJAX : Asynchronous JavaScript And XML. Par « abus de langage » : une requête HTTP asynchrone, même si elle n'utilise pas XML pour formater les données échangées.

REQUÊTE ASYNCHRONE

- Exemples élémentaires :

Appel AJAX et traitement réponse :
Extrait de 08_ajax.js

```
function traiterClicB1()
{

    $.get("reponseServeur.php",traiterReponseDemande01);
}

function traiterReponseDemande01(donnees)
{
    console.log(donnees);

    $("#unMessage").html(donnees);
}
```

reponseServeur.php

```
<?php

    $laDate = new DateTime();

    echo $laDate->format('H:i:s');
```

REQUÊTE ASYNCHRONE

- Possibilité de réaliser l'appel en JS sans JQuery
- Avec JQuery, 3 fonctions :
  - `$.get`
  - `$.post`
  - Et `$.ajax` qui est plus générale
- Possibilité de traiter les erreurs (par exemple script non trouvé), voir par exemple : https://api.jquery.com/jquery.ajax/

- Le X de AJAX est mis pour XML, un format de structuration de données texte.
- En pratique, on utilise souvent JSON, qui est facile à utiliser en JavaScript, dont il provient.

REQUÊTE ASYNCHRONE – RÉPONSE JSON

JSON ?

Un format de données, basé sur JavaScript.

Exemple (https://www.w3schools.com/whatis/whatis_json.asp) :

```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```

REQUÊTE ASYNCHRONE – RÉPONSE JSON

Exemple à consulter : traitement du bouton 3 de JS_o8_ajax.html, avec script JS o8_ajax.js et script php réponseServeurJSON.php

- Principe, simple :
  - On réalise une requête asynchrone
  - Le serveur répond en envoyant des données formatées en JSON

- Mise en œuvre, simple également :
  - On indique que le script php retourne du JSON en le commençant par :

```
header('Content-Type: application/json’);
```

  - On encode en JSON ce qui est envoyé, en php on peut écrire :

```
echo(json_encode($tableau3));
```

REQUÊTE ASYNCHRONE – RÉPONSE JSON

- Mise en œuvre, suite, traitement de la réponse en JavaScript :
  - Les données reçues sont encodées en JSON : on peut voir la structure avec un console.log
  - Dans le cas d'un « tableau » : accès aux données via les indices ou les noms
  - Possibilité d'utiliser un for, similaire au foreach de PHP :

```
for(var cle in donnees)
{
   console.log("case :"+cle + ", données : "+ donnees[cle]);
}
```

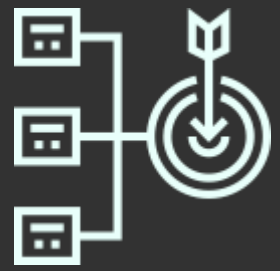
REQUÊTE ASYNCHRONE – USAGES

- Mettre à jour une page web sans la recharger complètement
- Echanger des données avec le serveur « en arrière plan », par exemple pour sauvegarder régulièrement les données saisies dans un formulaire long avant qu'il ne soit validé
- Se connecter à de nombreuses API disponibles en ligne
- …

JAVASCRIPT
SURVOL DE FONCTIONNALITÉS – **PARTIE 2**

OBJECTIFS DU COURS

- Partie 2 du cours :
  - Survol de fonctions JS utiles (ou JQ)
  - Quelques exemples d'utilisation de JavaScript



MANIPULATION DU DOM

- Modifier
- Ajouter
- Supprimer
- Cacher / Afficher

- Possible en JavaScript « pur » : présentation dans le cours en JQuery

MANIPULATION DU DOM : JQUERY

- `$(unSelecteur).uneMéthode()`

- Méthodes utiles pour lire ou écrire dans le dom:
  - html() : élément HTML (balises et contenu)
  - text() : contenu texte d'un élément HTML
  - val() : valeur d'un input (attribut value)

- Méthodes pour cacher ou afficher
  - hide()
  - show()
  - toggle() : « bascule » entre cacher et afficher

MANIPULATION DU DOM : JQUERY

- Méthodes utiles pour ajouter et supprimer du contenu:
  - remove() : suppression de l'élément
  - empty() : suppression du contenu de l'élément
  - Ajouter, en précisant où :
    - append(), prepend() : ajout avant ou après le contenu de l'élément sélectionné
    - after(), before() : ajout avant ou après l'élément sélectionné

- Méthodes pour cacher ou afficher
  - hide()
  - show()
  - toggle() : « bascule » entre cacher et afficher

MANIPULATION DU DOM : JQUERY

- Méthodes utiles pour modifier les classes :
  - addClass(maClasse) : ajout de la classe
  - removeClass(maClasse) : suppression de la classe
  - toggleClass(maClasse) : bascule entre suppression et ajout

- Pour gérer directement des styles css (attribut style), méthode css() :
  - Écriture : css("propriété", "valeur")
  - Lecture : css("propriété")

MANIPULATION DU DOM : JQUERY

Méthodes d'accès à un attribut (y compris id) :

- attr("nomAttribut")
- prop("nomPropriete")

MANIPULATION DU DOM : JQUERY

- Méthodes utiles pour le parcours du DOM
  - parent() : parent dans le DOM
  - parents() : liste des ancêtres, jusqu'à la racine

  - Exemple de parcours d'une liste avec Jquery :

```
$("#youpi").parents().each(function(){
    // code de la function à appliquer à chaque ancêtre
    console.log("Contenu de "+$(this).attr("id"));
    console.log($(this).html());
});
```

MANIPULATION DU DOM : JQUERY

- Méthodes utiles pour le parcours du DOM (suite)
  - children() : liste des enfants
  - find("selecteurRecherché") : liste des sélecteurs correspondant parmi tous les descendants (fils, petits-fils, etc.)
  - siblings() : liste des « frères », éléments qui ont le même parent
  - first() et last() : accès au premier ou dernier élément d'une liste, utile pour les méthodes « children() », « parents() », etc.

MANIPULATION DU DOM : JQUERY

- Evènements : des fonctions pour accéder aux évènements et définir les actions qui répondent aux évènements

```
$("#btnToggle").click(function(){
    // code de “reaction” au clic

    $("#msgInfo").toggle();
});
```

JSON : STRUCTURATION DE DONNÉES

- JavaScript Object Notation :
  - un format pour des données structurées
  - représenté sous la forme d'une chaine de caractères
  - notation structurée comme un « objet » javascript : accès avec une notation point ou crochet, au choix.

- Exemple : https://www.w3schools.com/js/js_json_intro.asp

- Exemple d'utilisation : requête « AJAX » qui nécessitent une communication de données structurées (sera revu en TP)

CANVAS : ZONE DE DESSIN

- Canvas (« canevas ») :
  - Une balise HTML
  - Une zone dans laquelle on peut dessiner en javaScript
  - 2D/3D (WebGL)

- Un élément particulier, qui nécessite un apprentissage à part entière

- Utilisé par certaines librairies

EXEMPLES D'UTILISATION DE JAVASCRIPT

- Requêtes HTTP asynchrones (« AJAX »)

- Validation de formulaires HTML :
  - ajout de contrôles au delà de ce que permet HTML 5, par exemple pour les validations entre 2 champs.
  - La méthode javascript setCustomValidity() permet de réaliser un affichage personnalisé dans le même

https://developer.mozilla.org/fr/docs/Web/API/HTMLSelectElement/setCustomValidity

EXEMPLES D'UTILISATION DE JAVASCRIPT

- Graphiques interactifs
- Cartes interactives
- Animations, jeux, 2D ou 3D
- …

QUELQUES COMPOSANTS JAVASCRIPT DE BOOTSTRAP

- Composants qui utilisent JS sans nécessité d'écrire du code pour le développeur :
  - Tooltips
  - NavBar

- Composants qui se contrôlent en écrivant du code JS :
  - Carousel
  - Modal

BIBLIOTHÈQUES JS (CONTEXTE WEB)

- Nombreuses, parfois très utilisées :
  - JQuery
  - Chart.js : création de graphes
  - D3.js : création de graphes
  - Parsley.js : validation de formulaires
  - Three.js : WebGL (3D)
  - …

FRAMEWORKS JS

- Framework : un cadre de travail, pas uniquement une série de fonctions
- Exemples :
  - React
  - Angular
  - Vue
  - Ionic
  - Ember
  - …

EN GUISE DE CONCLUSION

- Un langage riche
- Très employé
- Des approches parfois différentes de vos habitudes : un modèle orienté objet qui n'est pas basé sur des classes par exemple, non abordé dans ce cours
- ~~Vu~~ survolé ici en tant que langage côté client dans un contexte web, existe dans d'autres contextes