**JavaScript**

1. **Variable et Operateur**

Pour déclarer une variable on utilise :

*let* prenom1 = "mark";

*var* prenom2 = "mark";

* **Var :** agir pour une variable globale
* **let :** agir pour une variable locale
* **const :** variable qui ne peut pas être modifie

locale et globale a une **boucle.**

**La concaténation** permet d’ajouter des variable entre elle. Elle se fait avec **+**

**Exple : let variable1 = ‘’je m’appelle ’’ + prenom1 ;**

Les 03 types de **boite dialogue** :

* affichage du text

console.log("je suis un text");

alert("hello word");

* boite dialogue de confirmation

confirm("je confirme");

* boite de saisir des valeurs

prompt("Entrez une Valeur");

1. **Fonctions**

* **function nom\_fonction(…){ ….} :** fonction sans paramètre
* **function nom\_fonction(paramètres){ ….} :** fonction avec paramètres
* **function nom\_fonction(param=valeur){ ….} :** fonction avec valeur par défaut
* **function (){…} :**  fonction anonyme
  + **variable() :** pour exécuter cette fonction anonyme
* **string.parseInt() :** convertis une chaine de caractère nombre en type **Integer** 
  + **Exple :** ‘’12’’.**parseInt()**  -> **12**
* **string.parseFloat() :** convertis une chaine de caractère nombre en type **Float**
  + **Exple :** ‘’12.4’’.**parseFloat()** -> **12.4**
* **Number.toString() :** convertis un nombre en chaine de caractère.
* **Condition ternaires : [condition] ? [True] : [False] ;**

*let* x=3;

x<4 ? console.log('X inférieur à 4') : console.log('X supérieur à 4');

* **Les doubles pipe || permettent de créer une valeur par défaut :**

**Exple :**

*let* age= prompt("quel est votre age?")|| 12;

**-** Ici si l’utilisateur ne donne aucune valeur alors c’est **12**  vas être appelée

* **Exception :**

1. Les Tableaux

* Tableau simple :
  + **let nomTab = new Array(1, 2, 3, 4) ; ancienne façons**
  + **let nomTab = Array(1, 2, 3, 4) ;**
  + **let nomTab = [1, 2, 3, 4] ;**
* Tableau associatif :
  + **let tableauAssociatif = {**

**prenom: "Mark",**

**nom: "Zuckerberg",**

**position: "PDG de Facebook",**

**};**

* Traitement sur les tableaux :
  + monTableau.**push**("cinq"); **ajoute un élément à la fin du Tab.**
  + monTableau.**pop**() ; **supprime le dernier élément du Tab.**
  + monTableau.**unshift**("zéro") ; **ajoute un élément au début du Tab.**
  + monTableau.**shift**() ; **supprime le dernier élément du Tab.**
  + let indexUn = monTableau.**indexOf**("un"); **retourne l’index d’un élément du Tab.**
  + monTableau.join("\*"); **permet de concaténer les élément du Tab avec \***
  + **fonction "Splice"** permet de supprimer ou ajouter les élément dans un tableau a 1D ou 2D :

**monTableau.splice(index\_depart, nbre\_elem\_supprime, elem\_a\_ajouter);**

1. **Fonction fléchée**
2. **Closures**

**Les Closures** : permettent de sauvegarder les variables locales à l’intérieur des fonctions fléchées en comparaison des fonctions simple, qui a chaque appelle de la fonction la variable locale est de nouveau réinitialisée

1. **Objets**

* **Créer un Objet :**
* **Affection par décomposition :**
* **Methode sans décomposition :** ici on assigne chaque valeur de notre liste a des variables

* **Methode avec décomposition:** ici on assigne chaque valeur de notre liste dans une autre liste
* **4 type d’objets :**
* **L'objet Set**

Permet de stocker des **valeurs uniques de tout type**. Il s'agit d'une sécurité comparé à un simple tableau. Je vous recommande d'utiliser cet objet lorsque vous voulez stocker des informations sensibles qui ne doivent pas être dupliquées. L'exécution d'un objet Set est également plus rapide qu'un objet Array (donc qu'un tableau), même si sur de petits tableaux, cela ne se remarque pas.

* **L'objet Map :**

Permet de stocker **des paires clé-valeur** et de mémoriser leur ordre d'insertion.

* **L'objet WeakSet :**

Offre une flexibilité supplémentaire en comparaison de l'objet **Set**. Il n'accepte que **des objets**. **Une fois que l'objet stocké dans le WeakSet passe à null, il est automatiquement retiré.**

* **L'objet WeakMap :**

Offre une flexibilité supplémentaire en comparaison de l'objet **Map**. Il n'accepte que **des objets en clé**. On utilise souvent cet objet pour stocker des données. **Une fois que l'objet en clé n'existe plus (par exemple, si sa valeur passe à null), la valeur est automatiquement supprimée également.**

* **Passez un nombre infini d’argument : REST parameter**

Cela se fait avec **3 points (…)**

* **Spread Operator ou Operateur de décomposition**

L operateur Spread (**…**) prend un **tableau** et le décompose en **petit élément.**

* **Fusionnez plusieurs Tableau :**

* **Découper une chaine de caractère en plusieurs éléments**
* **Sélectionner une élément et stocker les autres dans une variables :**
* **Fusionner 2 Objets avec spread**

La fusion de 02 Objets à l’aide de spread prend en considération le dernier Objet dans le cas ou les objets ont **les même clés**, mais fais une **concaténation**  dans le cas contraire.

1. **BOM(Browser Objet Model)**

Il est constitué de :

* Objet **Window(**objet principal) : c’est l’objet global, il permet de :
  + Changer le titre dans l’onglet du navigateur(**window.title**)
  + Connaitre la taille du navigateur
  + Ouvrir une fenêtre(**window.open**)
  + Redimensionner une fenêtre(**window.resizeTo**)
  + Fermer une fenêtre(**window.close**) etc…
* Objet **Navigator** : permet de récupérer la position géographique de l’utilisateur. Entre autres il permet également de :
  + Récupérer la langue du navigateur(**navigator.language**)
  + Savoir si les cookies sont autorisés( **navigator.cookieEnabled**)
  + Connaitre le system d’exploitation(**navigator.platform**)
* Objet **History**: permet d’avoir une action sur l’historique des utilisateurs. On va pouvoir par exemple :
  + Pour aller à la page précédente(**history.back()**)
  + Pour aller à la page suivante(**history.forward())**
* Objet **Location :** permet de déterminer l’emplacement de l’utilisateur( le site sur lequel il se trouve, le dossier sur lequel il se trouve. Il permet **d’avoir une action sur URL sur lequel l’utilisateur se trouve actuellement.** Il est beaucoup utilisé lorsqu’ on veut  **faire des redirections.** On peut entre autres par exemple :
  + Recharger une page(**location.reload()**)
  + Faire une redirection(**location.href()**)
* Objet **Screen**: permet d’avoir des informations sur l’écran des utilisateurs. On peut avoir entre autres :
  + La largeur de l’écran(**screen.availWidth**)
  + La hauteur de l’écran(**screen.availHeight**
  + La résolution(**screen.pixel.Depth**)
* Objet **Document**: permet d’avoir des actions sur le html du site.

1. **DOM(Document Objet Model)**

* **Récupérer les éléments du DOM**

* **Modification d’un élément :**
* **Ajouter les éléments au DOM**
* **Supprimer l’élément du DOM**
* **Modifier le style d’un élément : .style**

1. **Les Évènements : clic, double-clic**

* **Les écouteurs : onclick(), onmouseover(), onmouseout() ...**

**NB : cette façon de faire est déconseillée**

* **Gestionnaire d’évènement : addEventListener()**
* **Supprimer un Évènement :** removeEventListener(élément, function)
* Gestion de la propagation des évènements

C’est le principe qui consiste a arrêté les évènements des **éléments filles** dans un **bloc parents**.

Pour résoudre cela il faut placer dans l’élément fille une fonction : **Event.stopPropagation()**

* **Planifier l’exécution d’un Script : setTimeout(), setInterval()**

**Exemple :** programme de décompte

* Exercice : Programme qui permet d’afficher et cacher
* **Annuler un évènement par défaut dans un formulaire :**

**preventDefault(),** permet d’annuler la fonction de base d’un élément html, par exemple celui du **formulaire**  avec sa methode **submit()** qui permet d’envoyer le formulaire.

formular.addEventListener("submit", (*e*) *=>* {

*e*.preventDefault();

});

1. **Programmation Oriente Objet**

* **Création d’un constructeur personnalise :**
* Partager les propriétés ou les méthodes entres objets : **prototype**
* Héritage :

* Classe
* Objet Date :

* Setter et Getter dans un objet Date
* Transformer une Date en format local
* Template String
* Utilisation des API
* Utilisation de **AJAX**  pour récupérer le prix du Bitcoins :
* **abort() :** permet de stopper toute activité et la connexion au serveur est interrompue et l’instance de l’objet XHR est remis a zéro. Ceci arrive lorsque la requête prend trop de temps.
* **La methode ‘’HEAD’’** permet de recevoir l’entête, cette technique est utilisée pour savoir l’existence du fichier sur un serveur.
* Il est possible de transmettre les paramètres par le biais de **‘’GET’’** sur URL dans la methode **open()**. Cela se fait à l’aide de : **?, &**

xhr.open('GET','http://mon\_site\_web.com/ajax.php**?**param1=valeur1**&**param2=valeur2');

* On devra tout avant tout **encoder**  les données( valeur1 et valeur2) des méthodes **‘’GET’’**  et **‘’POST’’** a l’aide de : **encodeURIComponent()**
* **getAllResponseHeaders() :**  permet de récupérer les valeurs de l’entête fourni avec fourni avec la réponse de la requête.
* **getResponseHeader(…) :**  permet de récupérer une seul valeur de l’entête et prend un paramètre par exemple : **‘’Content-type’’**
* Envois des données sans besoin de recharger la page :

methode **POST**

* Programmation Asynchrone et Promesse :
* **Asynchrone :**

C’est un programme qui exécute en même temps que les autres, n’attend pas la fin d’un autres programme pour s’exécuter.

De base JavaScript est une **programmation** **Synchrone** c est dire les programmes s’exécutent les un après les autres.

* **Promesse :**

Les promesse peuvent être : **En cours – honorer – rompue**

Une promesse est toujours **constante.**

* **Création d’une promesse :**
* Exemple : Ajout d’un script dans le head html :

* **Async et Await :**
* **Async** devant une fonction la force a retourné une promesse
* **Await** devant quelque chose, attente que la promesse soit

honorée.

* **Exercice de chargement de script ci-dessus, cette fois avec Async et Await et non avec .then(…)**
* **Récupérez les donnees avec fetch(url, Objet{…}) :**
* Avec **Async** et **Await**
* Avec **.then(…) .catch(…) :**

* **Envois des donnees avec fetch(…)**
* Avec **Async et Await :**

* Avec **.then(…) catch(…) :**
* **Les Axios : Récupérer les donnees**

C’est un bibliothèque qui permet de faire de l’Asynchrone facilement. Mais pour cela on doit d’abord ajouter son **cdn** dans le **head** du html.

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

* **Les Axios : Envois des donnees**
* **Les cookies**
* **Path**: Permet de préciser sur lequel répertoire (dossier) est disponible le cookie
* **domaine**: Permet de préciser sur quel nom de domaine est disponible le cookie
* **expires**: Permet de donner une date UTC jusqu'à laquelle le cookie est disponible, après cette date, il expirera
* **max-age**: Se substitue à expires, permet de donner une durée en millisecondes jusqu'à laquelle le cookie est disponible. Après cette date, il expirera
* **secure**: Permet de n'autoriser l'utilisation du cookie que sur les sites sécurisés (https)
* **L'erreur HttpOnly**

Pour créer un cookie, le serveur utilise une fonction qui possède, parfois, une option : httpOnly. Cette option interdit tout simplement la lecture / modification / suppression de ce cookie par JavaScript. Je vous rassure, si c'est vous qui programmez votre site, vous saurez à l'avance ce que vous faites. Mais il est bon de rappeler dans cette session que si vous mettez l'option httpOnly lorsque vous créez vos cookies via un autre langage de programmation (comme le PHP), votre cookie ne pourra pas être utilisé par JavaScript. Maintenant, vous le savez !

* **LocalStorage et SessionStorage**

**LocalStorage** = Stocké dans le navigateur même après qu'il soit fermé.

**SessionStorage** = Stocké dans le navigateur pendant la session

**Remarque :** Ces deux éléments ne sont pas envoyés au serveur et stocke des grands nbres de donnees.

Cependant Les **cookies** : Stocke des petits donnees qui vont être réutiliser par le navigateur. **Exple** : Identifiant user

On peut avoir les méthodes et les attributs suivants :

* **setItem**([clé], [valeur]) / **getItem**[clé] / **removeItem**[clé]
* **clear()** : supprimer tout
* **key**([index]) : obtenir la clef située à l'index donnée
* **length** : obtenir le nombre d'éléments stockés