LE TRAITEMENT DES DONNEES

Nous allons parler ici des données :

* Tableau
* Objets

Pour connaitre le type d’une variable, on utilise : **typeof**

**Console.log(typeof nom\_variable)**

* **Un tableau qui a des chaines de caractères comme éléments, on peut accéder à chaque lettre d’un mot du tableau (Une chaine de caractère est un tableau de caractère)**
* Un tableau en JS peut contenir plusieurs éléments de type différent comme :
* Des String
* Un number
* Un boolean
* Un autre tableau
* Un objet
* let array2=["Yaoundé",22,true,[2,3],{Nom: "AFANA JEOVANY"}];
* console.log(array2);//Cette ligne va afficher dans la console tous les éléments du tableau
* console.log(array2[4]["Nom"])//Cette ligne va afficher "AFANA JEOVANY"
* Pour afficher le nom ici, on fait : console.log(array2[4].nom)
* Explication :
* Lorsque nous faisons : array2[4], nous sommes déjà dans un objet et pour naviguer dans un objet, nous utilisons le «.»

1. LES OBJETS

Pour déclarer un objet en JS, on fat :

let objet={

    pseudo: "Denis",

    age: 33,

    technos: ["JavaScript","React","NodeJs"]

}

Explication : De la meme manière qu’un tableau peur contenir des objets, un objet peu également contenir des tableaux… C’est vice versa

* Pour ajouter un attribut à un objet, on fait :
* objet.adresse="Thies Lalane";

On met le nom de l’objet. le nom de l’attribut… Le programme va alors parcourir l’objet et s’il ne trouve pas un attribut qui porte déjà ce nom, il va juste l’ajouter à la suite des autres.

NB : Il n’est pas conseillé de créer un objet dans un objet alors si on a besoin d’avoir plusieurs objets, on peut ajouter ces objets dans un tableau. On aura alors un tableau d’objets

let data= [//On crée ici un tableau d’objet (On va en créer juste 3 ici)

{

    pseudo: "Samia",

    age: 33,

    technos: ["JavaScript","React","NodeJs"],

    admin:false,

},

{

    pseudo: "Denis",

    age: 33,

    technos: ["CSS","React","NodeJs"],

    admin:false,

},

{

    pseudo: "Nikolas",

    age: 33,

    technos: ["PHP","React","NodeJs"],

    admin:false,

}

]

1. LES STRUCTURES DE CONTROLE
2. Les structures ternaires (A chercher)
3. Les boucles for

* La boucle **for** avec **of**
* for(const user of data)
* {
* document.body.innerHTML+=`<li>${user.pseudo} </li>`;
* }

Lorsqu’on parcoure un tableau avec la boucle for et for, l’éléments(variable) qu’on précise (user ici) pour parcourir représente une case du tableau donc dans ce cas, il va à chaque fois représenter un objet… Donc grâce à lui on peut accéder à chaque objet du tableau et à ses attributs

**05/03/2024**

* De la même manière que nous pouvons savoir sue quel élément on n’a cliqué grâce au **e.target** appliqué au clic sur l’élément body, Nous pouvons aussi accéder à l’id de l’élément et agir en conséquence :
* document.body.addEventListener('click',(e)=>{
* console.log(e.target.id);
* switch(e.target.id)
* {
* Case :'javascript':
* document.body.style.background="red";break;
* Case : 'php':
* document.body.style.backgroundColor='blue';break;
* Case : 'python':
* document.body.style.backgroundColor='yellow';break;
* default: alert('Vous n\'avez sélectionné aucun élément.');
* }
* });

PROJET 1 : TEXT-ANIM (Dans un autre dossier)

1. **Les méthodes sur les strings**

* **Eval()** Permet de concaténer deux éléments de type différents : **eval(‘’1’’ + 2)** il va afficher **12**
* **Includes(chaine**) Permet de vérifier si notre chaine de caractère contient la chaine passée en paramètre à la méthode includes
* **parseInt()** Permet de transformer le type string en type **Int**.
* **isNan(variable)** Permet de vérifier si une variable n’est pas un nombre. La fonction renvoie **true** si la variable n’est pas de type number et **false** sinon.
* **Length** Permet de connaitre le nombre de caractères d’une chaîne
* **indexOf(caractère ou chaine\_de\_caractère)** Permet de trouver l’index d’un caractère ou d’une chaîne de caractère dans une chaîne de caractère… La fonction renvoie -1 si la chaine ou le caractère n’est pas trouvée dans la chaine… Sinon, elle renvoie l’index de l’élément cherché :
* let string='Je suis le plus beau';
* console.log(string.indexOf("suis"));//On va voir 3 s’afficher dans la console
* **Slice(nombre\_caractères\_a\_ignorer)** Permet d’ignorer un nombre précis de caractères qui sont au début d’une chaine, on peut ensuite afficher les caractères restants avec un console.log()

   console.log(string.slice(6));//On ignore les 6 premiers caractères de la chaîne lors de l’affichage dans la console

* **Split(caractère/Chaine\_a\_isoler)** Permet d’isoler un caractère ou une chaine de caractère d’une chaine… Il permet de séparer/ Décomposer une chaine… On peut l’utilise pour supprimer les espaces et compter le nombre de mots d’une chaine de caractères.
* console.log(string.split("i"));//Permet d'isoler la lettre i dans la chaîne string
* **toLowerCase()** Permet de transformer une chaine en minuscule
* **toUpperCase()** Permet de transformer une chaine en majuscule
* console.log(string.toUpperCase());//Affichhe la chaine en majuscule
* console.log(string.toLowerCase());//Affiche la chaine en minuscule
* **replace(element\_a\_remplacer,element\_qui\_va\_remplacer)** Permet de remplacer un caractère ou une chaine de caractère par une autre.
* console.log(string.replace("Je","Moi"));//Affiche string en remplaçant "Je" avec "Moi"

1. **Les méthodes sur les Numbers**

* **toFixed(nombre\_a\_virgule)** Permet de fixer le nombre de chiffre qu’on veut après la virgule.
* console.log(number.toFixed(3));//Affiche le nombre avec trois chiffres après la virgule
* **parseFloat()** Permet de transformer un nombre flottant qui est dans une chaine de caractère en float
* **Math.round(valeur) Permet** d’arrondir un nombre à la valeur qui lui est directement supérieure
* console.log(Math.round(5.8));//La fonction va arrondir le nombre à 6
* **Math.floor(valeur)** Permet d’arrondir un nombre à la valeur qui lui est directement inférieure (au plus bas)
* console.log (Math.floor(5.8));//La fonction va arrondir le nombre à 5
* **Math.ceil(valeur)** Permet d’arrondir au plus haut
* console.log(Math.ceil(5.1));//La fonction va arrondir le nombre à 6
* **Math.pow(valeur1,valeur2)** Permet d’afficher la puissance d’un nombre
* console.log(Math.pow(2,8));//Affiche 2 à la puissance 8
* console.log(2\*\*8);//Affiche 2 à la puissance 8
* **Math.sqrt(valeur)** Permet d’afficher la racine carrée d’un nombre

1. **Les Méthodes des Arrays (Tableaux)**

NB : Il est possible d’additionner deux tableaux en utilisant l’opérateur + mais dans ce cas, les éléments des deux tableaux seront juste côte à côte mais les tableaux ne seront pas fusionnés… Si on veut vraiment fusionner les deux tableaux, on utilise la fonction :

* **Concat(tableau)** Permet de concaténer deux tableaux
* let array3=["Javascript","Php","Python"];
* let array4=["Ruby","Solidity"];
* let newArray=array3.concat(array4);//On concatène les deux tableaux dans newArray
* console.log(newArray);
* On peut aussi utiliser le Spread Opérateur

let newArray= [...array3,...array4]; //On concatène les deux tableaux dans newArray

console.log(newArray);

* **Join(séparateur)** Permet de séparer les éléments d’un tableau lors de l’affichage, la fonction prend en paramètre le séparateur qu’on veut utiliser
* console.log(array3.join(' ')) ; //Le séparateur ici c'est l'espace
* **slice(nombre\_element\_a\_ignorer)** Permet de couper(ignorer) un ou plusieurs éléments du tableau lors de l’affichage

console.log(newArray.slice(3));//On ignore les 3 premiers éléments du tableau

* **indexOf(valeur)** Permet de rechercher l’index d’un élément dans un tableau
* **forEach()** Permet d’accéder à chaque élément d’un tableau/objet de manière individuelle
* newArray.forEach((languages)=>{
* console.log(languages);
* })
* **Every()** Permet de vérifier si tous les éléments du tableau contiennent une certaine valeur ou donnée
* **Some()** Permet de vérifier si une valeur ou une donnée existe au moins une fois dans une case du tableau

console.log(newArray.every((langage)=> langage == "PHP")) ;//Renvoie false car toutes les cases ne sont pas égales à PHP

console.log(newArray.some((langage)=> langage == "PHP"));//Renvoie true car on a au moins une case dans le tableau qui est égale à Php

* **Shift()** Permet d’enlever et retourner le premier élément du tableau… On peut affecter cet élément à une variable par exemple ou alors faire tout ce qu’on veut avec…
* let shift=newArray.shift(); //On enlève le premier élément du tableau et on  l'affecte à la variable shift

console.log(newArray); //On constate ici que le premier élément du tableau qui était là avant n'est plus et qu'il a été remplacé par l'élément qui le suivait directement

* **Pop()** Permet de supprimer le dernier élément du tableau
* **Splice()** permet de remplacer un ou plusieurs éléments d’un tableau par un autre et la fonction retourne le/les éléments qui ont été remplacés

const restArray=newArray.splice(1,2,"C++");//On remplace Les 2 éléments à partir de celui qui est à l’index 2 par `C++`

console.log(newArray);

1. **IMPORTANT**

* **Reduce()** Permet d’additionner, multiplier, soustraire tous les éléments d’un tableau

console.log (arrayNumber.reduce((x, y) =>x + y)) ; La fonction prend deux paramètres x et y et elle va additionner tous les éléments du tableau et renvoyer la valeur

* **filter()** Permet de filtrer les éléments d’un tableau par rapport à une condition donnée

console.log (array3.filter((number)=>number>10 && number <20));//La fonction prend un paramètre (number ici) qui va représenter les valeurs du tableau à chaque tour de boucle… On peut ensuite appliquer une condition (ici la console va afficher les cases du tableau qui ont une valeur comprise entre 10 et 20

* **sort()** Permet de trier les éléments d’un tableau (Dans l’ordre croissant ou décroissant)

console.log (array3.sort((a,b)=>b-a));//Comme on n’a mis ‘’b-a’’ les éléments seront triés dans l’ordre décroissant… Si on veut les trier dans l’ordre croissant on va utiliser ‘’a-b’’

NB : On peut également utiliser ces méthodes ensembles… Par exemple :

console.log (array3.filter((number)=>number >10).sort((a,b)=>b-a));// Ce bout de code va permettre d’afficher les éléments qui sont supérieurs à 10 et rangés dans l’ordre croissant

* **map()** fonctionne un peu comme forEach et permet de sélectionner chaque élément d’un tableau ou d’un objet

document.body.innerHTML=array3.map((number)=>`<li> ${number}</li>`).join(" ")

1. **METHODES POUR LES OBJETS**

Note\_\_\_ La puissance de la méthode **map()** se fait le mieux voir lorsque nous l’appliquons à sur des objets… Par exemple si nous avons un tableau d’objet que nous voulons afficher sur notre page ou alors afficher un attribut spécifique des objets (Comme le nom par exemple), voici comment nous pouvons utiliser la méthode **map()** :

document.body.innerHTML = data.map((*user*) => `<div class=’’user-card’’><h2>${*user*.pseudo}</h2></div>`).join("")

NB : Nous avons profiter pour ajouter une classe à nos div

document.body.innerHTML = data.map((*user*) =>

  `<div class=’’user-card’’>

      <h2>

        ${*user*.pseudo}

      </h2>

      <p>Age: ${*user*.age}</p>

      <p>Status: ${*user*.admin ? "Modérateur" : "Membre"}</p>

  </div>`).

  join("")

Note… On peut aussi trier nos éléments dans l’ordre croissant (Selon l’âge), et pour ça on va ajouter la méthode **sort()** à notre code

document.body.innerHTML = data

.sort((*a*, *b*) => *b*.age - *a*.age)

.map((*user*) =>

  `<div>

      <h2>

        ${*user*.pseudo}

      </h2>

      <p>Age: ${*user*.age}</p>

      <p>Status: ${*user*.admin ? "Modérateur" : "Membre"}</p>

  </div>`).

  join("")

Note\_\_\_ On peut aussi filtrer nos éléments lors de l’affichage (Ajouter une condition), et pour ça on va ajouter la méthode **filter()** à notre code :

document.body.innerHTML = data

.filter((*user*) => *user*.admin === false)

.sort((*a*, *b*) => *b*.age - *a*.age)

.map((*user*) =>

  `<div>

      <h2>

        ${*user*.pseudo}

      </h2>

      <p>Age: ${*user*.age}</p>

      <p>Status: ${*user*.admin ? "Modérateur" : "Membre"}</p>

  </div>`).

  join("")

1. LES DATES
2. Date classique

*//Date classique*

let date = new Date();

console.log(date);

1. Timestamp

let timestamp = Date.parse(date);

console.log(timestamp);

Ca nous donne le nombre de millisecondes qui se sont écoulé depuis 1970

1. IsoString
2. *//IsoString*
3. console.log(date.toISOString())
4. MANIPULATION DES DATES

* **toLocaleDateString()** est une méthode très puissante en JS qui permet de manipuler les dates avec plus de précision

function dateParser (*chaine*){

  let newDate = new Date(*chaine*).toLocaleDateString("fr-FR", {

    year: "numeric",

    month: "long",

    day: "numeric",

    hour: "numeric",

    minute: "numeric",

  });

*return* newDate;

}

console.log(dateParser(date));

**Ce code va afficher la date actuelle en France**

**NB : Ca fonctionne aussi avec les formats timestamp et IsoString**

function dateParser (*chaine*){

  let newDate = new Date(*chaine*).toLocaleDateString("fr-FR", {

    year: "numeric",

    month: "long",

    day: "numeric",

    hour: "numeric",

    minute: "numeric",

  });

*return* newDate;

}

console.log(dateParser(timestamp));

1. **DESTRUCTURING**

**C’est une méthode utilisée pour accéder aux éléments d’un objet sans avoir besoin de mettre à chaque fois le nom de l’objet…**

let moreData = {

  destVar: ["Element 1", "Element 2"],

};

const { destVar } = moreData;

console.log(moreData.destVar);

console.log(destVar);

**Ca s’applique aussi sur les tableaux, si par exemple nous avons un tableau qui contient des éléments et que nous voulons affecter (Isoler) chaque élément du tableau à une variable distincte (Déstructurer)**

let array5 = [70, 49, 10];

let [x, y, z] = array5

console.log(x)

console.log(y)

console.log(z)

* **Utilisation du destructuring avec les dates**
* const dateDestructuring = (*chaine*) =>{
* let newDate = *chaine*.split('T')[0]; *//On sépare notre date en deux parties (On obtient ici un tableau de deux éléments)*
* [y, m, d] = newDate.split("-");*//On sépare new Date en utilisant '-' comme séparateur*
* *return* [d, m, y].join("/"); *//On utilise la méthode Join() pour regrouper les éléments en utilisant '/' comme séparateur*
* }

1. **LES REGEX**

*//----------------Les Regex ----------------*

let mail = 'wilfriedd2ylan@gmail.com';

console.log(mail.replace(/wil/, "joe"));

console.log(mail.match(/[gsf]/));

*//Matcher tous les chiffres*

console.log(mail.match(/\d/));

*//Matcher toutes les lettres*

console.log(mail.match(/[A-Z]/));

*//-----Controler un mail-----*

*/\*\**

*\* w: Tous les caractères chiffres ou lettre*

*\*/*

console.log(mail.match(/*^*[\w\_-]+@[\w-]+\.[a-z]{2,4}*$*/i));

*//Afficher les nombres recus en les séparant par millièmes*

let separateur = 13424345436

console.log(separateur.toString().replace(/*\B*(?=(\d{3})+(?!\d))/g, "."));

*/\*\**

*\* Le résultat sera: 13.424.345.436*

*\*/*