

Cahier d'exercices JavaScript
PSB
Année 2020
V1.0

A/ Exercice de manipulation de base avec le langage

1/

```
test='valeur'
test+23
a='23'
b=16
a-b
a+b

console.log(1 + "2" + "2");
console.log(1 + "2" + "2");
console.log(1 + "-1" + "2");
console.log(+ "1" + "1" + "2");
console.log( "A" - "B" + "2");
console.log( "A" - "B" + 2);

012+0123
95+025
```

```
Chaîne de caractères :
var arr1="john".split("");
var arr2=arr1.reverse();
var arr3="jones3".split("");

arr2.push(arr3);

arr1 ? arr2 ? pourquoi ?
```

Attention parce que l'on a une référence sur le même objet. ce n'est pas une copie .. utilisé slice()

Conversions :

```
parseInt('12')
parseInt('12',8)
parseInt('12',16)
parseInt('111',2)
parseInt('blabla')

parseFloat('12,32')
Affichage 1 chiffre derrière la virgule
```

```
12
10
18
7
undefined

12.32
parseFloat('12.32').toFixed(1)
```

2/ Tableaux :

Manipulation au niveau des tableaux

```
var monTableau=[12,51,10,9,8,7,6] ;
```

Veillez modifier la position 2 à 5 avec la valeur 99 ?

```
monTableau.fill(99,2,5)  (7) [6, 7, 99, 99, 99, 12, 51]
```

Veillez trouver l'instruction pour trier dans l'ordre croissant et décroissant ce tableau.

```
si on exécute directement la fonction sort sur le tableau on obtient  
(7) [12, 51, 6, 7, 99, 99, 99]
```

```
monTableau.sort((a,b)=>{return Number(a)>Number(b)?-1:Number(a)<Number(b)?1:0})
```

```
monTableau.sort((a,b)=>{return Number(a)>Number(b)?1:Number(a)<Number(b)?-1:0})
```

On souhaite le copier dans une autre variable monTableau2 ?

```
var nouvelle_var=monTableau.slice() ;
```

Récupérez une valeur à un indice dans la tableau et au delà ... que renvoie la fonction ?

```
retourne undefined
```

Veillez ensuite modifier tous les éléments du tableau est les mettre à -1 ?

```
monTableau.fill(-1) ;
```

On souhaite rajouter une entrée test='PSB'

Veillez modifier le contenu de ce tableau

Quelle est la longueur de ce tableau ? expliquez pourquoi ?

```
monTableau["test"]="voiture";
```

La longueur n'a pas bougé, car length ne compte pas les associations

3/ Scope des variables :

```
var a=10  
console.log(a) ;  
function maFonction() {  
    a+=10 ;  
    console.log(a) ;  
}  
  
maFonction() ;  
console.log(a) ;
```

```
var a=10
console.log(a);
function maFonction() {
    var a;
    a=20;
    console.log(a);
}

maFonction();
console.log(a);
```

On passe dans le tableau ce que l'on appelle une fonction fléchée qui ne peut pas avoir de paramètre

Autres exemples de <https://blog.pragmatists.com/let-you-function-fléché-qui-peut-constant-1633e56a948d>

Soit le code suivant :

```
var callbacks = [];
for (var i = 0; i < 4; i++) {
    callbacks.push(() => console.log(i));
}
callbacks[0]();
callbacks[1]();
callbacks[2]();
callbacks[3]();
```

Solution ==>

```
var callbacks;
//var i;
callbacks = [];
for (let i = 0; i < 4; i++) {
    //let j=i; // on peut garder la var i mais on
    définit une nouvelle variable j qui est en let
    callbacks.push(() => console.log(i));
}
callbacks[0]();
callbacks[1]();
callbacks[2]();
callbacks[3]();
```

Donnez le résultat de l'exécution de ce code ... expliquez ce qui se passe et testez la solution

Explication : On inscrit dans chaque fonction l'instruction console.log(i) qui est évaluée lors de l'appel de la méthode. La valeur de i à la sortie de la boucle est ==4 donc les appels des fonctions afficheront dans la console cette valeur.

Maintenant si on définit une variable j en mode let avec le scope de la boucle elle même, l'évaluation de la valeur se fait au moment du chargement de la fonction anonyme. Donc l'évaluation par la suite se réalise comme il le faut.

4/ Fonctions avancées :

Passage de paramètres/arguments

Soit le programme suivant veuillez indiquer dans quel mode est transmis la variable car à la fonction et la variable poids

```
var Poids=1.5 ;
var car={"name" : "voiture1", "nbplaces":5,"Couleur":"rouge"};
console.log(car) ;
// objectif compléter la variable V en ajoutant le poids du véhicule
function adapte_voiture(V, poids) {
    poids*=1.3 ;
    V["poids"]=poids ;
}
adapte_voiture(car, Poids) ;
console.log(car, Poids) ;
```

Comment est passé le paramètre Poids et l'objet car ?

Si on ne souhaite pas qu'une fonction puisse modifier un objet passé en paramètre comment peut-on faire et que se passe-t'il lorsqu'on essaye de le modifier ?

Dans cette étape on est obligé d'utiliser la syntaxe suivante :

```
var Poids=1.5 ;
var car={"name" : "voiture1", "nbplaces":5,"Couleur":"rouge"};
console.log(car) ;
// objectif compléter la variable V en ajoutant le poids du véhicule
function adapte_voiture() {
    const [V, poids]=arguments ;
    poids*=1.3 ;
    V["poids2"]=poids ;
}
adapte_voiture(car, Poids) ;
console.log(car, Poids) ;
```

Arguments représente une liste qui porte les paramètres passés à l'appel de la fonction. le fait de le forcer à const bloque toute possibilité de modification si besoin.

4-2/ fonction initiale()

Veuillez définir une fonction qui reçoit un string et renvoie le même string avec une majuscule pour la première lettre et le reste en minuscules.

On peut commencer dans la console avec les manipulations de chaînes et des méthodes substr()

Ce qui nous donne "voiture".substr(0,1).toUpperCase()

Si on traduit ça dans une fonction

```
function initiale(txt) {
    if (txt==undefined || txt=="") {
        return " " ;
    }
    return txt.substr(0,1).toUpperCase() + txt.substr(1).toLowerCase() ;
}
```

4-3 / Cette même fonction on souhaite la définir directement dans le prototype de String. veuillez faire l'aménagement.

Pour l'association au prototype String ça ressemble beaucoup à la méthode ci dessous mais la variable sur laquelle on travaille sera exploitée par le mot clé this ... ce qui nous donne

```
String.prototype.initiale=function () {  
  if (this==undefined || this=="") {  
    return " ";  
  }  
  return this.substr(0,1).toUpperCase() + this.substr(1).toLowerCase();  
}
```

4-4/ On souhaite sur une chaîne de caractères modifier les numériques et les mettre entre parenthèses
Veuillez définir la fonction pour le faire ?

Exemple : demain il fait 32 à l'ombre

demain il fait (32) à l'ombre

```
function numentreparentheses(texte){  
  function encadrer(hit,numgroupe,str) {  
    return (numgroupe>0?(':'+hit+(numgroupe>0?')':''));  
  }  
  return texte.replace(/[0-9]*/g,encadrer);  
}
```

4-5/ Veuillez implémenter les fonctions suivantes

Ecriture d'une fonction de sum avec arguments

Ecrire une fonction qui doit fonctionner dans les cas suivants :

console.log(sum(2,3)); // Outputs 5

console.log(sum(2)(3)); // Outputs 5

```
function sum(x) {  
  if (arguments.length == 2) {  
    return arguments[0] + arguments[1];  
  } else {  
    return function(y) { return x + y; };  
  }  
}
```

Ecriture d'une fonction qui détecte si la chaîne est un palindrome et renvoie true

```
console.log(isPalindrome("level")); // true  
console.log(isPalindrome("levels")); // false  
console.log(isPalindrome("A car, a man, a maraca")); //true
```

```
function isPalindrome(str) {  
    str = str.replace(/\W/g, "").toLowerCase();  
    return (str == str.split("").reverse().join(""));  
}
```

Exemple de fonctions fléchées

Quelques exemples de manipulation de ce type de fonction :

Soit la fonction **var fonction1=x => x*x ;**
Utilisation de la fonction : fonction1(21)

Soit la fonction **var fonction2=(x,y) =>{return x+y;}**
Si on exécute fonction2(4,5) ==> 9
Si on exécute fonction2(4) ==> NaN

On peut définir la fonction avec des éléments par défauts
Soit la fonction **var fonction3=(x,y=0) =>{return x+y;}**
Si on exécute fonction2(4,5) ==> 9
Si on exécute fonction2(4) ==> 4

Blocs d'instructions :

try/catch/finally

```
const number = 42;  
try {  
    number = 99;  
} catch(err) {  
    console.log(err);  
}  
console.log(number);
```

Indiquez pourquoi il y a un problème ?

5/ Prototype et manipulation des classes

Soit l'exercice suivant à réaliser directement sous la console de votre navigateur

```
function Vehicule() {  
    this.type='non renseigné' ;  
    this.marque='non renseigné' ;  
}
```

```
function Voiture() {  
    this.type='voiture' ;  
    this.modele='non renseigné' ;  
    this.moteur='non renseigné' ;  
    this.puissance='non renseigné' ;  
}
```

```
function Renault() {  
    this.marque='Renault' ;  
}
```

```
Voiture.prototype=new Vehicule ;  
Renault.prototype=new Voiture ;
```

Veuillez saisir les différents éléments dans votre environnement pour constater le comportement qu'offre Javascript en terme d'héritage :

on utilise nos définitions :

```
var maVoiture=new Renault();  
var maVoiture2=new Renault();  
maVoiture2.kilometrage='124521' ;
```

Quels sont les attributs de maVoiture2 ?

Veuillez ajouter un nouvel attribut à la classe Voiture par exemple nombre de place .

Que deviennent les attributs de maVoiture2 ? après ce changement ?

Développement d'une méthode de tri d'objets.

Imaginons que l'on stocke une série d'objets de type quelconque de type personne dans un tableau
Veuillez trouver un moyen de classer le tableau dans l'ordre des noms des personnes.

```
var liste_personnes=[] ;  
for (var i=0;i<10;i++){  
    var enfant={"nom":"valeur"+i,"prenom" : "prenom"} ;  
    liste_personnes.push(enfant) ;  
}
```

Exploration d'une DOM

Visite d'une page Web

Créer une fonction qui reçoit la DOM de la page chargée et qui visitera la page elle-même et tous les descendants et pas seulement les descendants immédiat. Pour chaque élément visité, la fonction devra fournir une fonction callback pour afficher le contenu.

```
function Traverse(p_element,p_callback) {  
    p_callback(p_element);  
    var list = p_element.children;  
    for (var i = 0; i < list.length; i++) {  
        Traverse(list[i],p_callback); // recursive call  
    }  
}
```

Essayez d'utiliser cette fonction sur une page simple avec en paramètre

```
var affiche=function(e) {console.log(e);}
```

```
Traverse(document, affiche) ;
```

Manipulation de la DOM d'une page

Veuillez charger la page de l'exercice PSB/Chapitre4 sur votre ordinateur et charger la page html dans votre navigateur

Utilisez la fonction traverse pour analyser le contenu de cette page.

Que contiennent les variables

window

document

On souhaite modifier l'affichage de la page chargée :

Veuillez mettre au point ce traitement

Voir cours Page 42

Voir cours page 43

Voir cours Page 44

Gestion des événements : Page 51

Lorsqu'on passe avec la souris sur les articles , on souhaite un changement de style et de taille des Articles.

Functional javascript

```
var colors = ['blue', 'black', 'red'];  
var cars = ...;
```

A partir de ce code, construire un véhicule par couleur

```
var colors = ['blue', 'black', 'red'];  
var cars = colors.map(buildCar);  
  
function buildCar (color) {  
    return new Car(color);  
}
```

Objets :

Utilisation d'une fonction privée attention chaque objet créé aura sa propre fonction privée ...

```
var Employee = function (name, company, salary) {
    this.name = name || "";
    //Public attribute default value is null
    this.company = company || ""; //Public attribute default value is null
    this.salary = salary || 5000; //Public attribute default value is null
    // Private method
    var increaseSalary = function () {
        this.salary = this.salary + 1000;
    };
    // Public method
    this.displayIncreasedSalary = function() {
        increaseSalary();
        console.log(this.salary);
    };
};

// Create Employee class object
var emp1 = new Employee("John","Pluto",3000);
// Create Employee class object
var emp2 = new Employee("Merry","Pluto",2000);
// Create Employee class object
var emp3 = new Employee("Ren","Pluto",2500);
```

Les fermetures

Exercice 1/ avec changement de couleur toutes les secondes par article

Charger le fichier exercice_1-4.html et veuillez modifier via la navigateur les paramètres de telle sorte que lors du click sur le bouton un les paragraphes les uns après les autres changent de couleur.

Voilà la première tentative

```
function parseelementengris(e) {  
    e.style.color='gainsboro';  
}  
function passegris() {  
    var lesarticles=document.querySelectorAll("article > p");  
    var cpt=0;  
    for (var article of lesarticles) {  
        setTimeout(function() {parseelementengris(article);},++cpt*1000);  
    }  
}  
document.querySelector('#pageUne').addEventListener('click',passegris);
```

Implémenter cette première méthode et corriger ce qui ne va pas

Exercice 2/

Elaboration d'un compteur

Veillez élaborer une fonction qui permet d'offrir la fonctionnalité compteur. A chaque appel de cette fonction on incrémente à 1. Contrainte on ne veut pas de variable globale qui soit incrémentée directement.

Des tentatives ont été développées :

```
var counter=0 ;  
function add() {  
    counter+=1 ;  
}  
  
add() ;  
add() ;  
add() ;
```

Résultat et commentaire :

```
var counter=0 ;  
function add() {  
    var counter=0 ;  
    counter+=1 ;  
}  
  
add() ;  
add() ;  
add() ;
```

Résultat et commentaire :

```
function add() {  
    var counter=0 ;  
    counter+=1 ;  
    return counter ;  
}  
  
add() ;  
add() ;  
add() ;
```

Résultat et commentaire :

Une solution avec la notion de closure (fermeture) est envisagée ce qui nous donne

```
var add=(  
    function() {  
        var counter=0 ;  
        return function() {  
            counter++ ;  
            return counter;  
        }  
    })();  
  
add() ;  
add() ;  
add() ;
```

Résultat et commentaire :

Exercice 3

On souhaite créer un formulaire avec trois champs à saisir et une aide adaptée à ces champs
Champs

Aide :

E-Mail :

Nom :

Age :

Veillez définir le programme qui permet d'effectuer cette saisir

```
<html>
  <head>
  </head>
  <body>
    <p id="aide">Des aides seront affichées ici</p>
    <p>E-mail : <input type="text" id="email" name="email"></p>
    <p>Nom : <input type="text" id="nom" name="nom"></p>
    <p>Âge : <input type="text" id="âge" name="âge"></p>
    <script>
      function afficherAide(aide) {
        document.getElementById('aide').innerHTML = aide;
      }
      function preparerAide() {
        var texteAide = [
          {'id': 'email', 'aide': 'Votre adresse e-mail'},
          {'id': 'nom', 'aide': 'Vos prénom et nom'},
          {'id': 'âge', 'aide': 'Votre âge (plus de 16 ans requis)'}
        ];
        for (var i = 0; i < texteAide.length; i++) {
          var item = texteAide[i];
          document.getElementById(item.id).onfocus = function() {
            afficherAide(item.aide);
          }
        }
      }
      preparerAide();
    </script>
  </body>
</html>
```

Testez ce programme et veuillez le corriger pour qu'il fonctionne ... affiche l'aide en fonction du focus sur le champ correspondant :

Corrections :

Tableaux :

```
monTableau.sort((a,b)=>{return Number(a)>Number(b)?-1:Number(a)<Number(b) ?1:0 });
```

Tableau de liste d'objet que l'on souhaite classer

Fonction pour comparer les différents éléments entre eux selon la propriété ou les propriétés désirées.

```
function compare( a, b ) {  
  if ( a.nom < b.nom ){  
    return 1;  
  }  
  if ( a.nom > b.nom ){  
    return -1;  
  }  
  return 0;  
}
```

Fonction pour les parenthèses :

```
function numentreparentheses(texte) {  
  function encadrer(hit,numgroupe,str){  
    return (numgroupe>0 ?'(' :')+hit+(numgroupe>0 ?')' :')';  
  }  
  return texte.replace(/[0-9]*/g,encadrer);  
}
```

Réf exercice : <https://developer.mozilla.org/fr/docs/Web/JavaScript/Closures>

Exercice sur les fermetures :

```
<html>
  <head>
  </head>
  <body>
    <p id="aide">Des aides seront affichées ici</p>
    <p>E-mail : <input type="text" id="email" name="email"></p>
    <p>Nom : <input type="text" id="nom" name="nom"></p>
    <p>Âge : <input type="text" id="âge" name="âge"></p>
    <script>
      function afficheAide(aide) {
        document.getElementById('aide').innerHTML = aide;
      }
      function creerCallbackAide(aide) {
        return function() {
          afficheAide(aide);
        };
      }
      function prepareAide() {
        var texteAide = [
          {id: 'email', 'aide': 'Votre adresse e-mail'},
          {id: 'nom', 'aide': 'Votre prénom et nom'},
          {id: 'âge', 'aide': 'Your age (you must be over 16)'}
        ];
        for (var i = 0; i < texteAide.length; i++) {
          var item = texteAide[i];
          document.getElementById(item.id).onfocus = creerCallbackAide(item.aide);
        }
      }
      prepareAide();
    </script>
  </body>
</html>
```

Manipulation DOM

Exercice de manipulation et de création d'objets dans la page actuelle

On souhaite ajouter 5 boutons dans la page avec un comportement donné (affichage dans la console)

Voici la première version du code

```
for (var i = 0; i < 5; i++) {  
    var btn = document.createElement('button');  
    btn.appendChild(document.createTextNode('Button ' + i));  
    btn.addEventListener('click', function(){ console.log(i); });  
    document.body.appendChild(btn);  
}
```

Solution proposée ;

```
for (var i = 0; i < 5; i++) {  
    var btn = document.createElement('button');  
    btn.appendChild(document.createTextNode('Button ' + i));  
    btn.addEventListener('click', (function(i) {  
        return function() { console.log(i); };  
    })(i));  
    document.body.appendChild(btn);  
}
```

Expliquez la solution

Explications :

Exercice 1 : Web Socket

Élaboration d'un chat en ligne avec Socket.io et Express

Création des répertoires et préparation de l'environnement.

```
$ mkdir socket-app  
$ cd socket-app  
$ npm init
```

```
$ npm install socket.io express --save  
$ npm install jquery --save
```

Remplir les différents éléments :
nom de l'auteur
fichier app.js
lancement npm start

Regardez le contenu du fichier packages.json qui définit les dépendances de l'application.

Création du fichier de démarrage de l'application nodejs

```
// app.js  
var express = require('express');  
var app = express();  
var server = require('http').createServer(app);  
var io = require('socket.io')(server);  
  
app.use(express.static(__dirname + '/node_modules'));  
app.get('/', function(req, res, next) {  
  res.sendFile(__dirname + '/index.html');  
});  
  
server.listen(4200);
```

```
<!doctype html>  
<html lang="en">  
  <head>  
  
  </head>  
  <body>  
    <h1>Hello World!</h1>  
    <div id="future"></div>  
    <form id="form" id="chat_form">  
      <input id="chat_input" type="text">  
      <input type="submit" value="Send">  
    </form>  
    <script src="/jquery/dist/jquery.js"></script>  
    <script src="/socket.io/socket.io.js"></script>  
  </body>  
</html>
```

Pour le moment rien de fantastique

Mise en place de l'échange via la web socket

Ajout dans le fichier html

```
io.on('connection', function(client) {  
  console.log('Client connected...');  
  
  client.on('join', function(data) {  
    console.log(data);  
  });  
});
```

```
<script src="/jquery/dist/jquery.js"></script>  
<script src="/socket.io/socket.io.js"></script>  
<script>  
  var socket = io.connect('http://192.168.1.26:4200');  
  socket.on('connect', function(data) {  
    socket.emit('join', 'Hello World from client');  
  });  
</script>
```

Les deux parties se connectent via la web socket

Le serveur renvoie un message sur le client

```
io.on('connection', function(client) {  
  console.log('Client connected...');  
  
  client.on('join', function(data) {  
    console.log(data);  
    client.emit('messages', 'Hello from server');  
  });  
});
```

```
// index.html  
...  
    socket.on('messages', function(data) {  
      alert(data);  
    });
```

Pour finir notre application chat :

```
// app.js  
...  
io.on('connection', function(client) {  
  console.log('Client connected...');  
  
  client.on('join', function(data) {  
    console.log(data);  
  });  
  
  client.on('messages', function(data) {  
    client.emit('broad', data);  
    client.broadcast.emit('broad', data);  
  });  
});
```

```
<!--index.html-->  
...  
<script>  
var socket = io.connect();  
socket.on('connect', function(data) {  
  socket.emit('join', 'Hello World from client');  
});  
socket.on('broad', function(data) {  
  $('#future').append(data+ "<br/>");  
});  
  
$('#form').submit(function(e){  
  e.preventDefault();  
  var message = $('#chat_input').val();  
  socket.emit('messages', message);  
});  
</script>
```

Vous pouvez tester votre chatSimple

Pour un fonctionnement plus évolué voir les exemples de socket.io

Contenu du fichier packages.json

```
{  
  "name": "socket-app",  
  "version": "1.0.0",  
  "description": "Application de demo de chat",  
  "main": "app.js",  
  "scripts": {  
    "start": "node ./app.js"  
  },  
  "author": "Y.Stroppa",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.17.1",  
    "jquery": "^3.4.1",  
    "socket.io": "^2.3.0"  
  }  
}
```

Pour exécuter : npm start

Autres exercices

```
soit la fonction suivante :  
function readHugeList(l) {  
    var liste=[];  
    for (var i =0 ; i<l;i++) {  
        liste.push("valeur"+ i) ;  
    }  
    return liste ;  
}
```

```
soit la fonction suivante :  
var list=readHugeList() ;  
var nextListItem=function() {  
    var item=list.pop() ;  
    if (item) {  
        // process pour traiter les elements item  
        console.log(item) ;  
        nextListItem() ;  
    }  
} ;
```

```
soit la fonction suivante :  
var list=readHugeList() ;  
var nextListItem=function() {  
    var item=list.pop() ;  
    if (item) {  
        // process pour traiter les elements item  
        setTimeout(nextListItem(),0) ;  
    }  
} ;
```