

JAVASCRIPT

1 - Bases du langage

SOMMAIRE

Sommaire.....	1
JavaScript – introduction, tour complet et bases du code	3
Installation des fichiers de tests	3
JavaScript – Introduction	3
Intro	3
Environnement de travail	5
Balise <script>	6
Exemples	7
Quelques références	9
JavaScript – Premiers usages	10
Console Javascript	10
04 - variable, addition, concaténation, prompt() – A tester	12
05 – premières fonctions – A tester	13
06 - document.write() et test – A tester	14
07 – document.getElementById() et boucle for – A tester	14
08 – Button : programmation événementielle – A tester	15
09 – onclick, onmouseover, on mouseout – A tester	16
10 – console – A tester	18
11 – Paramètres en sortie : fonction d’inversion – A tester	18
Organisation du cours	21
Bases du code - 1	22
Les 3 + 1 types	22
Opérations de base	23
Affichage : console.log	24
Commentaires	24
Variable	25
Conversions de types	25
Expression et évaluation d’une expression	26
Tests	27
Boucle	28
Fonction	29
Paramètres en sortie des fonctions : exemple 11	30
Fonctions prédéfinies	31
Exercices – Série 1	33
1 – Calculs sur des figures	33
2 – Jour de la semaine	33
3 – Table de multiplication	34
4 – Jeu des allumettes (jeu de Marienbad)	35
JavaScript – tableaux et structures	39
Installation des fichiers de tests	39
Bases du code – 2 – Tableaux et Structures	39
Tableaux : exemple 1	39
Structure ou Objet en JS : exemple 2	41

Boucles spéciales : exemples 1 et 2	42
Tableau de structures : exemple 3	43
Structure avec fonctions : exemple 4	44
Exercices – Série 2	45
1 – Tableau de notes	45
2 - Tableau d'élèves avec des notes – Tri d'une structure.....	46
3 - Tableau d'élèves avec des notes – creerEleve.....	47
4 – Jeu de grammaire	47
5 – Pipotron , Poétron	48
JavaScript – POO	49
Installation des fichiers de tests	49
Programmation objet	49
Principes	49
Structure avec méthode – exemple 1	49
Classe : notion de prototype – exemple 2	50
Héritage - exemple 2	52
Exercice 1 : une IHM pour l'exemple 2	53
Tableau d'objets.....	54
Exemple 3	54
Exercice 2 : une IHM pour l'exemple 3	54
Exercice 3 : tableau d'élèves avec notes et photos en POO	54

Edition : mars 2019

JAVASCRIPT – INTRODUCTION, TOUR COMPLET ET BASES DU CODE

Installation des fichiers de tests

Dans le cours :

Les exemples sont présentés dans un chapitre en vert.

Les exercices à faire sont présentés dans un chapitre en jaune.

JavaScript – Introduction

Intro

Qu'est-ce que c'est ?

Le JavaScript (JS) est un langage de script (comme le HTML ou le PHP).

Le code JS qui s'ajoute à la page HTML dans une balise <script>.

Il est interprété par le navigateur. Il permet **de rendre plus dynamique et interactive la page HTML**, sans passer par le serveur.

Il peut aussi communiquer avec le serveur en utilisant les technologies AJAX.

C'est un langage objet, non orthodoxe !

Rien à voir avec Java !

Historique

Wiki : <https://fr.wikipedia.org/wiki/JavaScript>

Inventé en **1995** par Brendan Eich pour **Netscape** premier navigateur Web populaire (l'ancêtre de Firefox).

Aujourd'hui, tous les navigateurs comprennent le JavaScript aujourd'hui.

Standardisé par l'ECMA International sous le nom d'ECMAScript. Dernière version standardisée : ECMAScript 5, sorti en 2009. ECMAScript 6 et 7 qui ne sont pas encore standardisées.

Côté serveur

Apparition en 2009 de la plate-forme [Node.js](https://nodejs.org/), qui permet d'écrire en JavaScript des applications Web très rapides : l'environnement Node.js exécute du JavaScript côté serveur pour générer du HTML dans lequel du JavaScript pourra toujours être exécuté côté client.

<https://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js/node-js-mais-a-quoi-ca-sert>

Développement Front et Framework JS - JQuery

Le JavaScript sert à améliorer le visuel de la page web. C'est un complément au CSS. Il participe à la spécialisation du travail entre Frontend et Backend.

Des bibliothèques-framework existent pour faciliter le travail et découpler les usages (la logique métier) de la technique (le DOM).

Jquery : la plus populaire aujourd'hui. Lancée en 2006. Pour dynamiser les pages web.

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

Développement d'application : Angular

Avec du HTML et du JavaScript on peut développer des applications côté client. Des bibliothèques et/ou frameworks permettent de faciliter le travail.

➤ **AngularJS**

C'est une autre bibliothèque-framework JavaScript créé en 2009 chez Google. Pour développer des applications côté client.

https://www.w3schools.com/angular/angular_intro.asp

➤ **JointJS**

C'est un site qui propose des bibliothèques payantes permettant de développer des applications comme par exemple un modèleur UML :

<http://resources.jointjs.com/demos/kitchensink>

L'intérêt ici est d'avoir un outil en ligne. On peut tester le modèleur UML.

Il y a là un champ de développement commercial ou libre considérable !

<https://www.jointjs.com/opensource>

Désactiver JavaScript

On peut sur chaque navigateur, désactiver JS. Le JavaScript ne se substitue donc pas aux vérifications qu'il faut faire côté serveur.

Beaucoup de sites ne pourront pas fonctionner sans JS.

Bonnes pratiques

JS a évolué depuis sa création. Les premiers usages peuvent être aujourd'hui considéré comme obsolètes et relevant de mauvaises pratiques.

Donc il faut faire attention à ne pas copier-coller n'importe quel exemple récupéré sur le web !

Environnement de travail

Editeur orienté front-end

- *Sublime text (ou Notepad ++ ou Bracket ou autre)*

Navigateur

- *Firefox, Chrome, Safari, etc.*

Environnement WAMP

On peut installer les fichiers de test dans un environnement WAMP (dans le répertoire www du serveur local).

Ce n'est pas obligatoire puisqu'on fonctionne, dans un premier temps (sans AJAX), uniquement côté client.

« Bac à sable » (coder en ligne) : à éviter !

Pour faire des tests éventuellement. Pas très intéressant. Mieux vaut prendre le contrôle sur son fichier.

<https://jsfiddle.net/>

<http://jsbin.com/>

<http://codepen.io/>

Balise <script>

Le code JS se place entre les balises <script> et </script>

Où mettre la balise <script> ?

On peut placer des balises <script> peuvent être placées dans le <head> ou dans le <body>.

En général, on ne met qu'une balise <script> par fichier HTML.

Aujourd'hui, on préfère les placer à la fin du <body>, juste avant le </body> : en effet pour bien s'exécuter, le JS doit d'abord avoir chargé la page HTML (mais l'affichage se fait après le chargement complet, donc après l'exécution du code JavaScript).

Bonne pratique si on met la balise <script> dans le <head>

```
<script>
  window.onload=fonction() {
    code JS
  }
</script>
```

On utilise le « window.onload » pour dire que le code JS sera exécuté après que la page ait été chargée. On met le code dans la fonction, entre les accolades.

Meilleure pratique : code JS avant le </body>

```
<body>
  code de la page HTML

  <script>
    code JS
  </script>
</body>
```

En mettant le code avant le </body>, on n'a plus besoin de mettre le « window.onload ». Quand le code JS s'exécute, la page HTML a été chargée.

Bonne pratique ultime : inclusion d'un fichier externe

```
<body>
  code de la page HTML

  <script src="script.js"></script>
</body>
```

On peut inclure un fichier contenant du code JS avec l'attribut « src » dans la balise <script>. On fera ça dans tous nos exemples.

Exemples

Chargez les fichiers

Les exemples du cours sont dans un fichier zip fournis avec l'article du cours.

- JavaScript_01_exemples_01_bases_et_tour.zip

Téléchargez le zip et dészippez-le. On obtient un répertoire :

tour JavaScript_01_exemples_01_bases_et_tour

Mettez ce répertoire : dans un dossier « Partie_1 » que vous aurez mis dans un dossier JavaScript.

Ce dossier JavaScript peut être mis où vous voulez sur votre machine. Vous pouvez le mettre dans le répertoire web « www » du serveur WAMP mais ce n'est pas utile. Les fichiers HTML contenant du JavaScript peuvent être exécutés sans environnement serveur.

Ce dossier contient les codes des exemples de 01 à 11.

01 - Hello world !

➤ HTML

```
<body>
  Ce qu'on veut dans la page
  A la fin, une balise script
  <script>
    alert('Hello world!');
  </script>
</body>
```

La partie JavaScript est écrite dans la balise script.

On utilise la fonction alert.

02 - balise script seule

➤ HTML

```
<script>
  alert('Hello world!');
</script>
```

Ca fonctionne aussi sans HTML, évidemment ! C'est du HTML !

03 - Hello world ! fichier JS

➤ HTML

```
<body>
  <script src="hello.js"></script>
</body>
```

➤ JavaScript : hello.js

```
alert('Hello world!');
```

On peut aussi coder le JavaScript dans un fichier séparé.
C'est plus lisible.

Exercice : testez ces trois fichiers

Quelques références

Mozilla

*** <https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide>
<https://developer.mozilla.org/fr/docs/Web/JavaScript>
<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference>

Eloquent JavaScript

*** sommaire : <http://fr.eloquentjavascript.net/contents.html>

En anglais : <http://eloquentjavascript.net>

En français : <http://fr.eloquentjavascript.net>

w3school

<https://www.w3schools.com/js/default.asp>

Communauté JavaScript à Paris

<http://parisjs.org>

Console Javascript

Le problème

Comment tester notre code ? Comment afficher le contenu des variables qu'on va manipuler sans modifier pour autant la page web ?

Accès à la console JavaScript

Dans le navigateur, on peut afficher une console JavaScript

➤ *Firefox*

Outils / Développement Web / Console Web : onglet Console

➤ *Chrome*

Afficher / Option pour les développeurs / Console JavaScript : onglet Console

➤ *Safari*

Développement / Afficher la console JavaScript : onglet Console

➤ *Etc.*

Raccourcis clavier : alt-cmd-i

Outils de développement : alt-cmd-i

Taper du code directement

On peut taper du code directement, comme en python, ou dans une console SQL

```
> a=3
> a
> b=a+a
> b
> alerte(b)
etc.
```

Afficher du texte dans le mode console

➤ *console.log()*

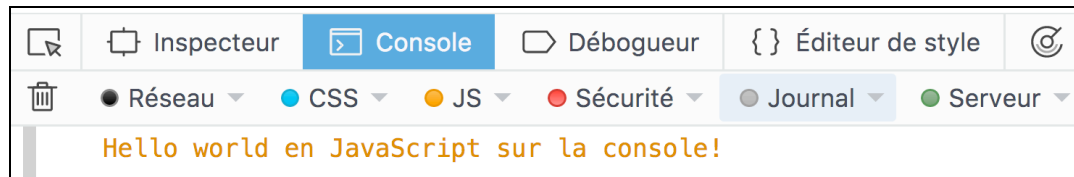
L'instruction « console.log() » permet d'afficher du texte dans le mode console à partir de code JavaScript.

```
console.log("Hello world en JavaScript sur la console!");
```

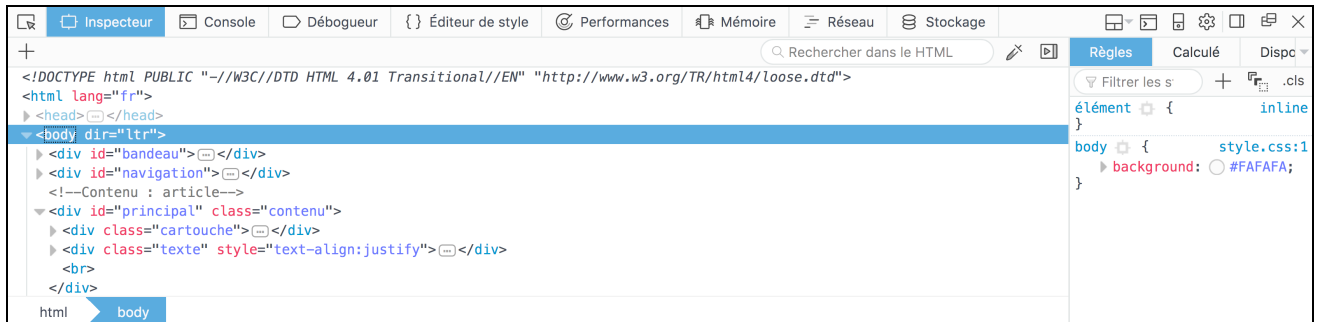
On peut passer des variables en paramètre et n'importe quelle expression.





➤ *Mode console / Journal*

Pour voir l’affichage des instructions console.log, on ouvre le mode console / Journal (les onglet Réseau, CSS, etc. ne sont pas toujours affichés).



Fonctionnalités de l’environnement dans Firefox



- **Inspecteur** : pour parcourir le code HTML
- **Console** : pour avoir les erreurs et le « **mode console** »
- **Débogueur** : pour mettre des points d’arrêt dans le code
- **Editeur de style** : pour voir le CSS. On peut le changer pour voir les résultats
-  : pour faire apparaître la console dans l’inspecteur, par exemple.
-  : pour présenter les outils verticalement.
-  : pour présenter les outils dans une fenêtre à part.
-  : pour accéder aux configuration : par exemple le thème sombre.

➤ **HTML : index.html**

```
<script src="script.js"></script>
```

➤ **JavaScript : script.js**

```
v1 = 5;
v2 = 10;
resultat = v1+v2;
alert(v1 + ' + ' + v2 + ' = ' + resultat);
texte1 = 'Le double du résultat est : ' + resultat*2;
alert(texte1);

// on n'est pas obligé de mettre var
var v3, v4, resultat2, texte2;
v3 = prompt('Entrez le premier chiffre :');
v4 = prompt('Entrez le second chiffre :');
resultat2 = parseInt(v3) + parseInt(v4);
alert(v3 + ' + ' + v4 + ' = ' + resultat2);
texte2 = 'Le double du résultat est : ' + resultat2*2;
```

- alert pour afficher une variable dans une fenêtre
- prompt pour lire une variable dans une fenêtre
- parseInt() permet de transformer du texte en entier. Il existe aussi une fonction parseFloat(). On peut aussi ajouter parseFloat(...).toFixed(2) pour limiter à deux chiffres après la virgule.
- On peut additionner avec le signe « + » ou faire une concaténation.

http://www.w3schools.com/js/js_output.asp

➤ **HTML : index.html**

```
<script src="cours2.js"></script>
```

➤ **JavaScript : script.js**

```
function fonction_v1(){
    v1 = 5;
    v2 = 10;
    resultat = v1+v2;
    alert(v1 + ' + ' + v2 + ' = ' + resultat);
    texte = 'Le double du résultat est : ' + resultat*2;
    alert(texte);
}
fonction_v1(); // appel à la fonction

function fonction_v2(v1, v2){
    resultat = v1+v2;
    alert(v1 + ' + ' + v2 + ' = ' + resultat);
    texte = 'Le double du résultat est : ' + resultat*2;
    alert(texte);
}

v1 = prompt('Entrez le premier chiffre :');
v2 = prompt('Entrez le second chiffre :');

fonction_v2(parseInt(v1), parseInt(v2)); // appel à la fonction
```

On reprend le calcul précédent et on écrit des fonctions.

Une première fonction sans paramètre.

Une deuxième fonction avec paramètre.

06 - document.write() et test – A tester

➤ HTML : index.html

```
<script src="script.js"></script>
```

➤ JavaScript : script.js

```
document.write('<h1>Partie JavaScript : document.write écrit à la  
fin de la page</h1>');  
  
prenom = prompt('Entrez votre prénom :');  
age = prompt('Entrez votre age :');  
  
age = parseInt(age);  
  
if(age>25){  
    document.write('Désolé '+prenom+'<br>');  
    document.write('Vous n\'avez pas droit à la carte Jeune');  
}  
else{  
    document.write('Bonjour '+prenom+'<br>');  
    document.write('Vous pouvez bénéficier de la carte Jeune');  
}
```

- document.write permet d'écrire à la fin de la page HTML

07 – document.getElementById() et boucle for – A tester

➤ HTML : index.html

```
<body>  
    <h1>Début de la page</h1>  
    <div id="resultats"></div>  
    <h1>Fin de la page</h1>  
  
    <script src="script.js"></script>  
</body>
```

➤ JavaScript : cours.js

```
balise=document.getElementById('resultats');  
balise.innerHTML = '<h3>Table de 5 : </h3>';  
  
for(i=1; i<11; i++){  
    balise.innerHTML += '5 * '+i+' = '+5*i+'<br>';  
}
```

- balise=document.getElementById('resultats') permet de récupérer la balise dont l'id vaut "resultat".
- balise.innerHTML est le texte qu'on place dans la balise
- Le « += » permet d'ajouter du texte dans la balise
- La boucle for est une boucle standard.

08 – Button : programmation événementielle – A tester

L'objectif est de déclencher du code JavaScript en cliquant sur un bouton.

➤ *HTML : index.html*

```
<body>
  <h1>Début de la page</h1>
  <fieldset>
    <p>Zone de démonstration</p>
    <p id="demo"></p>
  </fieldset>

  <button onclick="document.getElementById('demo').innerHTML
= 'bien cliqué sur le bouton 1'">Cliquez moi 1!!!</button>

  <button onclick="boutonTest()">Cliquez moi 2 !!!</button>
<h1>Fin de la page</h1>

<script src="script.js"></script>
</body>
```

➤ *JavaScript*

```
function boutonTest() {
  document.getElementById('demo').innerHTML =
  '<p style="background-color:aqua"> bien cliqué sur le bouton
2</p>'
}
```

- La balise <button> sert à créer un bouton qui actionnera du code JavaScript.
- On peut ajouter des attributs dans la balise <button> qui définisse un événement à l'origine de l'exécution d'un code JavaScript. Ici l'événement « onclick ». La valeur de l'attribut, c'est du code JavaScript à exécuter quand l'événement est déclenché.
- Pour la première balise <button>, le code JavaScript est un « document.getElementById »
- Pour la deuxième balise <button>, le code JavaScript est l'appel à la fonction boutonTest()
- La fonction boutonTest() est défini dans le fichier JavaScript qui est inclus dans la balise <script>
- Le bilan est que la page HTML contient du code JavaScript directement dans la balise <button>

09 – onclick, onmouseover, on mouseout – A tester

L'objectif est de déclencher du code JavaScript en fonction de certains événements (onclick, onmouseover, etc.), ces événements pouvant s'appliquer à n'importe quelle balise.

L'objectif est aussi d'ajouter ces événements dans le code JavaScript et pas dans la page HTML.

➤ *HTML : index.html*

```
<body>
  <h1>Début de la page</h1>

  <p id="p1"> Premier paragraphe de test : cliquez moi pour
changer la couleur de fond</p>

  <p id="p2"> Deuxième paragraphe de test : passez sur moi pour
changer la couleur de fond</p>

  <p id="p3"
onclick="document.getElementById('p3').style.backgroundColor='yel
low';">
    Troisième paragraphe de test : cliquez moi pour changer la
couleur de fond
  </p>

  <h1>Fin de la page</h1>

  <script src="script.js"></script>
</body>
```

➤ *JavaScript*

```
// Paragraphe p1 :
baliseP1 = document.getElementById('p1');
baliseP1.onclick = function(){
  baliseP1.style.backgroundColor='aqua';
}

// Paragraphe p2 :
baliseP2 = document.getElementById('p2');
baliseP2.onmouseover = function(){
  baliseP2.style.backgroundColor='yellow';
}
baliseP2.onmouseout=function(){
  baliseP2.style.backgroundColor='';
}
```

- Dans le HTML, dans le premier <p> est repris en JavaScript.
- Dans le JavaScript : on récupère la balise p1.
- Sur cette balise on met dans l'attribut « onclick » une fonction qui définit l'action à réaliser.
- L'action à réaliser consiste à modifier le style.backgroundColor de la balise.
- Notez que le nom reprend celui du CSS : background-color, mais en « Camel Case » (pas de tiret, majuscule sur le deuxième mot).

- On fait la même chose sur la balise p2, mais cette fois sur les événements « onmouseover » et « onmouseout »
- Dans le code HTML, sur un paragraphe p3, on fait directement ce qu'on a fait sur le paragraphe p1.

10 – console – A tester

➤ HTML : index.html

```
<script src="script.js"></script>
```

➤ JavaScript : script.js

```
console.log("Tests de console.log");

a=5 ; b=3 ;
console.log(a + ' x ' +b + ' = ' + a*b) ;

function direBonjour(prenom) {
    var message = "Bonjour, " + prenom + " !";
    return message;
}

console.log(direBonjour("Baptiste"));
console.log(direBonjour("Sophie"));

console.log("Au revoir !");
```

console.log permet d'écrire dans la console de log. C'est utile pour déboguer son programme.

11 – Paramètres en sortie : fonction d'inversion – A tester

➤ HTML : index.html

```
<script src="script.js"></script>
```

➤ JavaScript : script.js

```
console.log("Tests de console.log");

////////////////////////////////////
console.log("INVERSION DE NUMBERS");
function inverser(a, b){
    console.log("Dans la fonction inverser : entrée : a="+a+" - b="+b);
    tmp=a;
    a=b;
    b=tmp;
    console.log("Dans la fonction inverser : sortie : a="+a+" - b="+b);
}

a=5;
b=10;
console.log("Avant inverser : a="+a+" - b="+b);
inverser(a,b);
console.log("Après inverser : a="+a+" - b="+b);

console.log("Avant inverser, en 'dur' : a=5, b=10");
```

```

inverser(5,10);

////////////////////////////////////
console.log("INVERSION DE TABLEAU");
function inverserTableauDe2(tab){
    console.log("Dans la fonction inverserTableauDe2 : entrée :
"+tab);
    tmp=tab[0];
    tab[0]=tab[1];
    tab[1]=tmp;
    console.log("Dans la fonction inverserCouple : sortie : "+tab);
}

tab = [2,4];
console.log("Tableau de départ : "+tab);
inverserTableauDe2(tab)
console.log("Tableau après inversion : "+tab);

////////////////////////////////////
console.log("INVERSION DE STRUCTURE");
function inverserCouple(couple){
    console.log("Dans la fonction inverserCouple : entrée :
a="+couple.a+" / b="+couple.b);
    tmp=couple.a;
    couple.a=couple.b;
    couple.b=tmp;
    console.log("Dans la fonction inverserCouple : sortie :
a="+couple.a+" / b="+couple.b);
}

couple = {
    a:2,
    b:4
}
console.log("Structure de départ : a="+couple.a+" /
b="+couple.b);
inverserCouple(couple)
console.log("Structure après inversion : a="+couple.a+" /
b="+couple.b);

////////////////////////////////////
console.log("INVERSION D'OBJET");
objet={
    a:5,
    b:10,
    toString:function(){
        return "a="+this.a+" - b="+this.b;
    },
    inverserCouple:function(){
        console.log("Dans la fonction objet inverserCouple : entrée :
a="+this.a+" / b="+this.b);
        tmp=this.a;
        this.a=this.b;
        this.b=tmp;
        console.log("Dans la fonction objet inverserCouple : sortie :
a="+this.a+" / b="+this.b);
    }
}

```

```
    }  
  }  
  console.log("Objet de départ : "+objet.toString());  
  inverserCouple(objet)  
  console.log("Objet après inversion classique:  
  "+objet.toString());  
  objet.inverserCouple();  
  console.log("Objet après inversion classique:  
  "+objet.toString());
```

Organisation du cours

Programmation impérative

Le JS permet de faire de la programmation impérative classique : variables, tests, boucles, fonctions, etc.

DOM

Il permet ensuite d'interagir avec la page HTML : il utilise pour cela une API : le DOM.

Programmation événementielle

Il permet aussi de gérer des événements : c'est de la programmation événementielle.

AJAX – XML - JSON

Enfin, il permet de communiquer avec le serveur : c'est la partie AJAX avec les formats de communication XML et surtout JSON aujourd'hui.

3 types simples :number, string, boolean

Le type d'une valeur détermine son rôle et les opérations qui lui sont applicables.

Les principaux types de bases du JS sont : nombre, chaîne de caractères, booléen

Type [number](#) : entier ou réel. Les réels s'écrivent avec un « . »

Type [string](#) : chaîne de caractère : entre guillemets ou apostrophes

Type [boolean](#) : true et false, en minuscules

1 type complexe : object

JS permet de définir des structures, des tableaux et des objets.

Tous les types complexes sont des « object ».

opérateur typeof

```
typeof 1 ; typeof(1) // number
typeof 1.1 ; typeof(1.1) // number
a=5 ; typeof a ; typeof (a) // number
typeof 'hello' ; // string
typeof true ; // boolean
typeof (1==1) ; // boolean
```

les structures, les tableaux et les objets sont de type « object »

```
typeof [1, 2] // object // tableau de 2 entiers
tab=[1, 2] ;
typeof tab // object // tableau de 2 entiers
typeof tab[0] // number
```

```
typeof {nom :'toto', age :15} // object // structure à 2 attributs
personne= {nom :'toto', age :15} ;
typeof personne // object // structure à 2 attributs
typeof personne.nom // string
```

Opérations de base

Exemple

4*3 : affiche le résultat

la division par 0 renvoie « Infinity »

« bonjour » ou 'bonjour' : affiche « bonjour »

« bonjour \n tout le monde » : le \n est un passage à la ligne

« bon »+ «jour » : affiche « bonjour »

« bonjour »[0] : vaut « b »

« bonjour »[3] : vaut « j »

« bonjour ».length : vaut 7

« bonjour ».toUpperCase : vaut « BONJOUR »

Principe

Chaque type permet d'accéder à des opérateurs et à des méthodes.

Type Number

toFixed(x) Formats a number with x numbers of digits after the decimal point

toString() Converts a number to a string

a.toFixed(2)

etc.

https://www.w3schools.com/jsref/jsref_obj_number.asp

Type String

substr() Extracts the characters from a string, beginning at a specified start position, and through the specified number of character

concat() Joins two or more strings, and returns a new joined strings

etc.

https://www.w3schools.com/jsref/jsref_obj_string.asp

Type Boolean

toString() Converts a boolean value to a string, and returns the result

https://www.w3schools.com/jsref/jsref_obj_boolean.asp

Affichage : console.log

```
console.log(3)
console.log(a) : affiche a
console.log(a, b)
console.log(st1, st2)
console.log(st1+st2) : concaténation
a=5 ; b=3 ;
console.log(a + ' x ' +b + ' = ' + a*b) ;
etc.
```

Commentaires

```
//
/* */
```


Variable

Présentation

Nom, valeur, type, adresse, signification

Le nom de la variable est constitué de maj, min, chiffre, \$, _ (underscore)

Le type est défini à l'usage

Déclarer une variable

```
var a ;
```

On n'est pas obligé de déclarer les variables. Elles le sont automatiquement quand on fait une affectation.

Affectation

```
a=5 ;
```

Incrémentation

```
a=a+1 ;
```

```
a+=1 ;
```

```
++a ;
```

`a++` ; // post incrémentation : si on affiche en même temps, c'est la valeur de a avant l'incrément qui s'affiche.

```
a=10 ;
```

```
console.log(a++) ; // affiche 10
```

```
console.log(a) ; // affiche 11
```

Utilisation d'une variable : lecture de sa valeur

si on utilise une variable déclarée mais sans valeur : undefined

si on utilise une variable non déclarée : reference error, can't find variable

Conversions de types

On peut changer le type d'une variable en lui donnant une nouvelle valeur.

```
a=5 ; ...
```

```
a=«texte »
```

Expression et évaluation d'une expression

Principes d'évaluation

`a=expression ;`

L'expression est évaluée : elle produit une valeur qui a un certain type. Ici ce quelque chose est affectée à « a ».

`console.log(expression) ;`

L'expression la plus simple, c'est une valeur ou une simple variable.

Les expressions peuvent être complexes en intégrant des opérateurs, des parenthèses, des fonctions.

Evaluation en fonction du contexte

Les expressions sont évaluées en fonction du type.

Le type est donné en fonction du contexte :

```
a=3;  
b=a+2 // b vaut 5 : a est un entier  
c="resultat="+a+2 // c vaut : "resultat=32" : a est une string
```

Ici c vaut : "resultat = 32"

En effet, le + est un + de concaténation : a est alors considéré comme une string

Type et expression booléenne

&&, ||, true, false

==, !=, <, <=, >, >=

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Les différents tests d égalité](https://developer.mozilla.org/fr/docs/Web/JavaScript/Les_différents_tests_d_égalité)

===, !==

L'égalité faible == fait une conversion de type. L'égalité forte === n'en fait pas.

« 17 » == 17 est true

« 17 » === 17 est false

if, else

```
if (condition) {  
  }  
else {  
  }
```

else if

```
if (condition) {  
  }  
else if (condition) {  
  }  
etc.
```

switch case

```
switch (variable) {  
  case valeur :  
    instructions ;  
    break ;  
  default :  
    instructions ;  
}
```

Boucle

while

```
while(condition){  
  instructions ;  
}
```

for

```
for (i = 1; i <= 5; i++) {  
  console.log(i);  
}
```

Déclaration d'une fonction

Une fonction est un regroupement d'instructions qui réalisent une tâche donnée.

Une fonction rend le code plus modulaire.

Une fonction est constituée d'une en-tête et d'un corps.

Une fois écrite, une fonction peut être appelée depuis n'importe quel emplacement du programme.

Une fonction peut recevoir des informations sous la forme de paramètres.

Une fonction peut renvoyer ou non une valeur de retour.

Portée des variables : variables locales et variables globales

https://www.w3schools.com/js/js_scope.asp

➤ *Variables globales*

Toutes les variables déclarées en dehors des fonctions sont des variables globales : elles sont utilisables **partout dans le code après leur déclaration, même dans les fonctions.**

➤ *Variables locales : dans les fonctions*

Les fonctions accèdent aux variables globales.

Toutes les variables déclarées avec un « var » dans les fonctions sont locales aux fonctions : elles ne sont pas utilisables en dehors des fonctions.

Les variables non déclarées avec un var sont globales, qu'elles aient été créées avant ou non : attention, c'est une source d'erreur ! **Il faut donc bien déclarer les variables locales des fonctions avec un var.**

Si une fonction déclare une variable locale qui existait déjà comme variable globale, la variable globale n'est plus visible dans la fonction. Elle ne sera pas affectée par les modifications qu'on fera sur la variable locale.

➤ *Le bon usage*

N'utiliser que des variables locales dans les fonctions : il faut donc toutes les déclarer avec un var.

Eviter les variables globales autant que possible.

Paramètres en sortie des fonctions : exemple 11

Principes

Une fonction retourne une valeur de n'importe quel type : number, string, boolean ou object (tableau, structure, objet et composés)

Les paramètres de type number, string, boolean sont toujours en entrée, jamais en sortie. Ils sont passés par valeur.

Les paramètres de type object toujours en entrée et en sortie. Ils sont passés par référence.

exemple

La fonction ci-dessous à 2 entiers en sortie.

C'est impossible. Donc l'inversion n'est pas effective après l'appel de la fonction.

```
function inverser(a, b){
  var tmp=a;
  a=b;
  b=tmp;
}
a=5;b=10;
inverser(a,b); // ne fait rien !
console.log(a) // vaut 5
console.log(b) // vaut 10
```

La fonction ci-dessous à une structure à 2 entier en paramètre : c'est une structure, donc un objet donc il est en entrée-sortie.

```
function inverserCouple(couple){
  tmp=couple.a;
  couple.a=couple.b;
  couple.b=tmp;
}
couple={a:5,b:10};
inverserCouple(couple);
console.log(couple.a) // vaut 10
console.log(couple.b) // vaut 5
```

Tester l'exemple : 11-parametres en sortie

JavaScript output

http://www.w3schools.com/js/js_output.asp

Toutes les possibilités d’affichage dans la page sont présentées :

➤ ***Affichage de log***

`console.log()`

➤ ***Affichage d’une fenêtre avec texte et champs de saisie***

`prompt()`

➤ ***Affichage d’une fenêtre d’alerte, avec ou sans confirmation***

`alert()` ou `window.alert()` sont équivalent : pas de confirmation

`confirm()` : permet d’annuler

➤ ***Affichage dans la page HTML :***

<https://developer.mozilla.org/fr/docs/Web/API/Document/write>

`document.write(5 + 6);`

➤ ***Affichage dans une page HTML vierge sur clic d’un bouton***

`<button onclick="document.write(5 + 6)">Try it</button>`

A noter qu’entre les guillemets du onclick on peut mettre plusieurs instructions séparées par des ;

`button onclick="resultat=5+6 ; document.write(resultat)">Try it</button>`

➤ ***Affichage dans un élément HTML d’un « id » css donné :***

`document.getElementById("demo").innerHTML = 5 + 6;`

fonctions mathématiques

http://www.w3schools.com/jsref/jsref_obj_math.asp

http://www.w3schools.com/js/js_math.asp

```
console.log(Math.min(4.5, 5)); // Affiche 4.5
```

```
console.log(Math.random()); // Affiche un nombre aléatoire entre 0 et 1
```

manipulation de chaînes : String

➤ *Présentation de toutes les méthodes :*

http://www.w3schools.com/jsref/jsref_obj_string.asp

➤ *Exemples*

http://www.w3schools.com/js/js_strings.asp

```
mot= « test » ;  
mot.length ;  
mot.toLowerCase() ; mot.toUpperCase() ;  
mot.charAt(0) ; mot[0] ;  
mot.indexOf(«A »)  
etc.
```

manipulation de dates

➤ *Présentation de toutes les méthodes :*

https://www.w3schools.com/jsref/jsref_obj_date.asp

```
var d = new Date();  
d.getDate() : retourne le jour du mois  
d.getDay() : retourne le jour de la semaine : 0 pour dimanche, 1 pour lundi  
getHour  
setDate, setHours,  
etc.
```

➤ *Exemples*

http://www.w3schools.com/js/js_dates.asp

etc.

Exercices – Série 1

1 – Calculs sur des figures

Ecrire une page qui permet de saisir la largeur et la longueur d'un rectangle puis qui affiche son périmètre et sa surface et qui permette de saisir le rayon d'un cercle et qui affiche le périmètre du cercle. Le résultat doit avoir 2 chiffres après la virgule.

Calculs sur des figures

Cliquez sur le bouton pour calculer le périmètre d'un cercle :

Cliquez sur le bouton pour calculer le périmètre d'un rectangle :

On fournit 2 boutons à l'utilisateur. Il peut saisir les valeurs. Le résultat s'affiche en dernière ligne de la page, dans une fenêtre d'alerte et dans la console de log.

2 – Jour de la semaine

Sur le même principe que l'exercice précédent, écrire une fonction qui affiche le jour de la semaine.

Regardez ici le fonctionnement de la fonction `getDay` :

https://www.w3schools.com/jsref/jsref_getday.asp

Le retour d'un `getDay` sur une date vaut 0 pour dimanche, 1 pour lundi, etc.

Pour cela, on se dotera d'une fonction qui renvoie le jour de la semaine à partir du chiffre correspondant au résultat du `getDay()`.

On affiche les résultats ainsi :

Affichage du jour de la semaine

Jour de la semaine

On est Lundi

L'encadrement est une balise `<fieldset>`

3 – Table de multiplication

1. Ecrire une page HTML avec du JS qui permet d'obtenir le résultat suivant en cliquant sur le bouton :

Table de multiplications

Cliquez moi

- 7 x 1 = 7
- 7 x 2 = 14
- 7 x 3 = 21
- 7 x 4 = 28
- 7 x 5 = 35
- 7 x 6 = 42
- 7 x 7 = 49
- 7 x 8 = 56
- 7 x 9 = 63
- 7 x 10 = 70
- 7 x 11 = 77
- 7 x 12 = 84
- 7 x 13 = 91
- 7 x 15 = 105
- 7 x 20 = 140
- 7 x 25 = 175

On pourra saisir la valeur 7 ou bien n'importe quelle autre valeur.

Principe de résolution :

On va construire le code HTML à produire dans une variable appelée : `innerHTML`

Quand elle est entièrement construite : `` etc. `` on écrira une instruction du type de :

```
balise.innerHTML=innerHTML;
```

2. Variante difficile : écrire une version qui permette de surligner en jaune chaque ligne quand on passe dessus avec la souris et que ça revienne sans couleur quand la souris s'en va.

Principe de résolution :

Au lieu d'avoir une balise « li » simple : ``, il s'agit de construire dynamiquement un li ressemblant à :

```
<li id="li1"
onmouseover=
"document.getElementById('li1').style.backgroundColor='yellow';"
onmouseout=
"document.getElementById('li1').style.backgroundColor='';"
>
```

4 – Jeu des allumettes (jeu de Marienbad)

Présentation du jeu

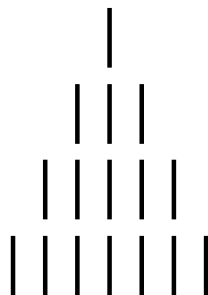
➤ *Test du jeu*

Par exemple :

http://therese.eveilleau.pagesperso-orange.fr/pages/jeux_mat/textes/marienbad.htm

➤ *Description du jeu*

On a 4 lignes d'allumettes avec 1, 3, 5 et 7 allumettes.



Le but du jeu est de ne pas ramasser la dernière allumette : qui prend la dernière perd.

Le jeu se joue à 2. A son tour, chaque joueur peut prendre autant d'allumettes qu'il veut mais sur une seule ligne.

➤ *Comment gagner à tous les coups ?*

D'abord, il ne faut pas commencer.

Ensuite, il faut laisser le jeu dans une configuration particulière expliquée maintenant.

Chaque ligne contient un nombre d'allumettes : 1, 3, 5 et 7 au départ. Il faut traduire ce nombre en binaire.

Situation de départ :

Nombre d'allumettes	Nombre d'allumettes en binaire		
1	0	0	1
3	0	1	1
5	1	0	1
7	1	1	1
Total en base 10 de chaque colonne binaire :	2	2	4

Le joueur 1 qui commence est dans une configuration telle que le total en base 10 de chaque colonne binaire est un nombre pair. C'est cette configuration qui fait que le joueur va perdre, si l'adversaire ne fait pas d'erreur.

Quand le joueur 1 va jouer, quoi qu'il fasse, il laissera un ou plusieurs totaux impairs.

Le joueur 2 devra jouer de telle sorte que chaque total soit à nouveau un nombre pair.

➤ *Une exception*

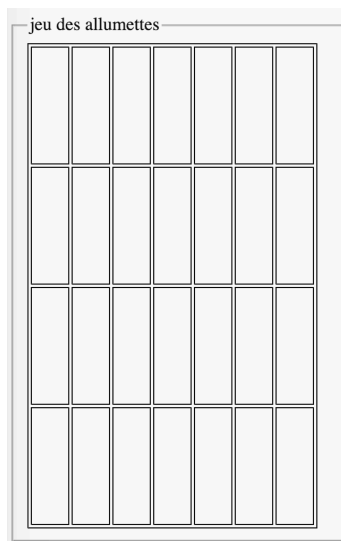
Il ne faut pas laisser 1 allumette sur les 4 lignes, ni 1 allumettes sur 2 lignes. Le total est pair.
Mais celui qui joue va alors gagner !

Etapes de résolution

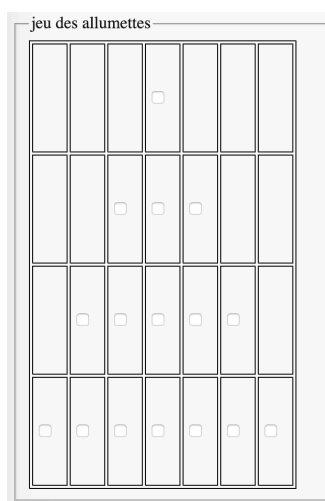
➤ *Etape 1 : HTML : Affichage d'un tableau de 4 lignes et 7 colonnes, sans CSS*



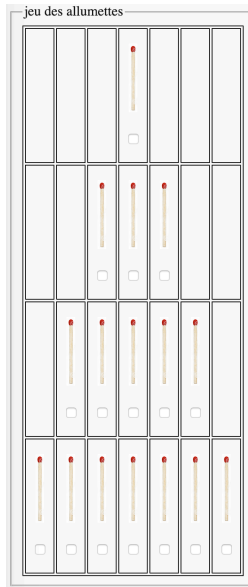
➤ *Etape 2 : HTML : Affichage d'un tableau de 4 lignes et 7 colonnes, avec CSS minimal*



➤ *Etape 3 : HTML : Ajout d'un input checkbox*



➤ **Etape 4 : HTML : Ajout d'une image d'allumette**



➤ **Etape 5 : JavaScript : afficher l'état des boutons en console**

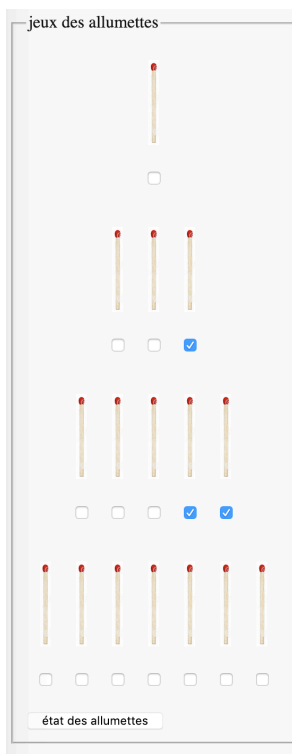
Pas de modification du HTML

JavaScript :

```
console.log("allumetteL1A1 : "+allumetteL1A1.checked);
```

allumetteL1A1, c'est la variable qui correspond à la balise. C'est un objet avec entre autre un attribut « checked ».

➤ **Etape 6 : JavaScript : afficher l'état des boutons sur demande**



JavaScript :

```
document.getElementById('etat').onclick = function(){
```

➤ **Etape 7 : JavaScript : effacer les allumettes : fonction « eval »**

Le bouton « état » devient un bouton « effacer ».

Pour effacer, on commence par supprimer le « src » de l'image :

```
if(allumetteL1A1.checked){ allumetteL1A1img.src=""; }
```

On va aussi faire une deuxième version qui boucle sur toutes les allumettes.

On construit le nom de l'allumette dans une variable : par exemple, la variable « allumette » vaudra « allumetteL2A3 ».

Ensuite, on utilise la **fonction « eval »** :

```
if(eval(allumette).checked){
```

Cette fonction permet de faire comme si une chaîne de caractères était un nom de variable.

➤ **Etape 8 : on efface tout le TD en les numérotant**

HTML

```
<td id="tdL1A1">
  <p></p>
  <p><input type="checkbox"></p>
</td>
```

JavaScript

```
baliseInput=baliseTD.getElementsByTagName("input")
if(baliseInput[0].checked){
  baliseTD.style.display="none"
}
```

➤ **Etape 9 : on ajoute un RAZ**

Le RAZ permet de recommencer la partie.

➤ **Etape 10 : on fait en sorte qu'à chaque nouveau chargement, on reparte avec un RAZ**

JavaScript

```
window.onload=function(){
  raz()
}
```

➤ **Etape 11 : on évite de pouvoir cliquer sur deux lignes**

➤ **Etape 12 : on affiche « gagné » ou « perdu » en bas de page**

JAVASCRIPT – TABLEAUX ET STRUCTURES

Installation des fichiers de tests

Dans le cours :

Les exemples sont présentés dans un chapitre en vert.

Les exercices à faire sont présentés dans un chapitre en jaune.

Les exemples du cours sont dans un fichier zip fournis avec l'article du cours.

- JavaScript_01_exemples_02_tableaux_structures.zip

Chargez ces fichiers et mettez-les :

Soit dans un dossier « Partie_1 » que vous aurez mis dans un dossier JavaScript.

Ce dossier JavaScript peut être mis soit où vous voulez sur votre machine, soit dans le répertoire web « www » du serveur WAMP.

Bases du code – 2 – Tableaux et Structures

Tableaux : exemple 1

Présentation

http://www.w3schools.com/jsref/jsref_obj_array.asp

- Les tableaux permettent de regrouper des données
- On peut mettre tout ce qu'on veut dans un tableau : number, string, booléen, objet
- On peut mélanger toutes sortes de types de données dans un même tableau.

Déclaration

- `var tab = [];` // Création d'un tableau vide
- `var tab = [5, 10, 12];` // Création d'un tableau plein
- On peut aussi écrire : `tab=[]` sans le var, ou `tab=new Array()` avec ou sans parenthèses, comme en POO. Cette pratique est à éviter.

Accès aux éléments

- Comme les caractères d'une chaîne, les éléments d'un tableau sont identifiés par un indice débutant à zéro : `tab[0]`, `tab[1]`
- `tab[4]=10;`
- On ne peut pas écrire `tab[0]=5` si le tableau n'a pas préalablement été déclaré.
- On peut mettre des indices négatifs dans le tableau.

length

- `tab.length` : retourne la position + 1 du dernier élément : le nombre d'éléments. Les éléments d'indices négatifs ne sont pas pris en compte.

Tous les éléments du tableau

- `tab` : liste toutes les valeurs, sauf les éléments d'indice négatif.
- `for(i=0 ; i< tab.length ; i++) console.log(tab[i]) ;`
- `for(key in tab) console.log(tab[key]) ;`

Trous dans le tableau !

- On peut mettre l'index qu'on veut : on n'est pas obligé de tout remplir. Par exemple si le contenu de `tab` s'affiche ainsi : `[5, 3, 5: 9, 8:2]`, c'est que `tab[0]=5`, `tab[1]=3`, `tab[5]=9`, `tab[8]=2`.
- `var tab=[5, 3] ; tab[5]=9 ; tab[8]=2 ;` produit le tableau précédent.
- Attention : `tab.length`, ne donne pas le nombre d'éléments. Dans le tableau précédent, c'est 9.

Fonctions de manipulation du tableau

https://www.w3schools.com/jsref/jsref_obj_array.asp

- `tab.includes(5) // true`
- `tab.indexOf(5) // 0` : la position de 5 dans le tableau
- `delete tab[0];` // supprime l'élément 0 qui est alors `undefined`.
- `tab.push(valeur)` : ajoute un élément après celui de l'index le plus élevé.
- `elt = tab.pop()` : sort du tableau l'élément ayant l'index le plus élevé. Sa valeur passe dans `elt`.
- `tab.sort()` : attention, c'est un tri alphabétique : 10 est avant 2 !

https://www.w3schools.com/js/js_array_sort.asp

Type

- `typeof tab` : retourne « object » : un tableau est un objet (au sens POO)
- `Array.isArray(tab)` : retourne « true » si c'est un tableau
- `tab instanceof Array` : retourne « true » si c'est un tableau

Application

Testez l'exemple 1

Structure ou Objet en JS : exemple 2

Présentation

- Une structure, c'est une variable qui contient des champs (ou propriétés ou attributs) qui peuvent contenir toute sorte de valeurs.
- Les structures sont utiles pour décrire des objets avec leurs caractéristiques.
- On accède au champs avec l'opérateur « . »

```
var eleve = {  
  id:1,  
  nom: "toto",  
  note: 15  
};  
eleve.nom; // vaut toto  
eleve.note; // vaut 1  
eleve.note=18; // on modifie la note  
eleve.note; // vaut 18  
  
eleve['note'] = 15 // on modifie la note : autre écriture
```

Tous les éléments du tableau

- eleve : liste tous les champs de la structure, avec les valeurs
- for(key in eleve) console.log(eleve[key]) ;

Méthodes applicables

- typeof eleve retourne object
- eleve.hasOwnProperty("nom") retourne true : pour savoir si une propriété appartient à une structure.

Application

Testez l'exemple 2

Affichez la structure dans la page HTML

Boucles spéciales : exemples 1 et 2

méthode forEach

La méthode `forEach()` permet d'exécuter une fonction sur chaque élément du tableau.

Exemple :

```
var tab = ['a', 'b', 'c'];

tab.forEach(maFonction(elementDuTableau));

function maFonction(elementDuTableau){
    console.log(elementDuTableau);
}
```

On peut aussi coder la fonction dans le `foreach` : on a alors une fonction anonyme (qui n'a pas de nom).

```
var a = ['a', 'b', 'c'];

a.forEach(function(element) {
    console.log(element);
});
```

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/forEach

boucle for / in

La boucle `for / in` permet de récupérer chaque élément d'une structure ou d'un tableau dans une variable.

Exemple avec une structure :

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x] + " ";
}

// text vaut : "John Doe 25"
```

https://www.w3schools.com/jsref/jsref_forin.asp

Tableau de structures : exemple 3

Présentation

On peut mettre des structures dans des tableaux :

```
var eleve1 = {  
    id:1,  
    nom: "toto",  
    note: 15  
};  
var eleve2 = {  
    id:2,  
    nom: "tata",  
    note: 16  
};  
var tabEleves = [eleve1, eleve2];
```

On peut aussi déclarer directement un tableau de structures :

```
var tabEleves = [  
    {  
        id:1,  
        nom: "toto",  
        note: 15  
    },  
    {  
        id:2,  
        nom: "tata",  
        note: 16  
    }  
]
```

Ou bien, compacté :

```
var tabEleves = [  
    { id:1, nom: "toto", note: 15 },  
    { id:2, nom: "tata", note: 16 }  
]
```

Accès aux données

tabEleves[1].nom vaut « tata ».

Exemple

exemple 3 – Tableau de structures

Affichez la structure dans la page HTML

Structure avec fonctions : exemple 4

Présentation

On peut déclarer une ou plusieurs fonctions en même temps qu'une structure.

C'est une première approche de la programmation objet.

Exemple

```
couple={  
  a:5,  
  b:10,  
  toString:function(){return "a="+this.a+" - b="+this.b;}  
}
```

Usages

```
console.log(couple.a) vaut 5  
couple.a = 3  
console.log(couple.toString) vaut "a=3 - b=10"
```

Exemple

exemple 4 – Tableau de structures

Affichez la structure dans la page HTML

Exercices – Série 2

1 – Tableau de notes

Définir en JavaScript un tableau contenant des notes d'étudiants comprises entre 0 et 20.

- Implémenter en JavaScript les fonctions qui permettent de :
 - ✓ afficher un tel tableau de façon standard HTML
 - ✓ savoir combien d'étudiants ont eu plus de 10
 - ✓ calculer la moyenne des notes
 - ✓ connaître la note maximale
 - ✓ rechercher une note particulière
 - ✓ trier le tableau
- Remarque de méthode :
 - ✓ Vous pouvez vous aider de : https://www.w3schools.com/jsref/jsref_obj_array.asp. Pour par exemple : indexOf ou la fonction de tri.
 - ✓ dans les fonctions, on ne fera aucun affichage.
 - ✓ On utilise document.write pour afficher dans la page HTML
 - ✓ On utilise un fichier HTML et un fichier JavaScript. On peut ajouter un peu de CSS pour le tableau.
 - ✓ Dans le fichier JavaScript, à la fin, on a d'abord les fonctions, puis la création du tableau, puis les appels aux document.write. Le fichier JS est comme un contrôleur qui include le modèle (les fonctions), puis « met la colle du contrôleur » (définit le tableau), puis include la vue (les document.write).

➤ *Objectifs à atteindre en terme de présentation :*

Tableau

Tableau de départ :

8	12	9	12	17	18	15	13
---	----	---	----	----	----	----	----

plus de 10 : 6

moyenne : 13.00

max : 18

9 est présent dans le tableau en position 3

Tableau trié :

8	9	12	12	13	15	17	18
---	---	----	----	----	----	----	----

2 - Tableau d'élèves avec des notes – Tri d'une structure

Dupliquer le travail de l'exercice précédent.

Dans l'exercice précédent, ajouter un prénom pour chaque note. On utilise forcément une fonction.

- Mettez à jour toutes les fonctions.
- Ajoutez une fonction de tri par nom et une fonction de tri par note (googler trier un tableau json). Afficher le tableau trié par nom puis trié par note.

➤ *Objectifs à atteindre en terme de présentation :*

Tableau de structures

Tableau de départ :

Prénom	Note
tata	8
tete	12
titi	9
toto	12
tutu	17
tate	18
tati	15
tato	13

plus de 10 : 6

moyenne : 13.00

max : 18

9 est présent dans le tableau en position 3

Tableau trié par notes :

Prénom	Note
tata	8
titi	9
toto	12

etc.

3 - Tableau d'élèves avec des notes – creerEleve


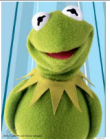

Dupliquer le travail de l'exercice précédent

- Créer une fonction qui permet de créer un élève : `creerEleve(nom, note, photo)`
- Remplir désormais le tableau complet en utilisant cette fonction.
- Créer une fonction qui permet d'afficher un élève de telle sorte qu'elle puisse être utilisée dans la fonction qui affiche tout le tableau (afficher veut dire : retourner un texte avec le code HTML). Mettez à jour la fonction.
- Afficher un élève au choix après les tris.

➤ *Objectifs à atteindre en terme de présentation :*

Tableau de structures avec images

Tableau de départ :

Prénom	Note	
tata	8	
tete	12	
titi	9	

etc.

4 – Jeu de grammaire

Etape 1 : déjà fait : exercice 5

- Mettez votre travail dans un dossier appelé : `JS_bases_exo10_1`
- Définir une variable contenant le verbe chanter.
- Créer une variable qui contient le « nom d'agent » du verbe chanter. Le nom d'agent, c'est celui qui fait l'action. Pour chanter, il s'agit de « chanteur » : le chanteur chante. Pour créer cette variable, supprimer les deux dernières lettres de ce mot et les remplacer par les lettres eur.
- Produire le texte suivant : « chanter : le chanteur chante »

Etape 2

Dupliquer le travail de l'exercice précédent dans un dossier appelé : `JS_bases_exo10_2`

- Généraliser le code précédent en écrivant une fonction traitant n'importe quel verbe du premier groupe.
- Créer un tableau avec des verbes du premier groupe.

- Appliquer la fonction à tous les verbes du tableau.

Etape 3

Dupliquer le travail de l'exercice précédent dans un dossier appelé : JS_bases_exo10_3

- Mettez des verbes du premier et du deuxième groupe dans le tableau (deuxième groupe : verbe finissant en « ir » à l'infinitif et en « issons » à la première personne du pluriel au présent.
- Définir un tableau à six cases contenant les pronoms personnels.
- Définir un tableau contenant les marques de conjugaison au présent de l'indicatif pour les verbes du premier groupe, et un autre pour les terminaisons des verbes du deuxième groupe.
- Écrire une fonction qui décide si le verbe appartient au premier ou au deuxième groupe.
- Mettre un verbe du tableau dans une variable appelée : « verbe »
- Afficher les formes conjuguées de « verbe » au présent de l'indicatif, chaque forme étant précédée du ou des pronom(s) personnel(s) adéquat(s).
- Produire une fonction qui affiche les formes conjuguées de n'importe quel verbe au présent de l'indicatif, chaque forme étant précédée du ou des pronom(s) personnel(s) adéquat(s).
 - ✓ On affichera au début : le nom du verbe et son groupe.
 - ✓ Si ce n'est pas un verbe du premier ou du deuxième groupe, on n'affiche pas la conjugaison.
 - ✓ Si c'est un verbe du premier groupe, on affiche, par exemple : le mangeur mange.
 - ✓ Pour les verbes du premier et du deuxième groupe, on affiche la conjugaison.

5 – Pipotron, Poétron

Pipotron : <http://www.pipotron.free.fr>

Poétron : <http://www.vudansvotreemail.com/poetron-sourire-doc-htm.htm?pos=4&>

Dupliquer le travail du jeu de grammaire dans un dossier appelé : JS_bases_exo10

- Créer un tableau de noms avec un article. Par exemple : le chat, les feuilles, l'arbre, un animal, Bertrand, etc.
- Avec le tableau de verbe, fabriquez une phrase aléatoirement en choisissant un nom pour le sujet, un verbe et un nom pour le complément.
- Attention à la conjugaison !

JAVASCRIPT – POO

Installation des fichiers de tests

Dans le cours :

Les exemples sont présentés dans un chapitre en vert.

Les exercices à faire sont présentés dans un chapitre en jaune.

Les exemples du cours sont dans un fichier zip fournis avec l'article du cours.

- JavaScript_01_exemples_03_objets.zip

Chargez ces fichiers et mettez-les :

Soit dans un dossier « Partie_1 » que vous aurez mis dans un dossier JavaScript.

Ce dossier JavaScript peut être mis soit où vous voulez sur votre machine, soit dans le répertoire web « www » du serveur WAMP.

Programmation objet

Principes

JavaScript permet de faire de la programmation objet.

JavaScript utilise les prototypes pour définir des modèles et partager des attributs entre objets.

Il s'agit d'une spécificité de ce langage.

Structure avec méthode – exemple 1

Structure avec méthode

Une structure ressemble à :

```
var eleve = {  
  id:1,  
  nom: "toto",  
  note: 15  
};
```

Pour un attribut, à la place d'une valeur on peut mettre une méthode :

```
var eleve = {  
  id:1,  
  nom: "toto",  
  note: 15  
  afficher : function(){  
    console.log("nom de l'élève : ", this.nom)  
    console.log("note de l'élève : ", this.note)  
  }  
}
```

```
};
```

A partir de là, la structure ressemble à une classe de la programmation objet.

On va se doter d'une variable de type structure qui servira de prototype pour créer d'autres objets grâce à une fonction spéciale de JS.

Utilisation

```
console.log("Affichage de l'\ élève : ")
eleve.afficher();

eleve.note=18;
console.log("Affichage de l'\ élève après modification de la note
: ")
eleve.afficher();
```

Limitation

Avec cette technique, on doit redéfinir les attributs à chaque fois qu'on crée une nouvelle variable de structure identique

Classe : notion de prototype – exemple 2

Création de Classe : notion de prototype – technique 1

➤ *Classe : création d'une variable de type structure qui sert de prototype*

Le prototype est l'équivalent de la classe.

Toutefois, c'est en réalité une variable de type structure : c'est donc un type « [object](#) »

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Object

Cette variable est appelée « prototype » : elle servira de référence pour la création des objets.

```
var Personnage = { // variable qui sert de Prototype
  nom: "", // valeur par défaut
  sante: 0, // valeur par défaut
  force: 0, // valeur par défaut
  xp: 0, // expérience, valeur par défaut

  // On peut mettre des fonction
  decrire: function () { // retourne la description à afficher
    var description = this.nom + " a " + this.sante +
      "points de vie, " + this.force +
      " en force et " + this.xp + " points d'expérience";
    return description;
  }
};
```

A noter qu'on écrit la variable avec une majuscule : Personnage. C'est pour dire que c'est le prototype, l'équivalent de la Classe.

➤ *Objet : création d'une variable à partir d'un prototype : fonction static [Object.create](#)*

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Object/create

On utilise la fonction static Object.create pour créer un nouvel objet à partir du prototype.

Ensuite, on peut donner des valeurs à chaque attribut.

```
var persol = Object.create(Personnage);
persol.nom = "Aurora";
persol.sante = 150;
persol.force = 25;
```

```
var perso2 = Object.create(Personnage);
perso2.nom = "Glacius";
perso2.sante = 130;
perso2.force = 35;
```

Création de Classe : fonction d'initialisation – technique 2

➤ *Ajout de méthodes dans le prototype*

On définit une fonction d'initialisation dans la structure. Cette fonction va initialiser les attributs de la structure.

De ce fait, il n'est plus nécessaire de les déclarer dans le prototype.

On peut aussi ajouter d'autres fonctions.

Notez le mot clé « this ».

```
var Personnage = {
  init: function (nom, sante, force) {
    this.nom = nom;
    this.sante = sante;
    this.force = force;
    this.xp = 0;
  },

  decrire: function () { // retourne la description à afficher
    var description = this.nom + " a " + this.sante +
      "points de vie, " + this.force +
      " en force et " + this.xp + " points d'expérience";
    return description;
  }
};

var persol = Object.create(Personnage);
persol.init("Aurora", 150, 25);
console.log(persol.decrire());
```

Utilisation du new

On peut aussi créer des objets avec un new. Mais il vaut mieux éviter.

<https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript/les-objets-5>

Héritage - exemple 2

Principes

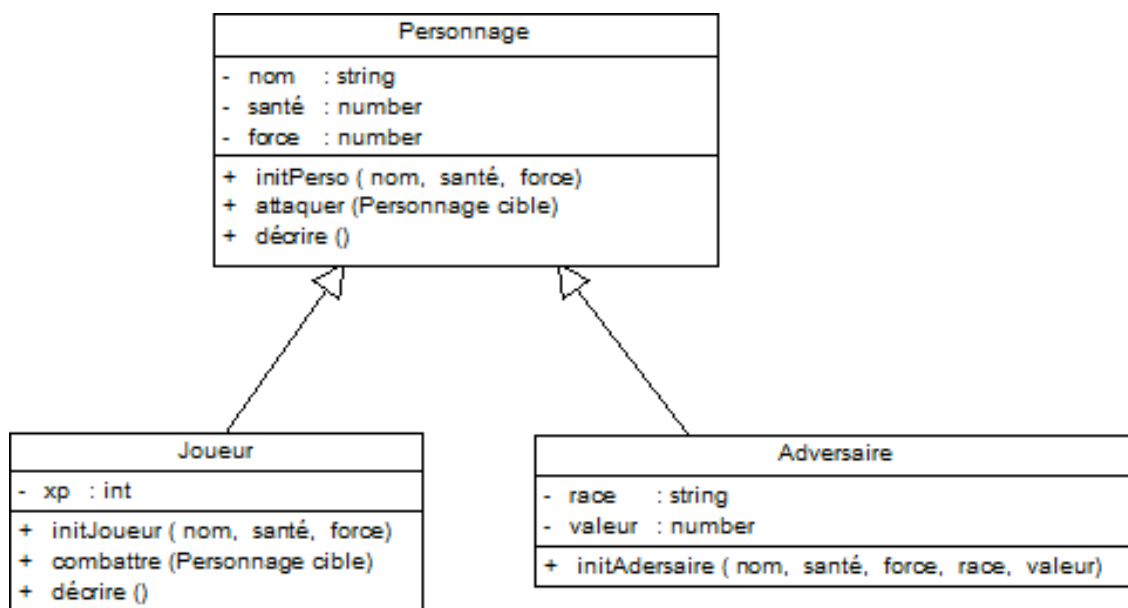
On peut simuler l'héritage avec les prototypes.

Une classe dérivée sera créée à partir d'un prototype. Ensuite, on ajoute à la variable créée des attributs et des méthodes. Elle devient un prototype pour des objets.

Application

Joueurs et Adversaires sont des Personnages. Seuls les joueurs ont de l'expérience. Les adversaires ont une race et une valeur.

Le modèle des classes est le suivant :



Bien noter que le modèle est très utile pour comprendre l'organisation des classes. Sans lui, on se perd rapidement dans le code.

➤ Code

A partir du **Personnage**, on crée un **Joueur** et on met à jour la fonction d'initialisation.

Idem pour l'adversaire.

A noter la syntaxe de la création de la fonction `initJoueur`

```
// On crée le prototype Personnage
var Personnage = {
  initPerso: function (nom, sante, force) {
    this.nom = nom;
    this.sante = sante;
    this.force = force;
  }
};

//on crée le prototype Joueur
var Joueur = Object.create(Personnage); // Prototype du Joueur
```

```

// on ajoute la fonction initJoueur au prototype
Joueur.initJoueur = function (nom, sante, force) {
    this.initPerso(nom, sante, force); // appelle initPerso
    this.xp = 0;                       // init la partie joueur
};

// on ajoute la fonction decrire au prototype
Joueur.decrire = function () {
    var description = this.nom + " a " + this.sante + " points de
vie, " + this.force + " en force et " + this.xp + " points
d'expérience";
    return description;
};

//on crée le prototype Adversaire
var Adversaire = Object.create(Personnage);

// on ajoute la fonction initAdversaire au prototype
Adversaire.initAdversaire = function (nom, sante, force, race,
valeur) {
    this.initPerso(nom, sante, force);
    this.race = race;
    this.valeur = valeur;
};

```

<http://www.pompage.net/traduction/classe-et-heritage-en-javascript>

Exercice 1 : une IHM pour l'exemple 2

L'exemple 2 travaille uniquement en console.

Produisez une IHM qui permette d'afficher les personnages et d'exécuter des méthodes.

Tableau d'objets

Exemple 3

On peut ranger les objets dans un tableau.

```
var Film = { // Prototype-Classe Film
  init: function (titre, annee) {
    this.titre = titre;
    this.annee = annee;
  },
  // Renvoie la description du film
  decrire: function () {
    var description = this.titre + " (" + this.annee + ")";
    return description;
  }
};

// création d'un tableau de films
var films = [];

// creation d'un film et rangement dans le tableau
var film = Object.create(Film);
film.init("Ta'ang", 2016);
films.push(film);

// creation d'un film et rangement dans le tableau
film = Object.create(Film);
film.init("Divines", 2016);
films.push(film);

// creation d'un film et rangement dans le tableau
film = Object.create(Film);
film.init("Juste la fin du monde", 2016);
films.push(film);

// affichage des films du tableau
films.forEach(function (film) {
  console.log(film.decrire());
});
```

Exercice 2 : une IHM pour l'exemple 3

L'exemple 3 travaille uniquement en console.

Produisez une IHM qui permette d'afficher les personnages et d'exécuter des méthodes.

Exercice 3 : tableau d'élèves avec notes et photos en POO

A partir de l'exercice 3 de l'étape 2 : 03-exercice-creer-eleve-correction (tableau d'élève avec photo), faites une version en POO :

- Coder l'étape 3 en programmation objet :
 - ✓ On se dote d'un prototype Classe pour le tableau complet et d'un prototype Eleve pour les élèves.
 - ✓ On se dotera aussi d'une fonction qui permette d'ajouter un nouvel élève.