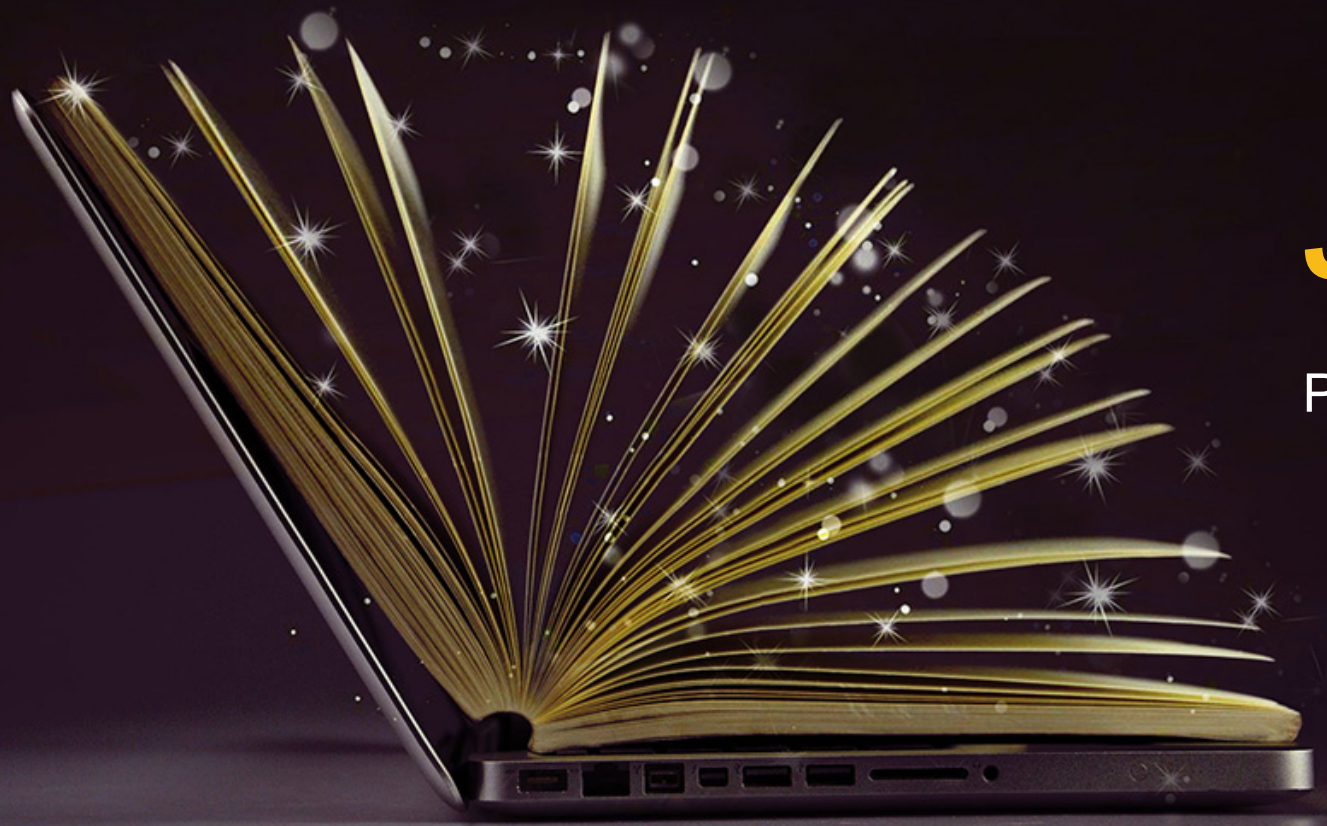


Javascript

Présentation du langage

Un peu d’histoire...

JavaScript existe depuis 1995, c’est un langage de programmation de scripts. Il est orienté objet (la base du langage est fourni avec des classes pour ses principales interfaces) qui est standardisé. On retiendra l’ECMAScript qui définit de nouvelles normes, de nouvelles fonctionnalités pour ce langage.

Ce langage est considéré comme l’une des technologies au cœur du WEB. Il permet de rendre interactives les pages d’un site internet.

Au delà du fait qu’il rende les pages d’un site interactives, il est aussi utilisé coté « serveur » avec « node.js ». C’est donc un langage extrêmement flexible, du fait qu’il est utilisé autant en front-end qu’en back-end.

La norme ES6

Cela fait désormais plus de 25 années qui séparent la première version de JavaScript de celle d'aujourd'hui. Et entre temps, beaucoup de versions comme l'ECMAScript 6 (ou ES2015 à cause de sa sortie en 2015) sont venues améliorer le langage.

Cette nouvelle version apporte plusieurs nouvelles méthodes mais aussi de nouvelles syntaxes comme par exemple « let », de nouvelles concaténations pour les chaînes de caractères, des paramètres par défaut, etc...

Insertion dans une page HTML

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>titre de la page</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
     integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
    <script>
      //un autre script ici
    </script>
  </head>
  <body>

    <script>
      //code JavaScript executable
    </script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
        integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
  </body>
```

Javascript peut s'insérer de trois manières différentes dans une page HTML.

En déclarant une balise script en haut de la page qui fait lien vers un script externe grâce à l'attribut « src ».

De la même façon mais avant la fermeture du body en bas de page.

Ou en déclarant deux balises « script » (une ouvrante et une fermante) n'importe ou dans la page HTML et en écrivant du JavaScript dedans.

Les variables

En utilisant le mot clef « var » ou « let » ou « const », il est possible de déclarer une variable.

```
var elem = 'hello world'
const element = 'lorem';
```

Il est possible de nommer la variable sans caractères spéciaux sauf avec « _ ».

Le « ; » n'est plus obligatoire en Javascript mais il peut être encore utilisé surtout si vous travaillez sur d'anciens navigateurs.

Les différents types de données

Il est possible de stocker différents types dans une variable.

Par exemple une chaîne de caractères:

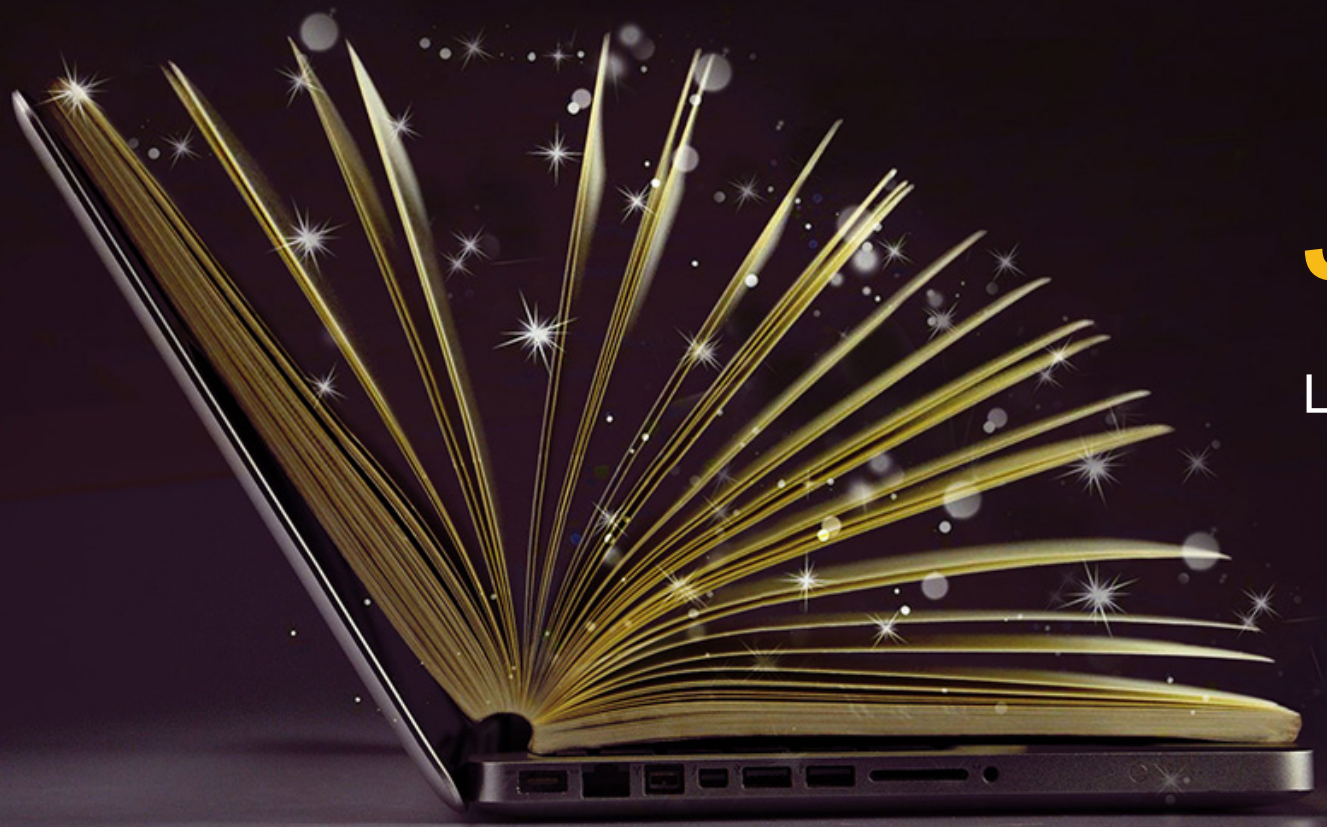
```
y = "Chaîne de caractères"
y = 'C\est évident'
```

Un nombre:

```
let t = 45
let u = 25.2654
let w = -568
let p = 1/4
```

Un boolean:

```
let valid = true
let wrong = false
```



Javascript

L’objet String

La concaténation

```
var grandeChaineDeCaracteres = "String ou chaîne de caractères en français, " +
   "permet de stocker du texte " + "dans une variable.";

console.log(grandeChaineDeCaracteres);
```

```
var a = "a";
var b = "b";
if (a < b) { // true
   console.log(a + " est inférieure à " + b);
} else if (a > b) {
   console.log(a + " est supérieure à " + b);
} else {
   console.log(a + " et " + b + " sont égales.");
}
```

```
var s_prim = "hello";
var s_obj = new String(s_prim);

console.log(typeof s_prim); // affiche "string"
console.log(typeof s_obj);  // affiche "object"
```

Permet de sauvegarder une chaîne de caractères dans une variable et ainsi de la manipuler dans le code avec plus de facilité.

Dans le premier exemple, le symbole « + » permet de concaténer les différentes chaînes de caractères dans une seule et même variable.

Dans le deuxième exemple, grâce aux opérateurs <, > ou =, il est possible de comparer des chaînes de caractères.

Dans le dernier exemple, il est possible d'utiliser l'objet « string » en créant une nouvelle instance de celui-ci sur une variable qui a une chaîne de caractères.

La recherche de caractère

Si votre variable est une chaîne de caractères,

il est possible de faire une recherche dedans avec la fonction « search » (en bleu sur l'exemple).

Cette fonction renvoi l'emplacement du caractère recherché.

```
var letter = 'L'
let search = lorem_255.search(letter) //return 0
console.log(search)
console.log(lorem_255[0])
```

Suppression d’un caractère

```
let txt = lorem_255.replace('Lorem','coucou')
console.log(txt)
```

Sur une chaîne de caractères, il est aussi possible de remplacer une partie d’une chaîne de caractères par une autre grâce à la fonction « replace() ». Elle prends deux paramètres, le premier c’est la partie à changer sur la chaîne à changer et le deuxième c’est la chaîne qui va modifier cette partie.

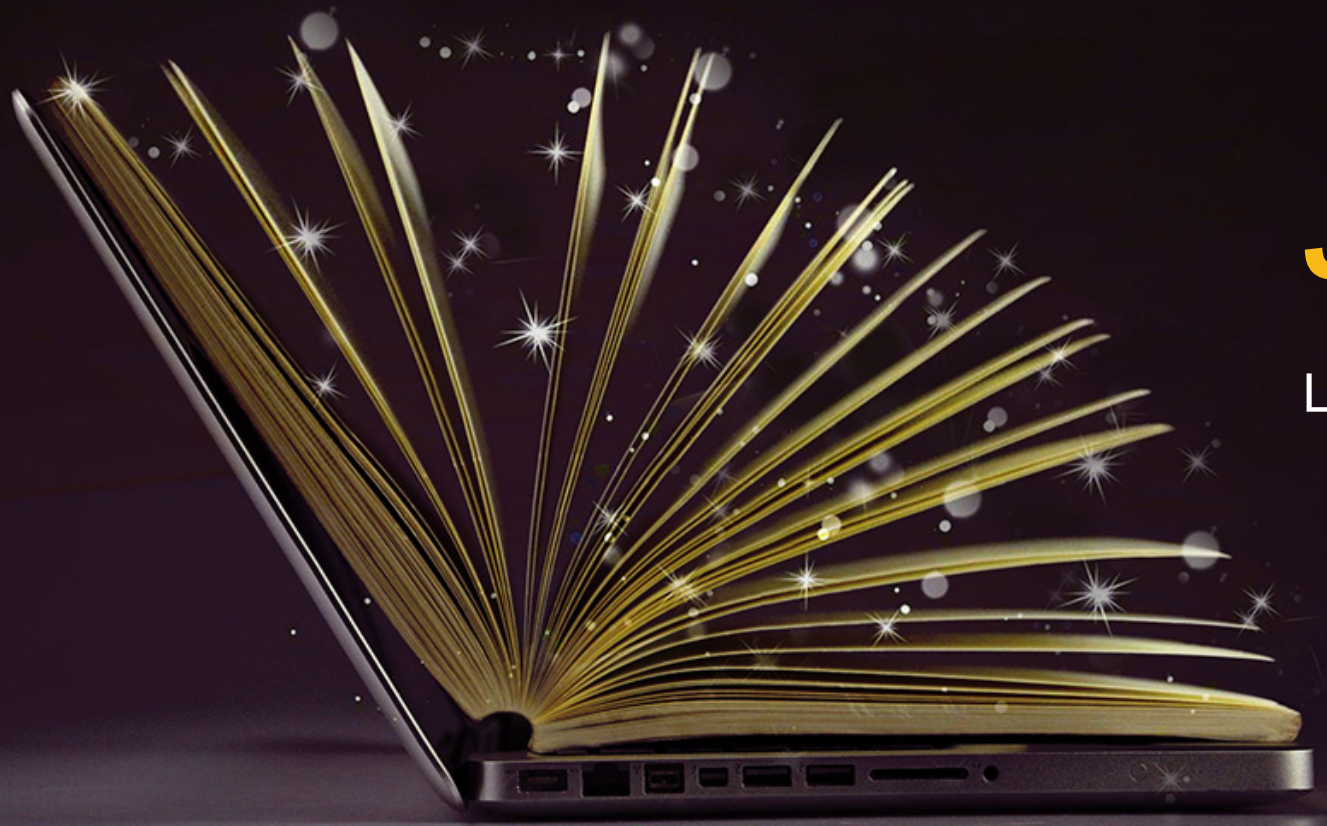
Pour faire une suppression, il suffit de mettre le deuxième paramètre sur du vide pour ainsi effacer la partie voulue.

Une autre fonction existe, elle soustrait une partie de la chaîne de caractères, c’est « substr() ». Attention, elle risque de devenir obsolète.

Formatage HTML d’une chaîne

Il est possible de formater du html (tant que vous respectez la structure d’un document HTML) dans une chaîne de caractères pour ensuite l’afficher dans un élément du document HTML.

```
var html = '<div class="container">' +
   '<div class="row">' +
   '<div class="col-12">' +
   '<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.' +
   ' Accusantium commodi deserunt dolores ea earum eum exercitationem.</p>' +
   '</div>' +
   '</div>' +
   '</div>'
var content = document.getElementById('content')
content.innerHTML = html
```



Javascript

Les tableaux

Créer un tableau

```
var sports = ['basket', 'volley', 'foot',
'natation', 'tennis', 'course'];

console.log(sports);
console.log(sports[0]);

var random = ['arbre', 795, [0, 1, 2]];

console.log(random.length);
console.log(random[2].length);
```

Les tableaux permettent de stocker différentes données (des chiffres, des chaînes de caractères, des variables) de façon structurer et ordonner.

Un tableau se déclare avec les « [ ] » pour délimiter ce qui se trouve dedans.

Dans l'exemple, la variable « random » contient deux tableaux (c'est un tableau a plusieurs dimensions). Le premier tableau contient l'ensemble tandis que le deuxième tableau se situe à la deuxième position du premier.

Gestion des index

Quand l'on déclare « nom_du_tableau[0] », on précise que l'on veut le premier élément du tableau car l'index d'un tableau commence à 0.

On peut aussi déclarer plusieurs index si un tableau contient plusieurs éléments ou plusieurs tableaux (plusieurs dimensions), comme par exemple « nom_du_tableau[0][1] ».

```
var random = ['arbre', 795, [0, 1, 2]];

console.log(random);
console.log(random[2][0]);
```

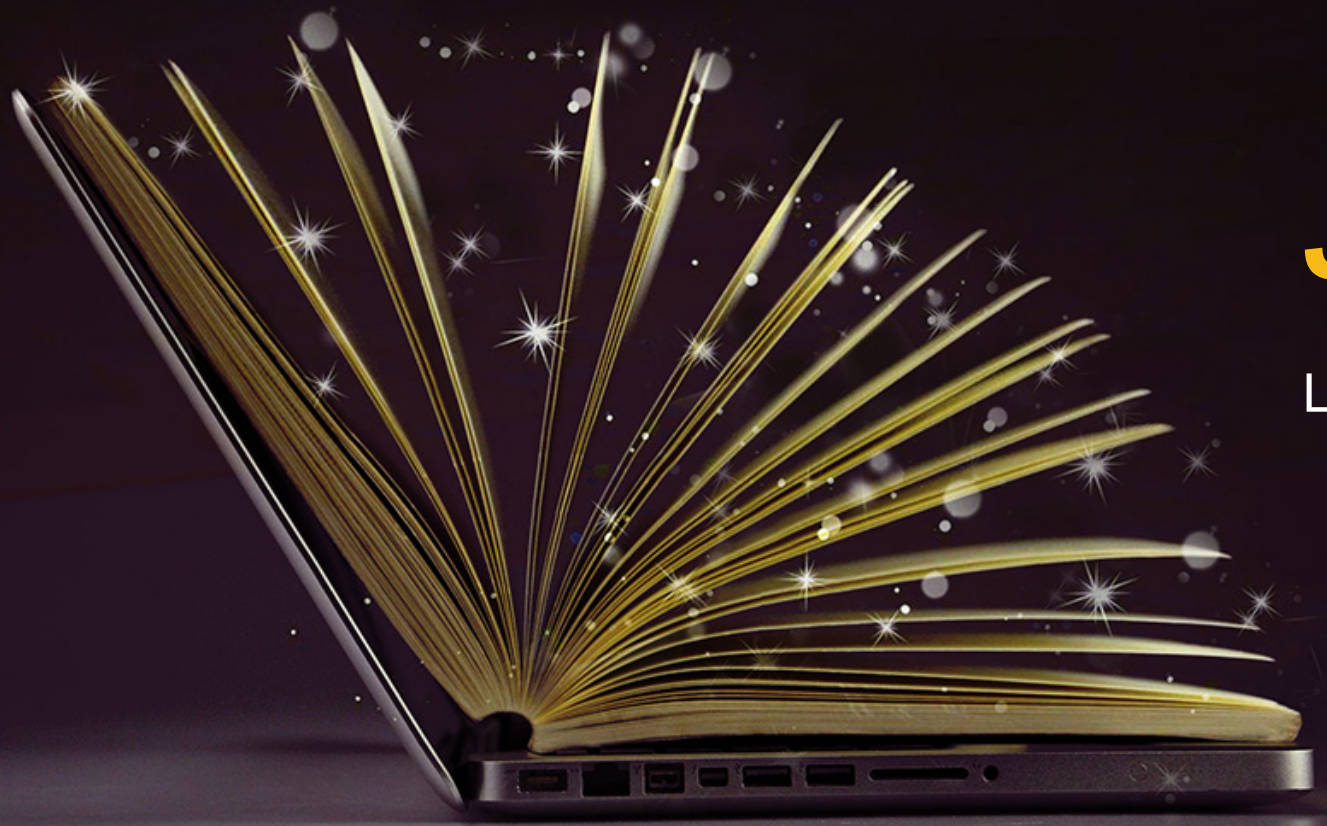
Manipulation d’un tableau

```
let array = ['0','1','2','3']
array[1] = '4';
array.push('5')
let removeLast = array.pop()
let removeFirst = array.shift()
console.log(array)
console.log(removeLast)
console.log(removeFirst)
```

Un tableau est manipulable grâce aux index avec par exemple « array[1] » pour récupérer ou modifier cette partie du tableau.

La fonction « push() » prend un élément en paramètres pour l’ajouter à la fin du tableau.

Quand a « pop() » et « shift », ces deux fonctions enlève un élément dans le tableau, respectivement le dernier et le premier.



Javascript

Les structures logiques et les boucles

Les opérateurs logiques

```
if(valid === true || lorem_255 !== "" && array.length > 1 ){
   //execute code si les conditions sont remplies
}
```

Dans cette condition (ici un if), si « valid » est égal à « true » ou « lorem_255 » est différent de vide et la taille de « array » supérieur à 1, le code rentre dans la condition.

Voici les différents opérateurs logiques et leurs significations:

« || » pour signifier « ou »

« && » pour signifier « et »

« ! » pour signifier « différent de »

Les boucles

Il existe trois types de boucles:

La boucle « for » s'éxecutera tant que la variable « i » sera inférieure ou égale à 5. A chaque itération, la variable « i » s'incrémente de 1.

```
for(let i =0; i <= 5; i++){
   //execute code sur chaque itération
}
```

La boucle « forEach » permet de boucler sur un tableau. La longueur du tableau sert de limite pour boucler.

```
array.forEach((data, index) => {
   //execute code on element
})
```

La boucle « while » tournera tant que la variable « boucle » sera inférieure à 5.

```
let boucle = 0
while (boucle < 5){
   boucle++
   console.log(boucle)
}
```

Les structures conditionnelles

```
if(condition à réaliser){
   //code qui s'execute si la condition est bonne
}else{
   //code qui s'execute si la condition n'est pas bonne
}
```

```
if(condition à réaliser){
   //code qui s'execute si la condition est bonne
}else if(){
   //code qui s'execute si cette deuxième condition est bonne
}else{
   //code qui s'execute si les conditions ne sont pas bonnes
}
```

Le « if » sert de condition. Si cette condition est respectée, la première partie du code s'exécute sinon c'est l'autre partie qui s'exécute.

Le « else if » rajoute une nouvelle condition.

Il existe aussi les conditions ternaire comme celle-ci:

```
(condition) ? instruction si vrai : instruction si faux
```

Les structures conditionnelles

En plus du « if », si vous avez des conditions à mettre en place le « switch » peut être une bonne alternative.

Il faut faire attention de ne pas avoir trop de conditions sinon le « switch » perds de son utilité !

On rentre une variable dans le « switch » puis on ajoute un « case » qui compare avec la variable. Si le cas se vérifie, le code s'exécute sinon il passe au cas suivant. Sinon la variable ne passe dans aucuns cas, c'est celui par défaut qui est pris en compte.

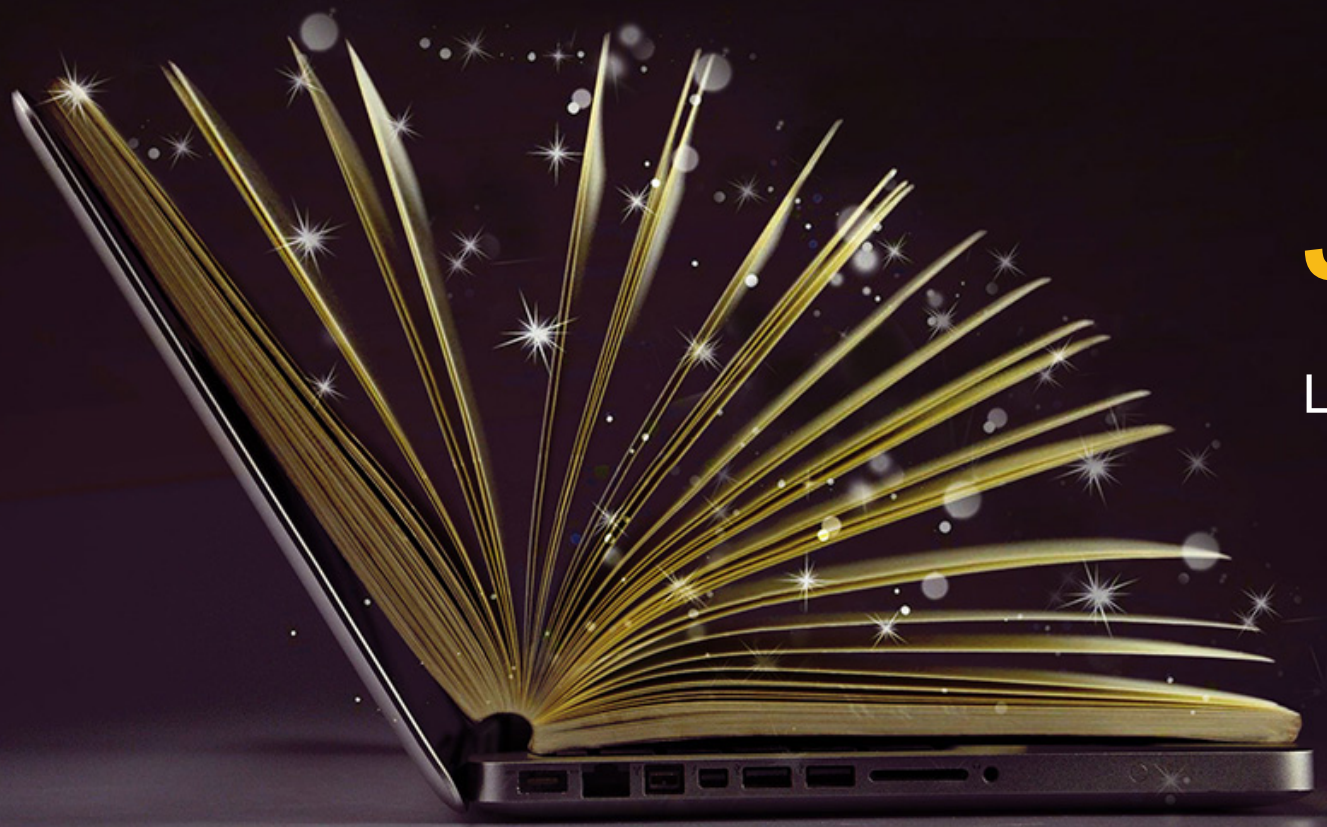
```
switch (word) {
 case "attente":
   code à executer
   break;
 case "debut":
   code à executer
   break;
 default:
 code à executer par défaut
}
```

Fonctions

```
function nom_de_la_fonction(paramétres){
    //code qui s'execute
}
```

```
nom_de_la_fonction('hello')
```

Une fonction c'est un morceau de code que l'on peut exécuter ou l'on veut dans son script (tant que la fonction est déclarée) en l'appelant par son nom et si besoin en passant des paramètres entre les parenthèses.



Javascript

Les Objets

Présentation des objets

```
const objet = {
    empty: "",

    array: ["1", "2", "3", "4", "5"],

    multiplicate(a,b){
        return a*b
    },

    test(elem){
        //function code
        this.multiplicate(5,5)
    }
}

objet.multiplicate(4,4)
```

A la différence d'un tableau, l'objet littéral se crée entre « { » et « } ». En créant un objet, il est possible d'y ranger des variables, des tableaux ou des fonctions.

Ensuite, comme une fonction, il est possible d'appeler tout ce qui se trouve dans l'objet en question.

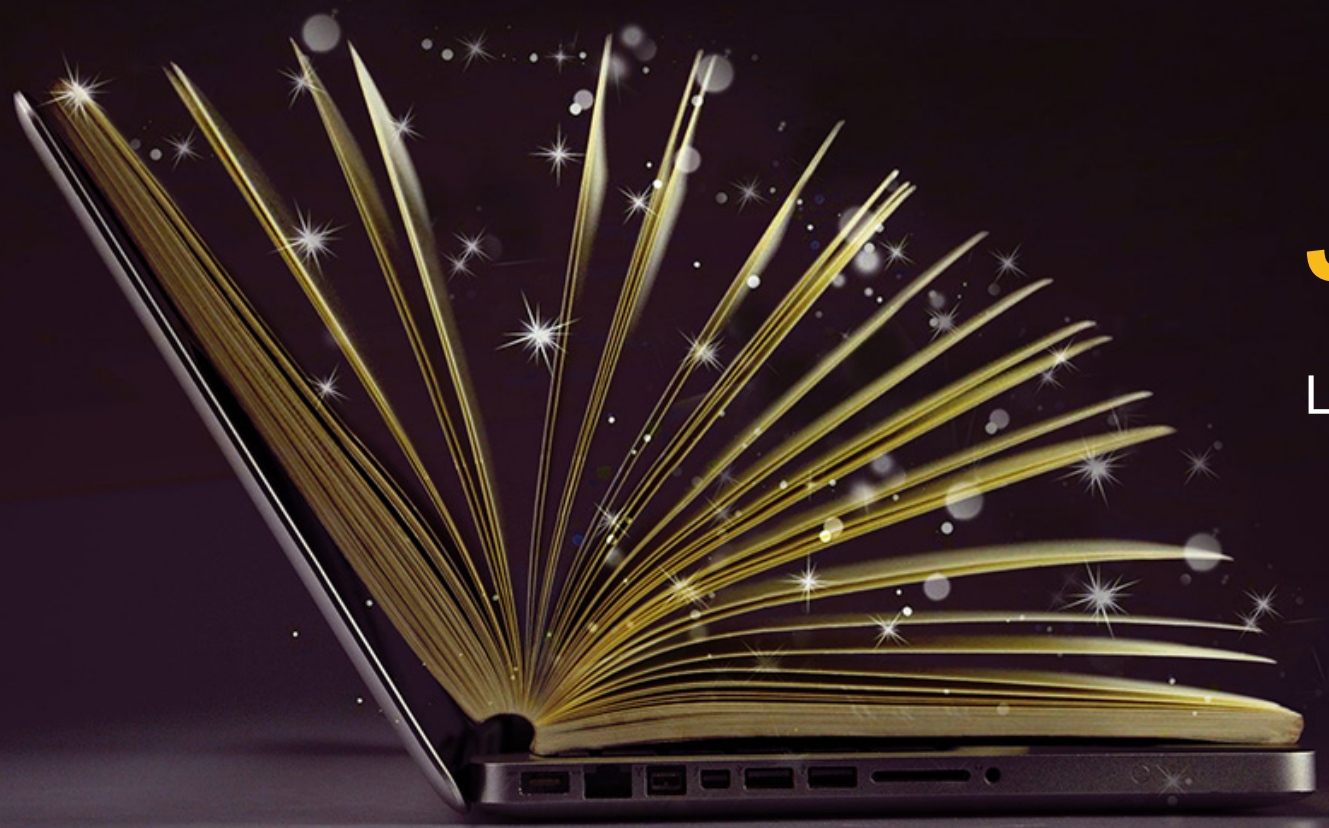
Le JSON

Le JSON ou JavaScript Object Notation permet de sérialiser (compresser des données et ainsi faciliter le transport de celles-ci) de grande quantité de données.

Le JSON est basé sur du JavaScript mais en est distinct.

Dans cet exemple, des données en JSON d'une réponse venant d'une API.


```
{
  "coord": {"lon": -122.08,"lat": 37.39},
  "weather": [
  {
    "id": 800,
    "main": "Clear",
    "description": "clear sky",
    "icon": "01d"
  }
],
  "base": "stations",
  "main": {
  "temp": 282.55,
    "feels_like": 281.86,
    "temp_min": 280.37,
    "temp_max": 284.26,
    "pressure": 1023,
    "humidity": 100
},
  "visibility": 16093,
  "wind": {
  "speed": 1.5,
    "deg": 350
},
  "clouds": {
```



Javascript

L’objet Math

```
console.log(Math.random(max));
```

Générer un chiffre aléatoirement

Math est un objet qui permet d'utiliser différentes fonctions mathématique mais aussi de manipuler les chiffres pour par exemple arrondir au supérieur quand il est décimal. Renvoyer un nombre aléatoire compris entre 0 et 1. Pour le bon fonctionnement de l'objet, il est impératif de passer un nombre.

Ici, « Math.random() » permet de sélectionner un chiffre à virgule aléatoirement entre 0 et 1.

Arrondir les valeurs

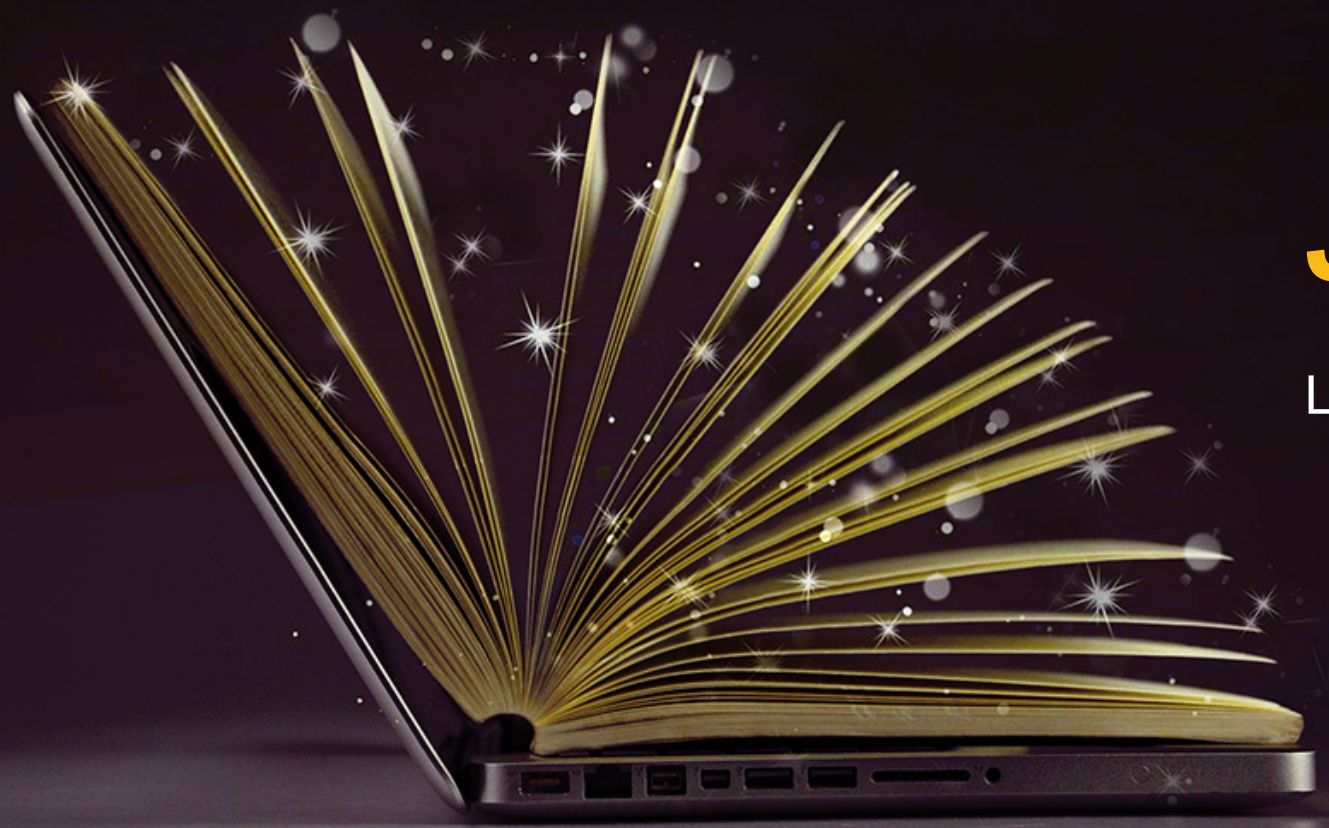
Dans l'exemple, l'on crée une fonction « getRandomInt » qui a comme paramétre « max », un chiffre envoyé à la fonction.

Ensuite elle retourne un entier inférieur ou égal à un nombre (ici, « Math.floor ») qui provient de la multiplication d'un nombre décimal entre 0 et 1 (avec « Math.random ») et un entier inférieur ou égal à la valeur de la variable « max ».

```
function getRandomInt(max) {
  return
Math.floor(Math.random() *
Math.floor(max));
}

console.log(getRandomInt(3));
// sortie attendue: 0, 1 or 2

console.log(getRandomInt(1));
// sortie attendue : 0
```



Javascript

L’objet Date

Déterminer la date

```
var date1 = new
Date('December 17, 1995
03:24:00');
// Sun Dec 17 1995 03:24:00
GMT...

var date2 = new Date('1995-
12-17T03:24:00');
// Sun Dec 17 1995 03:24:00
GMT...

console.log(date1 === date2);
// sortie attendue: false;

console.log(date1 - date2);
```

- Pour manipuler les dates en JavaScript, il existe un objet dédié: « Date ». Il est possible de créer une nouvelle date en rajoutant le mot « new » devant l'objet, ainsi l'on crée une nouvelle instance de l'objet.
- Le format de la date est en GMT (ou UTC), l'échelle de temps internationale. Plusieurs fonctions liées avec cet objet permettent de transformer la date en chaîne de caractères, de changer la locale pour qu'elle soit lisible dans une autre langue.

Opération sur les dates

- Il est aussi possible de paramétrer une date grâce aux paramètres liés à l'objet « date ». Cela permet de définir la date dans sa globalité, un mois, une année, un jour, une heure, etc…

```
var dateLocale =
date1.toLocaleString('fr-FR',{
   weekday: 'long',
   year: 'numeric',
   month: 'long',
   day: 'numeric',
   hour: 'numeric',
   minute: 'numeric',
   second: 'numeric'});

console.log(dateLocale);
```

```
//On crée une date
let date1 = new
Date(2019, 0, 25, 12,
30, 15);

//On modifie la date
date1.setDate(31);
date1.setMonth(2);
date1.setFullYear(2018
);
date1.setHours(10);
date1.getUTCHours();
date1.setMinutes(0);
date1.setSeconds(0);
date1.setMilliseconds(
0);
```

Formatage des dates

- Dans cet exemple, vous pouvez voir les différentes fonctions pour paramétrer une date.
- Il est possible de modifier le jour grâce à setDate, le mois grâce à setMonth et ainsi de suite...
- Pour les fonctions commençant par « get », il est juste possible de récupérer la valeur en question.