

# Javascript cours n°1

A l'issue de ce cours vous aurez révisé les bases de Javascript :

- variables : déclaration, types, portée
- conditionnelles
- boucles
- fonctions
- tableaux

# Historique, normes...

- Créé en mai 1995, par Brendan Eich (Netscape)
- But : ajouter de l'interactivité à html dans Netscape Navigator
- Mocha, LiveScript, JavaScript
- « Java... » pour des raisons marketing principalement



# Historique, normes...

- Au départ côté client, dans un navigateur
- Recherche Altavista : « comment désactiver Javascript ? »
- Aujourd'hui :
  - Nombreuses librairies et frameworks (Jquery, Angular, Vue, React, Svelte...)
  - Mobile hybride : Cordova, Ionic, etc.
  - Desktop multi-plateformes (Electron)
  - JS côté serveur (Node.js, Express)
    - JS « vrai » langage avec lequel on développe des applications

# Historique, normes...

- Un langage, *n* navigateurs
- Standardisation : ECMAScript
- ES3 1999
- ES5 2009
- **ES6** 2015 = ES2015
  - let, const, arrow functions, classes, ...
- ES7 en 2016 : Array.includes(...), \*\*
- Une nouvelle version par an, ES10=ES2019
- ES12 en 2021...

(Voir par exemple

<https://www.freecodecamp.org/news/ecmascript-2016-es7-features-86903c5cab70/>)

# Historique, normes...

## Browser Support for ES6 (ECMAScript 2015)




Safari 10 and Edge 14 were the first browsers to fully support ES6:

				
Chrome 58	Edge 14	Firefox 54	Safari 10	Opera 55
Jan 2017	Aug 2016	Mar 2017	Jul 2016	Aug 2018

- `replaceAll` (ES2021) <https://developer.mozilla.org>

## Browser compatibility

Update compatibility data on GitHub

													
	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet	Node.js
<code>replaceAll</code>	85	85	77	No	71	13.1	85	85	No	No	13.4	No	No

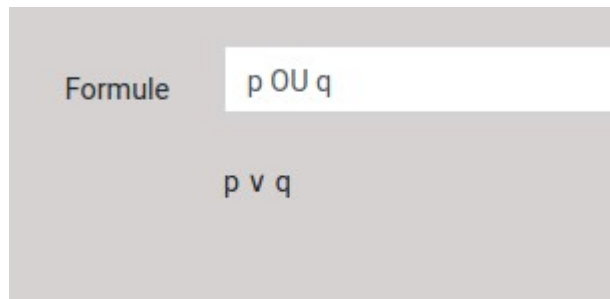
(09/2020)  
09/2021 :  
ok sauf IE

# Historique, normes...

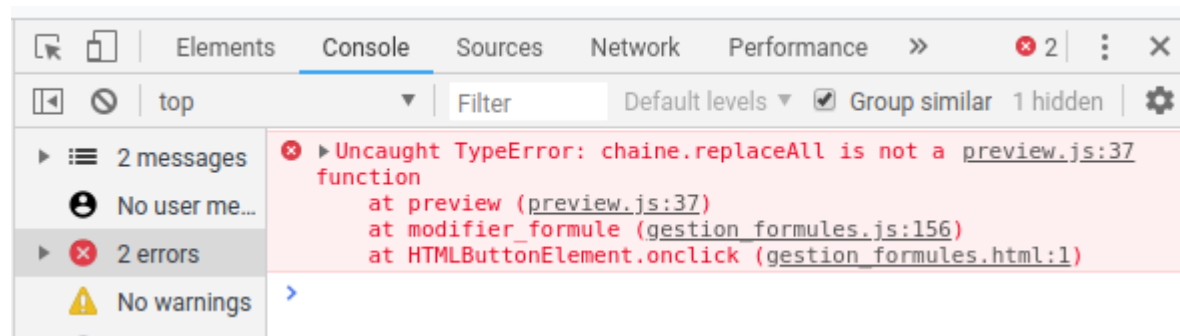
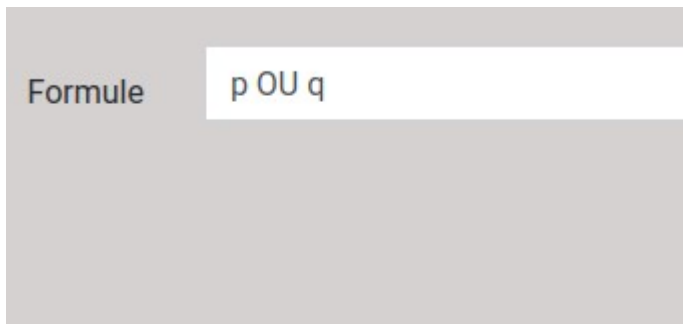
- Trois remarques :
  - Les navigateurs anticipent les normes
  - Ce qui fonctionne sous un navigateur version *desktop* peut ne pas fonctionner sur ce même navigateur version mobile
  - Toujours travailler avec la console des outils de développement du navigateur, afin de voir les erreurs

# Normes et *debug*

- Utilisation de `replaceAll` :
- Chrome 85, Firefox 80 (09/2020, linux)



- Opera 55 (09/2018, linux)



# Types, variables

- Déclaration de variables :
- Avant ES6, ou ce qu'il ne faut plus faire...
  - `declaree`, portée sur tout le fichier JS
  - `pasDeclaree`, portée sur tout le fichier JS
  - `declaree` : portée sur la fonction

```
var declaree = 1;

function declarations() {
    pasDeclaree = 1;
    var declaree = 2;
}

declarations();
console.log(pasDeclaree); // 1
console.log(declaree); // 1
```



# Types, variables

- Déclaration de variables :
- Depuis ES6, la portée des variables est réduite au maximum (« bloc »)  
→ `var2` et `pi` inconnues hors du bloc `if`

```
function declarations() {  
    let var1;  
    if (var1===undefined) {  
        let var2 = 12;  
        const pi = 3.14159;  
    }  
    console.log(var1, var2, pi);  
}
```

```
! Uncaught ReferenceError: var2 is not defined tdl_2.js:14:5  
    declarations ...avascript/tdl_2.js:14  
    <anonymous> ...avascript/tdl_2.js:17
```

- `let` (variable)
- `const` (constante)
- `var` est toujours possible, mais votre IDE affiche un *warning*

# Types, variables

- Différents types en JS
  - number : entiers et réels entre  $-(2^{53} - 1)$  et  $2^{53} - 1$
  - string : au plus  $2^{53} - 1$  caractères
  - boolean : true, false
  - bigint (ES2020, 9999...99n)
  - object
  - function
  - ((symbol))
  - undefined : variable non initialisée
  - null

# Types, variables

- Permissivité :
  - Une variable peut être réaffectée, et changer de type

```
let var2 = 12;  
var2 = 'toto';
```

- Egalité simple VS égalité stricte (égalité type & valeur)

```
let var2 = 25;  
let var3 = "25";  
if (var2==var3) {  
    console.log("égalité");  
}  
if (var2===var3) {  
    console.log("égalité stricte");  
}
```

# Types, variables

- `==` : conversion puis test d'égalité
- Conséquence
  - `'123' == 123 == 123.000`
  - booléen `true` `== '1' == 1 == 1.000`
  - booléen `false` `== '0' == 0 == 0.0000`
  - `null == undefined` (et à rien d'autre)
- `===` est à privilégier (*warning* IDE sinon)

# Types, variables

- Classe enveloppe (wrapper) pour le type primitif **number**

```
Number.isInteger(...)
```

```
Number.parseInt(...)
```

```
Number.parseFloat(...)
```

```
Number.MAX_SAFE_INTEGER
```

```
Number.MIN_SAFE_INTEGER
```

- **Remarque** : `parseInt` et `parseFloat` existent aussi sur l'objet global

# Parenthèse : l'objet Math

- Objet global permettant de faire des opérations mathématiques de base

```
Math.abs(-5)
```

```
Math.round(5.2)
```

```
Math.pow(2, 10)
```

```
Math.min(2, 5, 1, ...)
```

```
Math.max(10, 8, ...)
```

# Types, variables

- Les chaînes
  - Simples, doubles ou *back* quotes
  - *Backquotes* pour accéder à des variables

```
let titre = 'Monsieur';  
let entete = `bonjour ${titre}`;
```

- Immuables ! Ne peuvent être modifiées
- Propriété applicable : `s.length`
- **Fonctions** applicables :
  - `s.charAt(2)`
  - `s.substring(3)`
  - `s.substring(3, 5)`
  - `s.indexOf('sch')`
  - `s.lastIndexOf('sch')`

# Types, variables

- Les chaînes

- **Fonctions** applicables (suite) :

```
s.includes('sch')  
s.startsWith('sch')  
s.endsWith('sch')  
s.toLowerCase()  
s.toUpperCase()  
s.trim()  
s.replace('old', 'new')  
s.replace(regExp, 'new')  
s1.concat(s2) (ou s1+s2)  
...
```



# Types, variables

- Les tableaux sont des objets
- Déclaration :

```
let o0=new Array();  
let o1=Array();  
let o2=[];  
let o3=[5, 6, 7];  
let o4=Array( items: 5,6,7);  
let o5=new Array( items: 5,6,7);
```

- Attention ambiguïté :

```
let o6=Array( arrayLength: 3);  
let o7=new Array( arrayLength: 3);
```

→ la déclaration littérale [...] est préférée

# Types, variables

- Propriété d'un tableau : `tab.length`
- Accès direct à un élément : `tab[1]`
- Fonctions (cf cours suivant...)
  - `push, unshift`
  - `pop, shift`
  - `splice`
  - `reverse`
  - `sort`
  - `forEach, map`
  - `includes, indexOf`
  - `keys, values`
  - `etc.`

# Types, variables

- Les objets
- (les tableaux sont des objets, les chaînes aussi)
- Possibilité de définir des classes
- Notation JSON (JavaScript Object Notation)

```
let unObjet = {  
  "promo" : "lpwmce",  
  "nb" : 25  
};
```

# Opérateurs

- Très similaires au langage C
  - Arithmétiques : `+`, `-`, `*`, `/`, `%`, `**`  
(la division donne un `number`, pour lequel `Number.isInteger` renvoie vrai si la division « tombe juste »)
  - Incrémentation/décrémentation  
`++x`, `--x`, `x++`, `x--`
  - (In)égalités  
`==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`

# Opérateurs

- Très similaires au langage C
  - Logiques  
    & & ,    | | ,    !
  - Affectations  
    = ,    += ,    -= ,    \*= ,    /= ,    % =

# Expressions conditionnelles

- if, switch/case, ?

```
if (i > 100) {  
    doSomething();  
} else if (i > 10) {  
    doSomethingElse();  
} else {  
    doThat();  
}
```

```
let ch="xxx";  
switch (ch) {  
    case "toto":  
        doSomething();  
        break;  
    case "titi":  
    case "tutu":  
        doSomethingElse();  
        break;  
    default:  
        doThat();  
}
```

```
let statut;  
let res = age > 18 ? statut="majeur":statut="mineur";
```

# Boucles

- for, while, do

```
for (let i=0; i < 10; i++) {  
    console.log(i);  
}
```

```
let trouve=false;  
while (!trouve) {  
    trouve=cherche();  
}
```

```
do {  
    trouve=cherche();  
} while (!trouve);
```

- break : sortir de la boucle
- continue : aller directement à l'itération suivante
- Autres formes de for vues plus tard

# Fonctions

- Trois possibilités

```
const carre = function(x) {return x*x;
```

```
}function carre2(x) {  
    return x*x;  
}
```

```
const carre3 = (x) => {return x*x;
```

- `carre2` peut être appelée avant sa déclaration
- `carre3` : *arrow function* très utilisée en react/vue/etc.



# Fonctions

- Paramètres par défaut, permissivité

```
function puissance(x, y :number =2) {  
    return x**y;  
}
```

```
console.log(puissance()); // NaN  
console.log(puissance(x: 10)); // 100  
console.log(puissance(x: 10, y: 5)); // 100000  
console.log(puissance(x: 10, y: 5,3,4,7,8)); // 100000
```

# Fonctions

- *Rest parameters*

```
function tri(...valeurs) {  
    valeurs.sort();  
    return valeurs;  
}
```

```
console.log(tri(valeurs: 1, 5, 3));  
console.log(tri(valeurs: 1, 5, 3, -12, 87, 100, 2));
```

- (tri par défaut en faisant des conversions en chaîne)

# Interactivité basique

- Afficher une boîte de dialogue
- Afficher une boîte de conformation OK, Annuler
- Afficher une boîte de saisie

```
alert("La mise à jour a bien été effectuée.");  
let reponse = confirm("Voulez-vous vraiment quitter cette application ?")  
if (reponse) {  
    console.log("parti");  
}  
  
let valeur = prompt( message: "Quel âge as-tu ?");
```

# Integration js/html

- Plusieurs possibilités, nous en reparlerons

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>test js</title>
</head>
<body>

  <script src="td1_2.js"></script>
  <script src="td1_3.js"></script>
</body>
</html>
```