



JavaScript

Adrien Hergat

 lesentrecodeurs.com

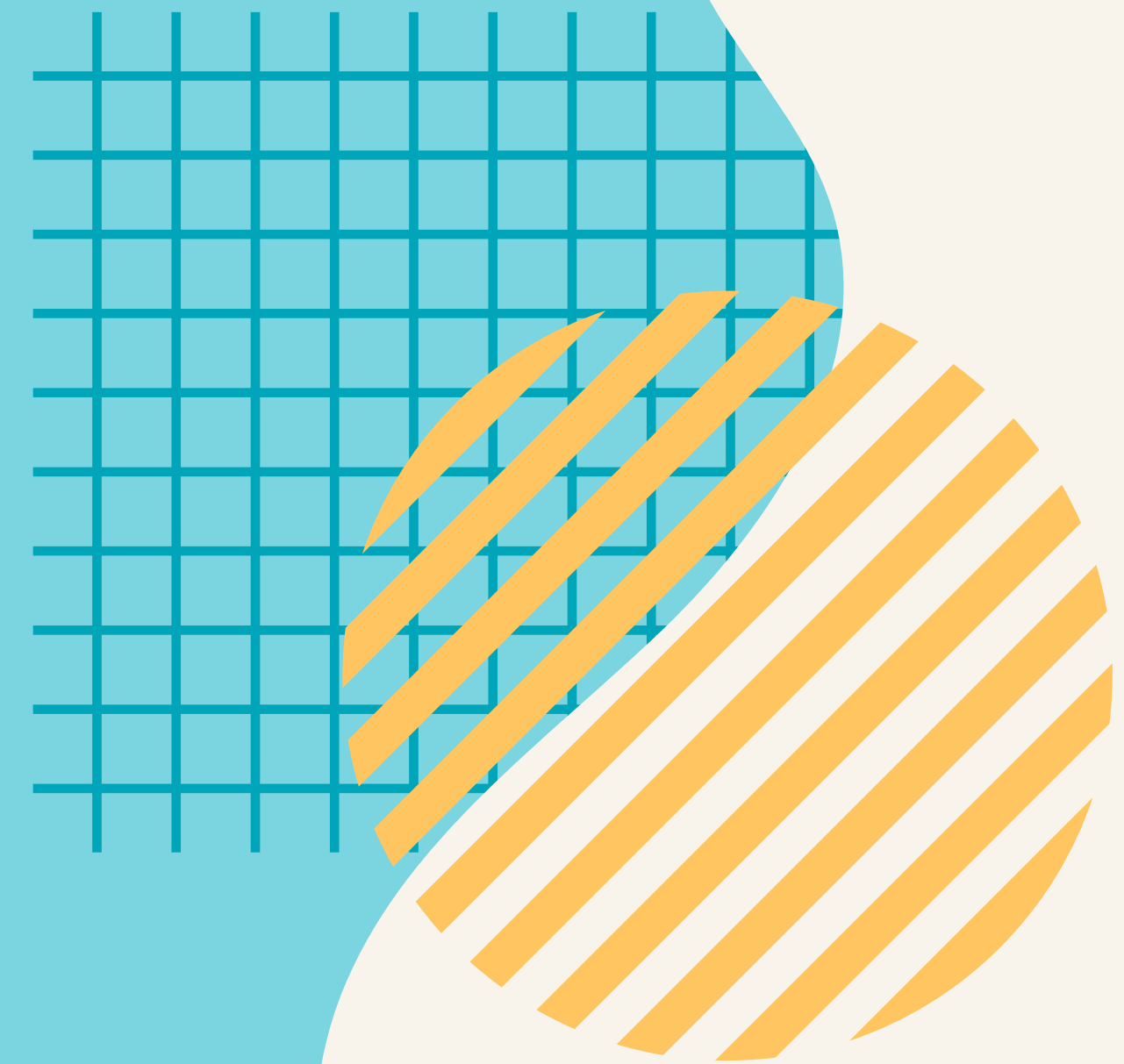
 adrien@lesentrecodeurs.com

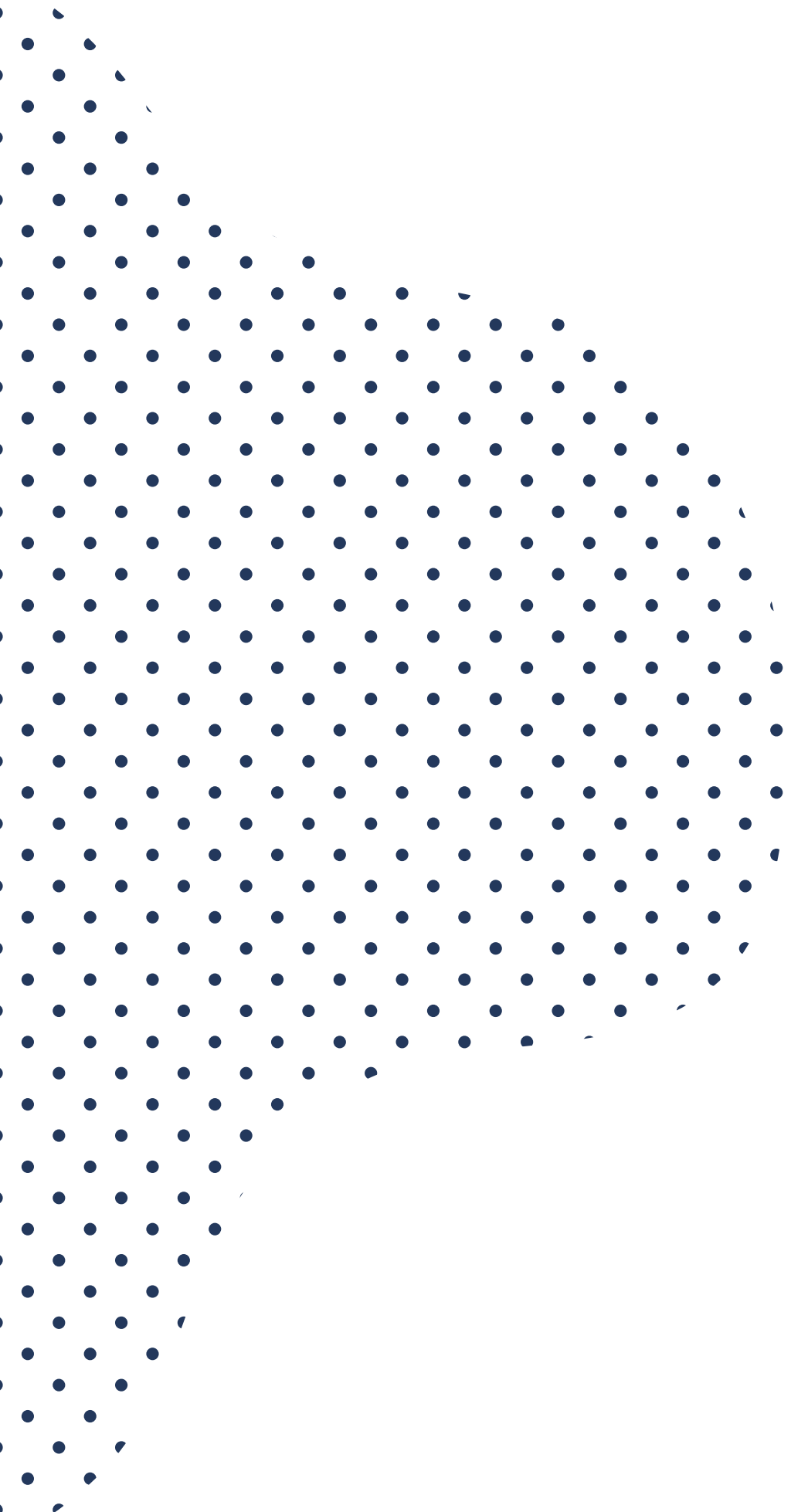
 [hermoult#5790](#)

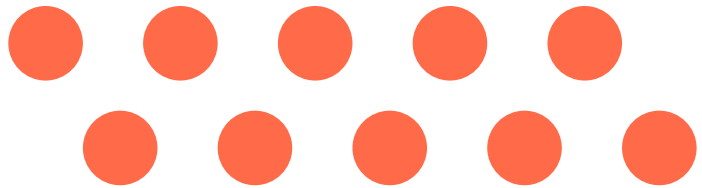


Le plan

1. Pourquoi JavaScript
2. Les variables
 - types, scope, concat, operator etc
3. Les boucles
4. Structures de controle
5. Procédures et fonctions
6. DOM et events





Always code as if the guy who
ends up maintaining your code
will be a violent psychopath who
knows where you live. 

John Woods, Software Engineer

Pourquoi JavaScript

Historiquement les technologies Web

HTML, CSS

JavaScript

Issu du navigateur Netscape (Brendan Eich, 1995), numéro 1 au milieu des années 1990. Détrôné par MicroSoft avec Internet Explorer (qui utilise JScript et non JavaScript), il revient très fort avec Firefox (2002) puis Chrome (2008).

Interprété ou compilé à la volée par le navigateur. Pour en apprendre d'avantage sur la compilation de Javascript

Spécificité Javascript

Langage le plus populaire du moment

Web moderne et dynamique

Accessible**

Très grande communauté

Très permissif

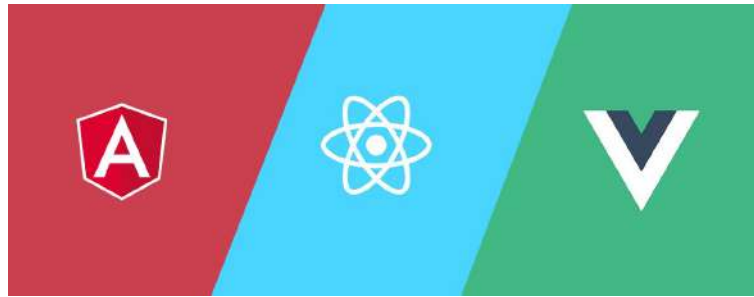
Synchrone/Asynchrone

Sa force

Modifier dynamiquement les éléments dans le DOM : Document Object Model

Le DOM

<https://survey.stackoverflow.co/2023/#technology-most-popular-technologies>



Et maintenant ?

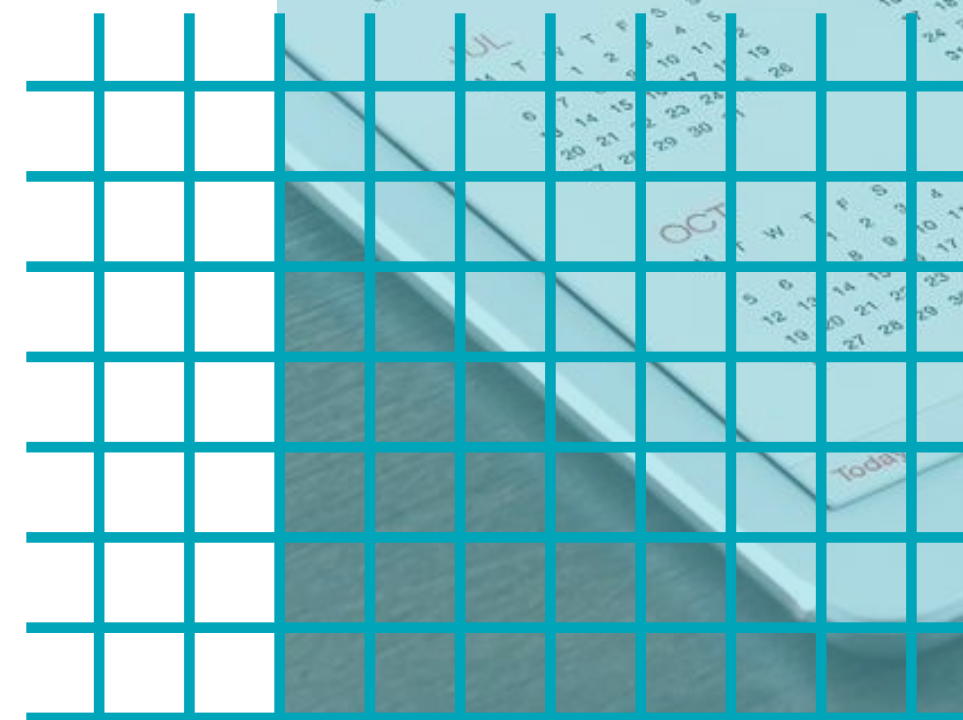
JavaScript pour développer sur bureau, mobile et tablette.

React, Angular 2+, Vue etc

Possibilité de développer des applications serveurs.

Nodejs, Deno, Express, Meteor, etc

<https://2020.stateofjs.com/en-US/>



Domaine d'utilisation

🤖 IoT

😡 Systèmes embarqués

😊 Applications Desktop

😊 Web

😊 Jeux-vidéo

😊 Mobile

😡 🤖 ML

Electron

Angular, React, Vue, Hapi, ExpressJs

Phaser, Threejs

React Native, NativeScript

TensorflowJs

Évolution du langage

jQuery

-

ES6 = ES2015**

ES7 = ES2016

ES8 = ES2017

ES9 = ES2018

ES10 = ES2019

ES11 = ES2020

ES12 = ES2021

ECMAScript

Les variables

mot clé **nom de la variable** "=", opérateur d'affectation

```
var jeSuisUneVariable = "variable globale"; // variable globale

function uneFonction() {
  var jeSuisUneVariable = 1;
  // une var peut être redéclarée
  // ici elle sera locale
  console.log(jeSuisUneVariable);
}

uneFonction();
console.log(jeSuisUneVariable);
```

Le scope est la portée de la variable

Scope de var : global ou fonction

Les variables

```
function anyFunction () {  
  const uneVariableEnLectureSeule = 32;  
  try {  
    uneVariableEnLectureSeule = 44;  
  } catch (e) {  
    console.error(e); // error  
  }  
  console.log(uneVariableEnLectureSeule); // 32  
}  
  
anyFunction();  
if (true) {  
  const autreVariableEnLectureSeule = "OUI";  
  console.log(autreVariableEnLectureSeule); // OUI  
}  
  
console.log(autreVariableEnLectureSeule); // error
```

Scope de const et let : global, fonction et bloc

Les variables

```
function anyFunction () {  
  let uneVariable = "BIM";  
  try {  
    uneVariable = "BAM";  
  } catch (e) {  
    console.error(e); // pas d'erreur, réaffectation  
  }  
  console.log(unVariable); // BAM  
}  
  
anyFunction();  
if (true) {  
  let variableDansScopeIf = "BOOM";  
  console.log(variableDansScopeIf); // OUI  
}  
  
console.log(variableDansScopeIf); // error
```

const et **let** sont apparus depuis ES6 pour limité la vie d'une variable dans un bloc. En bloc étant défini par { ... }

Les variables et scopes

```
1 var msg = "Bijour Var Out";  
2  
3 function greet () {  
4     let msg = "Bijour Let In";  
5     console.log(msg);  
6 }  
7  
8 greet();
```

Résultat

Bijour Let In

Les variables et scopes

```
1  let msg = "Out";  
2  
3  console.log(msg);  
4  
5  function greet () {  
6    msg = "In";  
7    console.log(msg);  
8  }  
9  
10 console.log(msg);  
11 greet();  
12 console.log(msg);
```

Résultat

```
Out  
Out  
In  
In
```

Les variables et scopes

```
1  let msg = "Out";
2
3  console.log(msg);
4
5  function greet () {
6      msg = "In";
7      const msg2 = "Const In";
8      console.log(msg + msg2);
9  }
10
11 console.log(msg);
12 greet();
13 console.log(msg + msg2);
```

Résultat

```
Out
Out
InConst In
C:\projects\js\web1\varAndScope.js:13
console.log(msg + msg2);
                  ^
ReferenceError: msg2 is not defined
```

Les types

String

Number

Boolean

Null => rien

Undefined => non défini

Symbol (ES6)

BigInt ($> 2^{53}$) (ES6)

Object

Opérateurs arithmétiques

Opérateur	Nom de l'opération associée
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo (reste d'une division euclidienne)
**	Exponentielle (élévation à la puissance d'un nombre par un autre)

Opérateurs d'affectation

Opérateur	Définition
<code>+=</code>	Additionne puis affecte le résultat
<code>-=</code>	Soustrait puis affecte le résultat
<code>*=</code>	Multiplie puis affecte le résultat
<code>/=</code>	Divise puis affecte le résultat
<code>%=</code>	Calcule le modulo puis affecte le résultat

Opérateurs d'incrémentation

Exemple (opérateur + variable)	Résultat
<code>++x</code>	Pré-incrémentation : incrémente la valeur contenue dans la variable x, puis retourne la valeur incrémentée
<code>x++</code>	Post-incrémentation : retourne la valeur contenue dans x avant incrémentation, puis incrémente la valeur de \$x
<code>--x</code>	Pré-décrémentation : décrémente la valeur contenue dans la variable x, puis retourne la valeur décrémentée
<code>x--</code>	Post-décrémentation : retourne la valeur contenue dans x avant décrémentation, puis décrémente la valeur de \$x

Concaténation

```
const prefix = "En JS, la concaténation prend se fait ";  
const suffix = "avec le signe +, qui est aussi l'opérateur d'addition";  
  
const preSuffix = prefix + suffix;  
console.log(preSuffix);  
  
// En ES6, on a corrigé ca avec ` et ${}  
console.log(`${prefix}${suffix}`);
```

Opérateurs de comparaison

En javascript, les égalités sont parfois étonnantes, ceci vient principalement de la coercion

```
// Avec les triple =, on compare la valeur mais aussi son type
console.log( "1 == \"1\"", 1 == "1" ); // true
console.log( "1 === \"1\"", 1 === "1" ); // false
console.log( "1 === 1", 1 === 1 ); // true
console.log( "{} === {}", {} === {} ); // false ???
console.log( "[] === []", [] === [] ); // false
console.log( "[] == ![]", [] == ![] ); // true ???
console.log( "[] == []", [] == [] ); // false
console.log( "1 !== 1", 1 !== 1 ); // false
console.log( "1 > 1", 1 > 1 ); // false
console.log( "1 >= 1", 1 >= 1 ); // true
// ATTENTION, nombreux problèmes dus à la coercion
```


Structures de contrôle

```
const min = 3;
const max = 99;

if (min > max) {
  console.log("error");
} else if (min === max) {
  console.log("equal");
} else {
  console.log("looks great");
}
```

Dans un contexte booléen if(monTest), toute valeur en js est évaluée à true en dehors de

- false
- 0
- ""
- null
- undefined
- NaN => Not a Number

qui sont elles falsy

```
1  const min = 3;
2  const max = 99;
3
4  switch (min < max) {
5    case false:
6      switch (min === max) {
7        case true:
8          console.log("equal");
9          break;
10         default:
11           console.log("error");
12       }
13     break;
14   default:
15     console.log("looks great");
16 }
```

Les boucles

Mot clé / keyword

for (let x = ..., condition d'arrêt, incrémentation ou décrémentation) { // code }

```
1 const classMatesNumber = 31;
2
3 for (let i = 0; i < classMatesNumber; i++) {
4   console.log("Élève " + i);
5 }
```

```
// ES6
console.log("=== for of");
for (const number of arrayOfNumbers) {
  console.log(number);
}

console.log("=== for in");
for (const key in arrayOfNumbers) {
  console.log(arrayOfNumbers[key]);
}
```

Mot clé / keyword

for (const value of values) { // code }
for (const keyOfValue in values) { // code }

on parcourt toute ma boucle

Les boucles

Mot clé / keyword
`do { // code } while (condition d'arrêt);`

```
1  const classMatesNumber = 31;
2  let i = 0;
3  do {
4      console.log("Élève " + i);
5      i++;
6  } while (classMatesNumber > i);
```

Mot clé / keyword
`while (condition d'arrêt) { // code };`

```
const classMatesNumber = 31;
let i = 0;
while (classMatesNumber > i) {
    console.log("Élève " + i);
    i++;
}
```

Functions

Pourquoi les fonctions ?

code unique

- modification centralisé
- debug facilité
- gain de temps
- moins d'erreur
- une seule responsabilité

fonction récursive

- exercice chrono

Functions

Mot clé / keyword

nom de fonction

arguments optionnels

```
function functionName() {  
    // do something  
}
```

Accolades du début et de fin de bloc

```
// je déclare une fonction avec le mot clé `function`.  
// J'ajoute un paramètre (optionnel)  
function functionName(thisParamIsOptional) {  
    // some code  
}  
// j'appelle la fonction avec le nom de la fonction suivi () avec les paramètres si nécessaire  
functionName("yesWeCan");
```

Functions

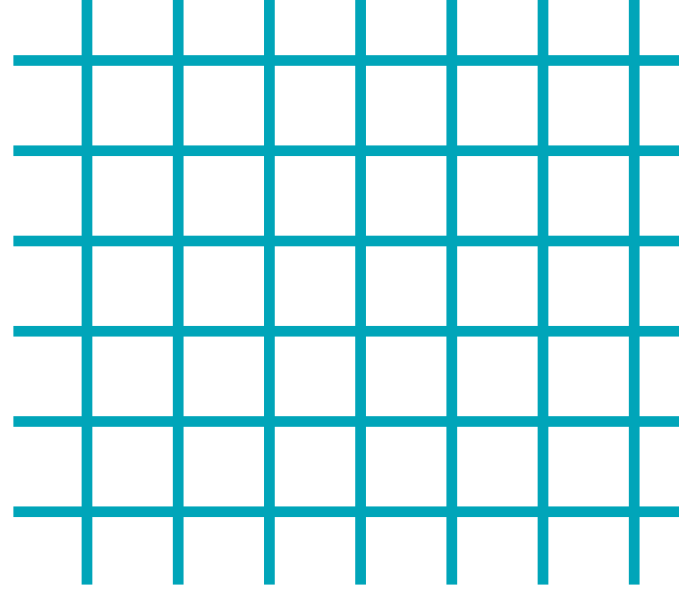
Fonction anonyme

Fonction qui n'ont pas de noms.
On les utilise souvent quand ce sont des fonctions de rappels. (callback function)

```
// fonction anonyme qu'on appelle directement
(function(){
  // some code
})();

// fonction anonyme appelée au bout de 200ms
setTimeout(function() {
  // some code
}, 200);
```

DOM et JS

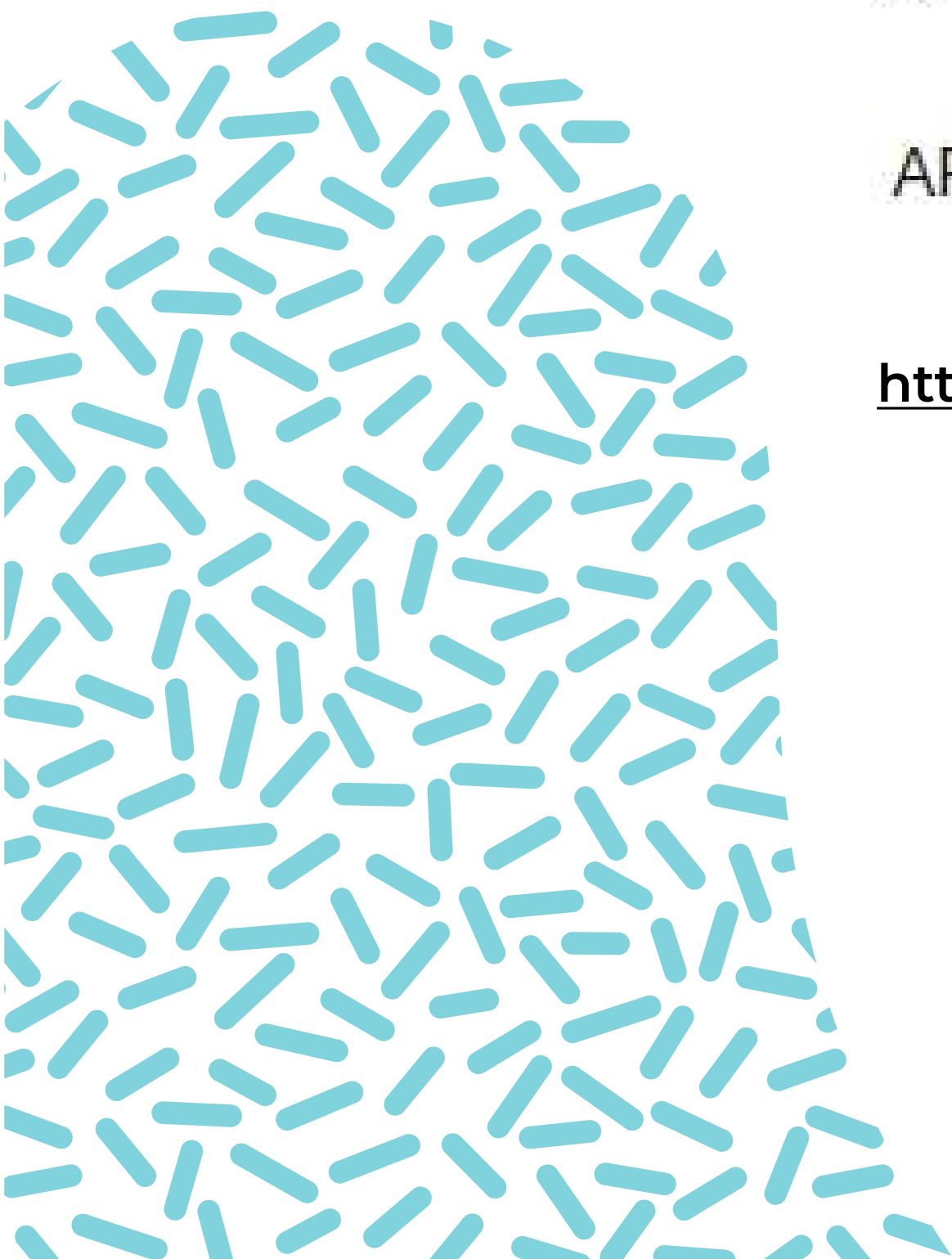


DOM pour Document Object Model

À l'origine, JavaScript et le DOM étaient fortement liés, mais ils ont fini par évoluer en deux entités distinctes. Le contenu de la page est stocké dans le DOM et on peut y accéder et le manipuler via JavaScript, de la sorte qu'on pourrait écrire cette équation approximative :

API(page Web ou XML) = DOM + JS(langage de script)

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



document vs window

```
> window  
< ▶ Window {0: Window, window: Window, self: Window, document: document, name: '', location: Location, ...}
```

window

Objet global d'un tab de votre navigateur. Il englobe document et screen !

- propose les fonctions `setTimeout` et `setInterval`
- objet `location` donnant accès à des fonctionnalités sur URL courante
 - `window.location.href = "https://lesentrecodeurs.com"`
- objet `history` donnant accès à des fonctionnalités sur la navigation
 - `window.history.back()`

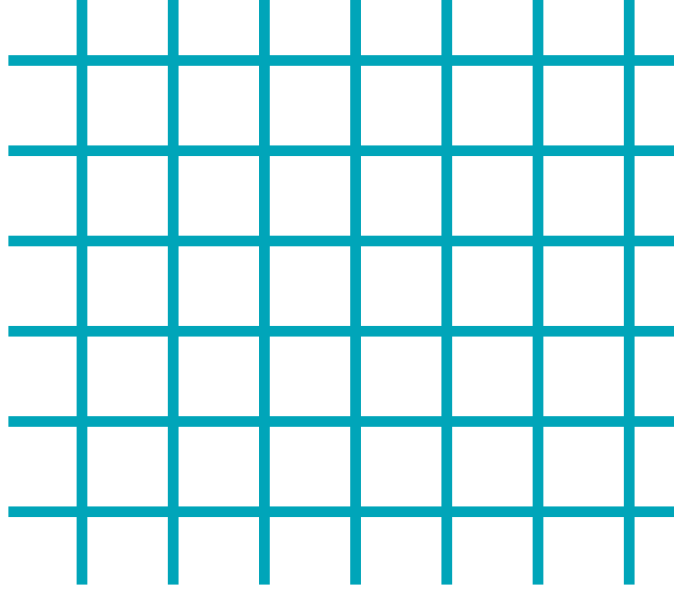
Query selector

`document.querySelector`
`document.querySelectorAll`

```
const htmlEl = document.querySelector("#idVoulu");

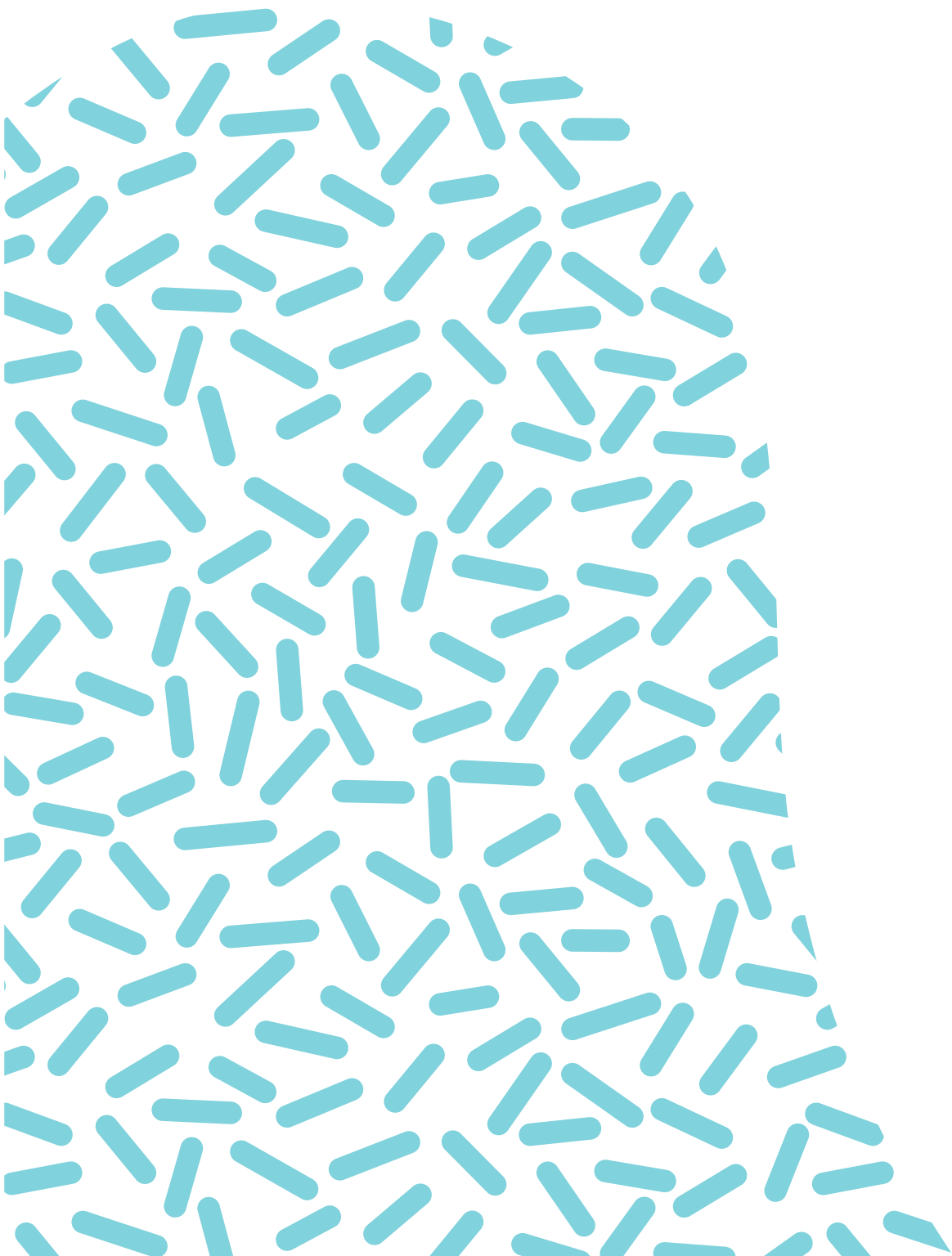
// l'élément avec id #uniqueId
const uniqueIdElOld = document.getElementById("uniqueId");
// l'élément avec id #uniqueId ES6
const uniqueIdEl = document.querySelector("#uniqueId");
// tous les éléments avec class .op
const opEltsOld = document.getElementsByClassName("op");
// tous les éléments avec class .op
const opElts = document.querySelectorAll(".op");
// tous les <p></p> de mon dom ES6
const tagElts = document.querySelectorAll("p");
```


Evenements



click
scroll
mousemove
et bien d'autres

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



Evenements

click

```
const btnDiv = document.getElementById("counterBtn");  
btnDiv.addEventListener("click", anyFunction);
```

Exercice compteur
Exercice youtube
Exercice sidebar

Evenements

mouseenter / mouseleave

```
const squareEl = document.querySelector(".square");  
squareEl.addEventListener("mouseenter", moveRight);
```

Exercice square

Evenements

mousemove

```
const squareEl = document.querySelector(".square");  
squareEl.addEventListener("mouseenter", moveRight);
```

Exercice square 2

Sources

StackOverflow

<https://developer.mozilla.org/fr/docs>

W3School

Oh my code, je parle le javascript, *Sonia Baibou*

Javascript Masterclass, *Pierre Giraud*

et ofc Wikipedia