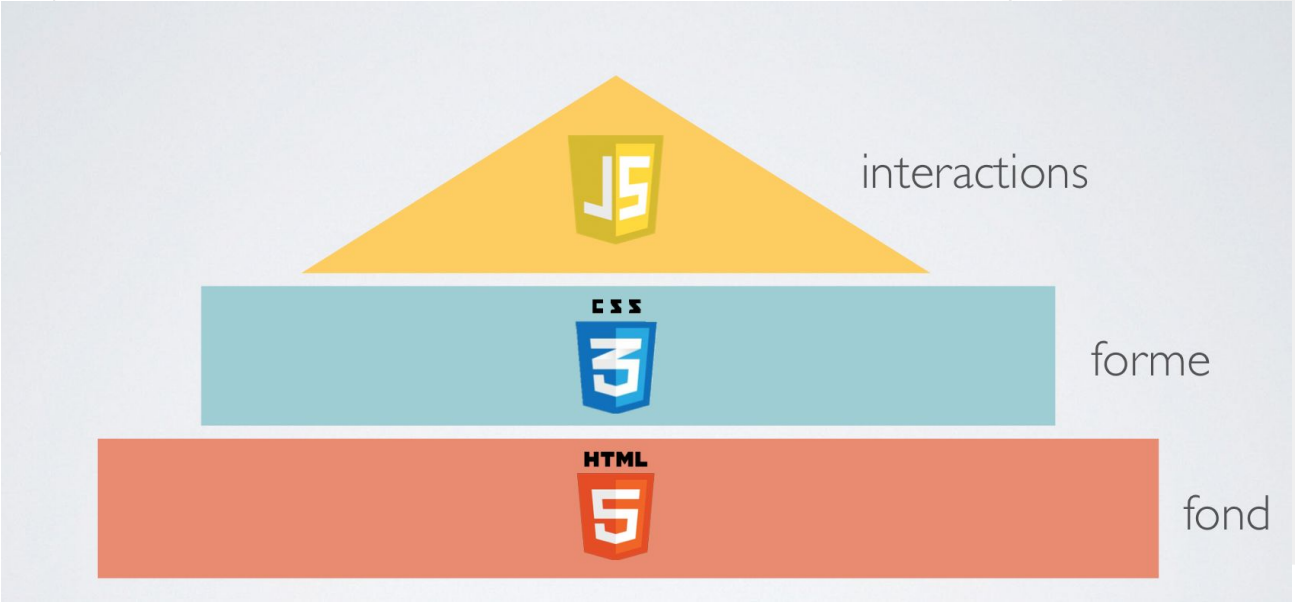
JAVASCRIPT

Rappel



Javascript != JAVA

Point culture

Le JS

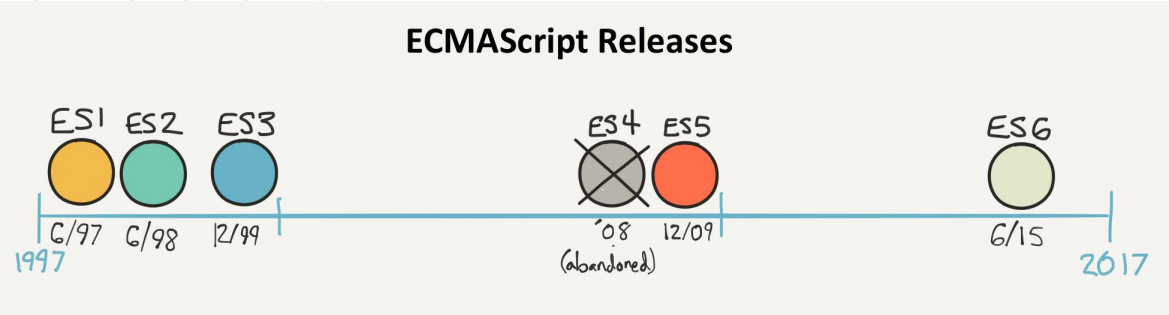
⇒ Inventé en 1995 par Brendan Eich

Objectif initial

⇒ Rendre dynamiques et interactives les pages Web

⇒ Donc **uniquement en front**

Evolution



Dernieres versions:
2016 - ES7
2017 - ES8
2018 - ES9


JS
maintenant

Comment inserer du JS

Fichier externe

```
<script type="text/javascript" src="monscript.js"></script>
```

La placer en fin de document juste avant </body>

Ou dans le <head>

Interne

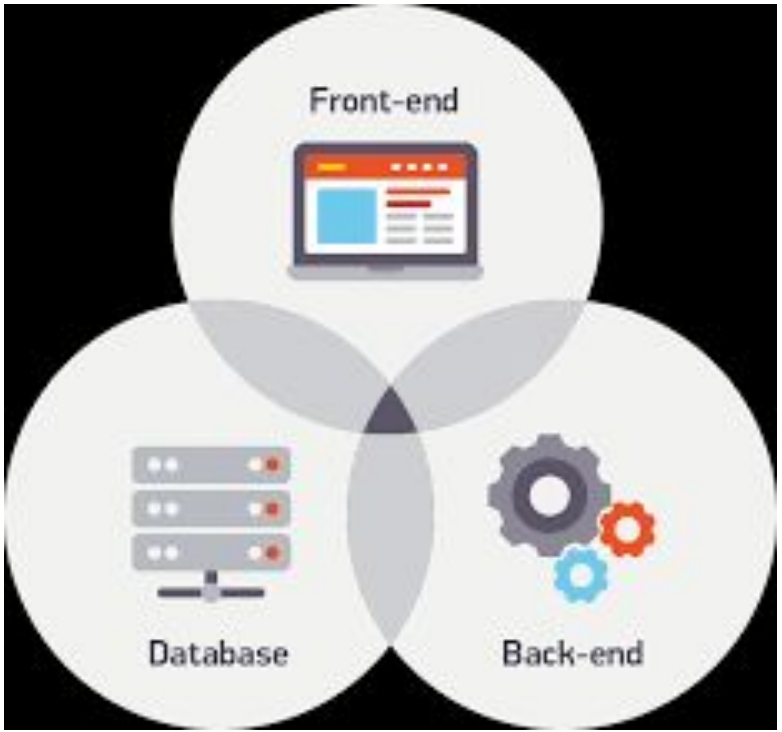
```
<script type="text/javascript">
  // Mon code Javascript
  ...
</script>
```

Hello world

```
<!DOCTYPE HTML>
<html>
 <body>

  <p>Before the script...</p>

  <script type="text/javascript">
   alert( 'Hello, world!' );
  </script>
 </body>
</html>
```



Le Js est présent partout dans la stack web moderne

Client / Serveur

NO SQL:
Mongodb / Mongoose

Note:
AI / machine learning
Tensor Flow Js
⇒ https://js.tensorflow.org/demos/

Front

Frameworks tel que:

- React Js
- Angular Js
- Vue js
- Ember Js

Back

Framework tel que

- Express Js
- Meteor js

JS
La base

Les variables

Avant es6 (2015)

```
var num = 5;
var string = "hello";
var myFunction = function (e) {
  // code
  };
  var tableau = [1, 2, 3, "4", {num: 5}, "six"];
  var object = {
     item1: "une string",
     item2: 3736,
     item3:  {
        sous _item1: 6252,
        sous_item2: "une autre string"
        sous_item3: function (param) {
        // code
        }
  }
}; // Suivant les règles du JSON
```

“Débugguer”

```
var maVariable = 4;
console.log(maVariable);

console.log("Ma variable est:", maVariable);
```

Les variables

Suite au es6 (2015)

const et let

```
let num = 5
const pi = 3.14
```

Arrow function

```
const myFunction = (e) => {
// code
}
```

Pareil que:

```
const myFunction = function(e) {
    // code
}
```

String concaténation

```
let myString = `${variable} vous dit bonjour`
```

Les opérateurs

Arithmetic

```
let x = 5, y = 8;
x = x+y;
x = x-y;
x= x*y;
x= x**y (Exponentiel);
x = x/y;
x = x%y;
x++;
x--;
```

Comparaison

```
let x = 8, y = 10;
if(x==y) { ... } //égale
if(x===y) { ... } //égale et même type
if(x!=y) { ... }
if(x!==) { ... }
if( x > y && x >=10 ) { ... }
if( x <= 40 || y>78 ) { ... }
```


Operation ternaire

```
if( x === y ) { A } else { B }
x === y ? A : B ;
```

Les objects

```
const arthur =
  {
    name : "Arthur", // Proprieté name ayant une valeur Bernard de type String
    surname: "Delatour", // Proprieté surname ayant une valeur Dupond de type String
    age: 10, // Proprieté age ayant une valeur 10 de type int
    display : function () { // Proprieté display ayant un type fonction
      console.log(this.name, this.surname, "a", this.age, "ans");
    },
    happyBirthday: function() { // Proprieté display ayant un type fonction
      this.age += 1;
      return this.age;
    }
  }

arthur.display();
arthur.happyBirthday();
arthur.display();
```

Une propriété peut avoir un type autre que string et int.

Une fonction peut être la valeur d'une propriété 🧐

Playground:

👉https://repl.it/@LoicBerthelot/nodeJs-Object

Les Expressions conditionnelles

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

```
switch(variable)
case A:
   //code
break;
case B: {
// code
break;
}
default:
// code
```


Example:

https://www.w3schools.com/js/tryit.asp?filename=tryjs_switch

Les Loop

For Loop

```
for(let i = 0; i< 50; i++){
console.log(i);
}
```

While

```
while (i < 10) {
text += "The number is " + i;
       i++;
}
```

Do While

```
do {
text += "The number is " + i;
i++;
}while (i < 10);
```

For each Loop

```
const array = ['a', 'b', 'c'];

array.forEach(function(element) {
  console.log(element);
});

// Pareil que
array.forEach((element) => {
  console.log(element);
});
```

Array

Ajout début et fin au tableau

```
let array = [1,2,3];

// Ajout en fin
array.push(4); // [1,2,3,4]
// Ajout au début
array.unshift(0); // [0,1,2,3,4]

console.log("array", array);
```

Manipuler des tableaux, insérer , retirer

```
let array = ['a','b','d'];
// Enlève 2 éléments à partir de l'index 0
array.splice(0, 2);
console.log("array", array); // [ 'd' ]

let array2 = ['a','b','d'];
// Enlève 1 élément à partir de l'index 2 et insère 'c'
array2.splice(2, 1, 'c');
console.log("array2", array2); // [ 'a', 'b', 'c' ]
```

Listing des fonctions

👉https://mzl.la/33t7il8

Collection (array avec objects)

```
let names = [{
    name: "Bernard",    // Proprieté name ayant une valeur Bernard de type String
    surname: "Dupond",  // Proprieté surname ayant une valeur Dupond de type String
    age: 24             // Proprieté age ayant une valeur 24 de type Int
  },
  {
    name: "Henry",
    surname: "Dubois",
    age: 20
  },
  {
    name: "Thibault",
    surname: "Tuillier"
  }
];
// Qui cela enlève t'il ?
names.splice(1, 1);
// Ajout proprieté age
names[1].age = 18;
console.log("names", names);
```

Playground:

👉https://repl.it/@LoicBerthelot/Online-NodeJs

JS
Fonctions pour array

Map

```
let array = [1, 4, 9, 16];

function timeTwo(val) {
  return val*2
}
// Pass a function to map
const mappedArray = array.map(timeTwo);

// Or do it inline it's the same
const mappedArray2 = array.map(x => x*2);

console.log(mappedArray);
// expected output: Array [2, 8, 18, 32]
```

Sandbox 👇
https://repl.it/@LoicBerthelot/Map-nodejs

Filter

```
let nbs = [5, 42, 26, 16, 28, 12];

// Filtre le tableau avec des valeurs supérieur à 20
const filteredNbs = nbs.filter(number => {
  if(number > 20) return true;
  return false
});

console.log(filteredNbs);
// expected output: [ 42, 26, 28 ]
```

D’autres exemples 👇
https://repl.it/@LoicBerthelot/Filter-nodejs

Et encore pleins d'autres

- **Sort** : fonction de triage
  👉https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/sort
- **Find** : fonction qui permet de trouver une valeur
  👉https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/find
- **Join** : fonction transformant un tableau en une chaîne de caractères
  👉https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/join
- **IndexOf**: fonction trouvant la position dans un tableau d'un élément
  👉https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/indexOf
- **Concat**: fonction qui concatainer deux tableaux
  👉https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/concat


JS
La base ++

Les classes

```
class Car {
  constructor(brand, seats) {
    this.brand = brand;
    this.seats = seats;
  }
  display() {
    return "I have a " + this.brand + " with " + this.seats + " seats";
  }
}

const myFirstCar = new Car("Ford", 4);
const mySecondCar = new Car("Tesla",2);

console.log(myFirstCar.display());
console.log(mySecondCar.display());
```

Sandbox 👇

https://repl.it/@LoicBerthelot/Class-Nodejs

Asynchrone en JS

```
const data = getData("https://api.meteo.com"); // Opération pouvant prendre du temps
//...
//...
// Le JavaScript est un langage qui a été pensé pour évoluer dans un environnement
monothread et asynchrone.
console.log("Data", data); // Data ici est null
```

Comment faire pour contrôler son flow?

Les callbacks

En javascript on utilise des callbacks afin de contrôler le flow asynchrone

```
function prendsDuTemps(callback) {
  setTimeout(
    function() {
      // code
      callback();
    },
  1000);
}

const callback = function () {
console.log("finish");
}

prendsDuTemps(callback);
```

Playground 👇
https://repl.it/@LoicBerthelot/Callback-nodejs

```
// Peut vite devenir infernal
demandeInfoServeur1(function(data1){
  demandeInfoServeur2(function(data2){
    demandeInfoServeur3(function(data3){
      console.log("Code illisible");
    })
  })
})
```

Promise

```
const promise1 = new Promise(function(resolve, reject) {
  /*
  GROS TRAITEMENT OU APPEL SERVEUR ASYNCHRONE
  */
  resolve({ message: "Success" });
});

promise1
.then(function(value) {
  console.log(value);
})
.catch(function(error) {
  console.error(value);
});
```

👉 https://repl.it/@LoicBerthelot/Promise

Async / Await

```
function resolveAfter2Seconds() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000);
  });
}

async function asyncCall() {
  console.log('calling');
  // Va attendre le résulat de cette fonction
  var result = await resolveAfter2Seconds();
  console.log(result);
}

asyncCall();
```

👉 https://repl.it/@LoicBerthelot/AsyncAwait

JS
Manipuler le DOM

Qu’est ce que le DOM ?

Le HTML DOM est un standard qui permet de:

- Récuperer
- Modifier
- Enlever

Des elements HTML dans une page

Pourquoi manipuler le DOM?

Objectifs:

- Trouver un élément
- Modifier un élément
- Réagir en fonction des actions de l’utilisateur
- Animer un élément
- **Communiquer avec le server (AJAX)**

1) Sélecteurs identiques au CSS

Selectionner son élement HTML

document.getElementById(“id”)
document.getElementsByClassName(“class”)

```
// Récuperer par id
const para1 = document.getElementById("my-first-paragraph");
// Récuperer par className (attention renvoit un tableau)
const para2 = document.getElementsByClassName("my-second-paragraph");
```

👇

https://repl.it/@LoicBerthelot/SelectElement

2) Manipuler le DOM

Une possibilité quasi illimité

- **Ajouter**
- **Modifier**
- **Enlever**

```
// SELECTIONNER ELEMENT
const para1 = document.getElementById("my-first-paragraph");

// Le manipuler
para1.innerHTML = "<span style='font-weight:bold'>Nouveau</span> paragraph"
para1.style.color = "blue";
para1.className = "active-paragraph";
```

👇

https://repl.it/@LoicBerthelot/Manipuler-HTML

3) Event handler

```html
<body>
    <p id="my-first-paragraph">Hello World 1 </p>
    <button onclick="changeParagraph()">Change color</button>
    <script src="script.js"></script>
</body>
```

👇 https://repl.it/@LoicBerthelot/Html-with-button

```js
function changeParagraph(){
  // SELECTIONNER ELEMENT
  const para1 = document.getElementById("my-first-paragraph");

  // Le manipuler
  para1.innerHTML = "<span style='font-weight:bold'>Nouveau</span> paragraph"
  para1.style.color = "blue";
  para1.className = "active-paragraph";
}
```

TP!

https://github.com/berthelol/19-20_TWOIng4_TP3