



# Initiation à JavaScript

**Module 01 – Introduction au JavaScript**

- Ce cours s'accompagne d'une iconographie afin de mieux se repérer :



Notions de sécurité



Accessibilité



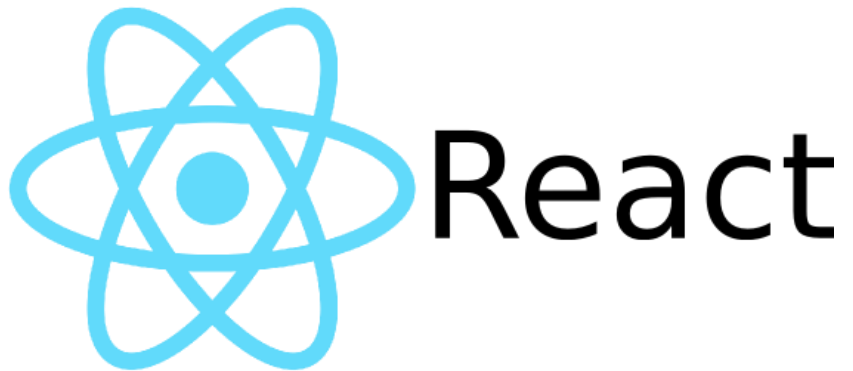
Eco-conception

## Objectifs

- Connaitre les usages du JavaScript d'aujourd'hui et d'hier
- Connaitre les caractéristiques de base du langage pour écrire des algorithmes

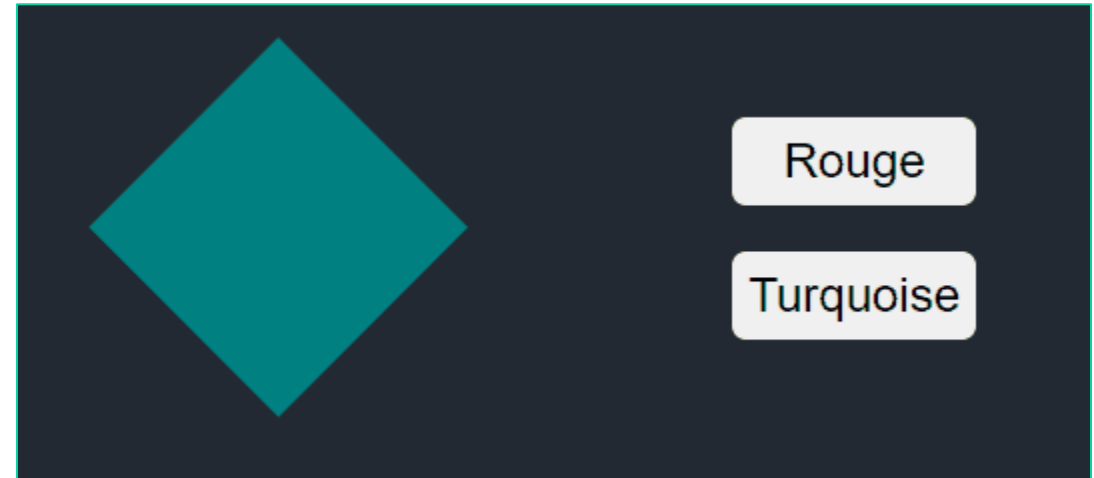
# L'usage du JavaScript aujourd'hui

## L'usage du JavaScript aujourd'hui

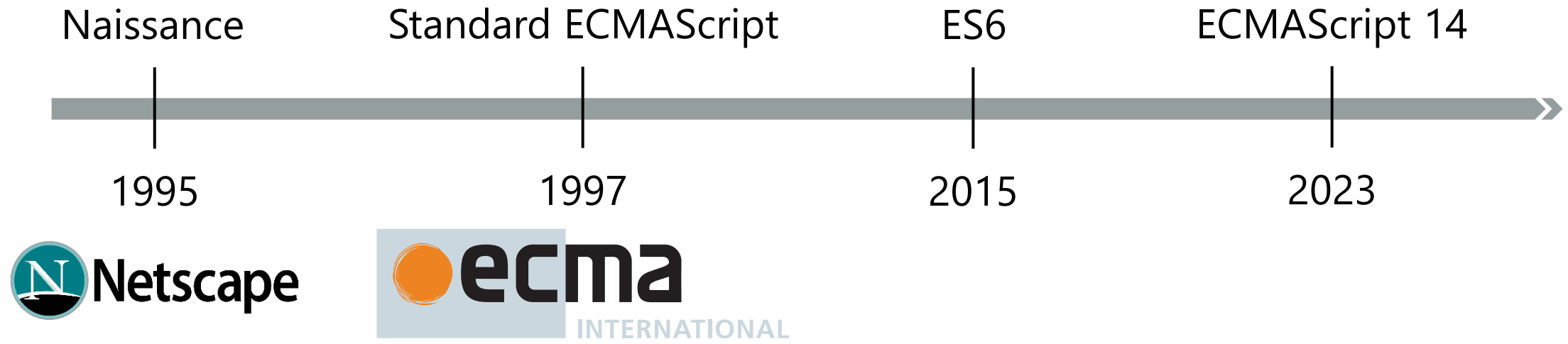


# Le JavaScript avant

## L'usage du JavaScript au tout début



## Historique et acteurs



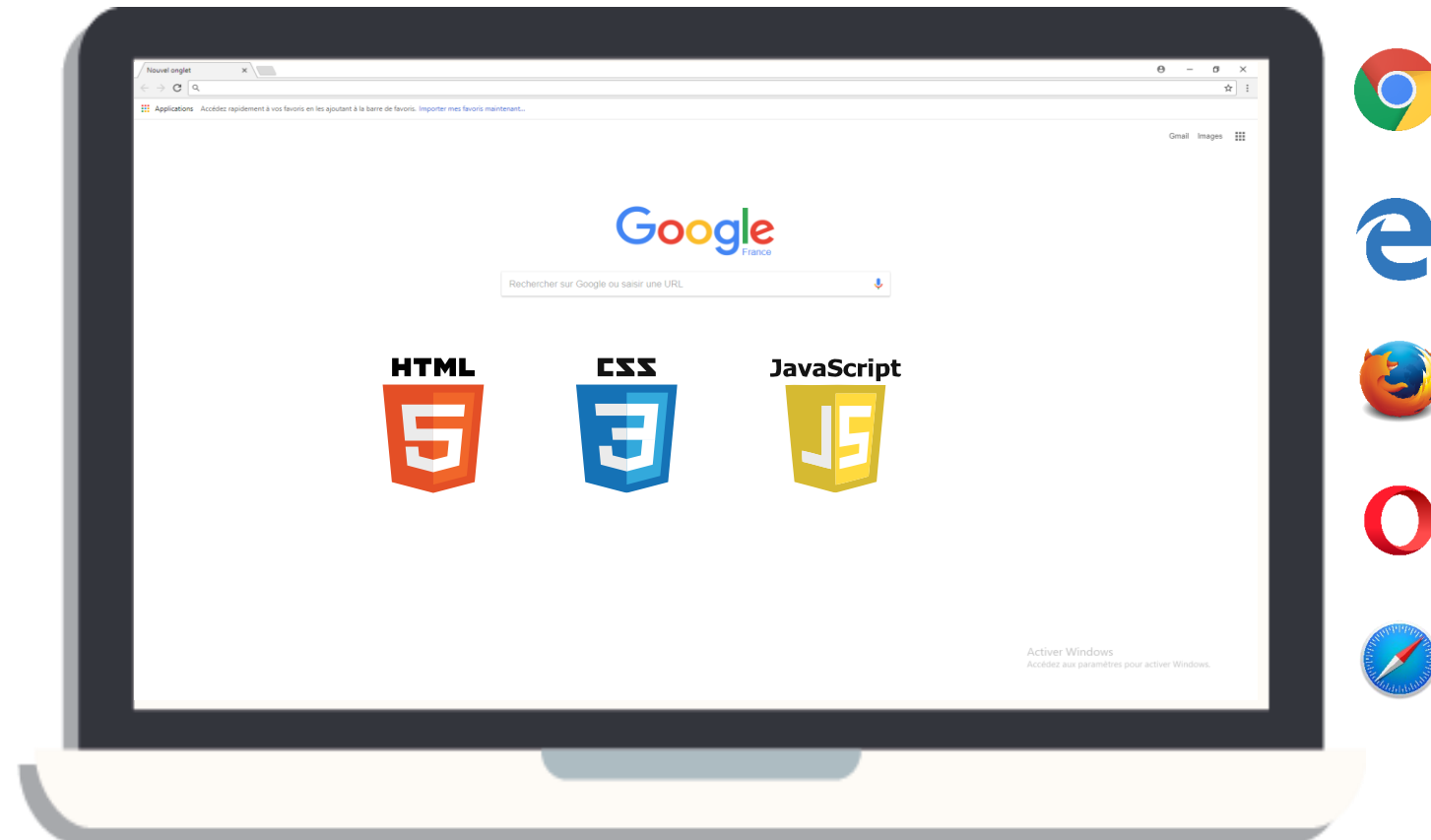
<https://www.ecma-international.org/memento/tc39-m.htm>

*The ECMAScript™ specification has been developed by Ecma TC39 whose membership includes all major browser vendors.*



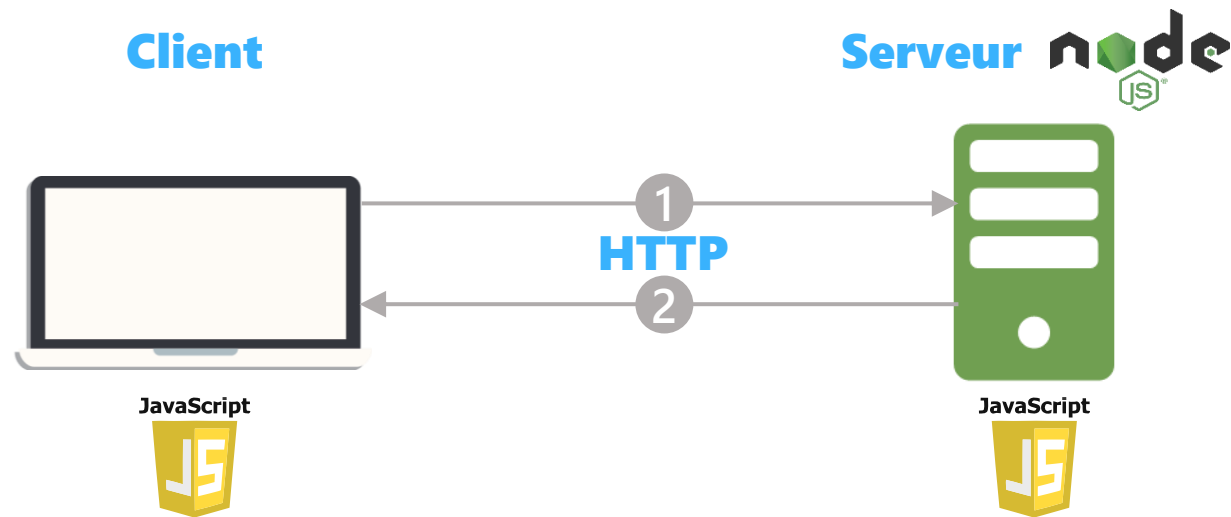
# Le mode d'exécution du JavaScript

## Un langage interprété par les navigateurs



# L'écosystème du JavaScript

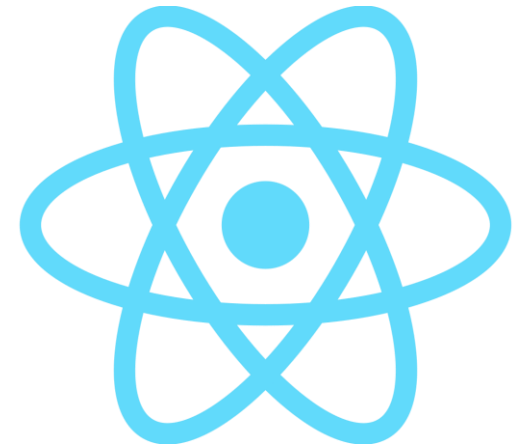
## Un langage fullstack



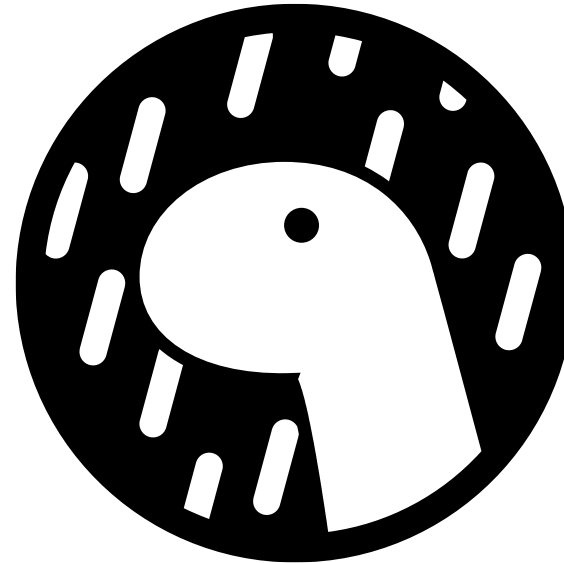
## Le moteur d'exécution

- Embarqué par le navigateur mais peut être autonome
-  est le moteur de 
-  est utilisé par  et 

## Des bibliothèques côté client




## Des librairies côté serveur



## Des gestionnaires de dépendances



est le gestionnaire de paquets officiel de 

### A quoi sert un gestionnaire de dépendances ?

Un **projet** est comme un **squelette**. Il possède les éléments de base.

Parfois, pour des besoins spécifiques, il faudra le compléter avec des **librairies** :

Envoyer un email au moment de l'inscription, générer des documents au format PDF, avoir un module de partage sur les réseaux sociaux, ...

Le **gestionnaire de dépendances** fonctionne comme un catalogue dans lequel nous allons pouvoir faire appel à ces dépendances.



# Les outils du cours

## Les outils

- Pour développer en JavaScript, il faut un éditeur
- Il existe beaucoup d'éditeurs de code pouvant gérer le JavaScript :
  - Visual Studio Code
  - WebStorm
  - Atom
  - Vim
  - Sublime Text
  - ...
- Pour ce cours, Visual Studio Code sera utilisé



## Les outils

- Par défaut, les éditeurs ne possèdent pas de moteur pour interpréter le JavaScript
- Il faut donc compléter l'installation par Node.js



# Démonstration

# Introduction au langage JavaScript

## Les langage JavaScript

- Le JavaScript respecte les paradigmes :
  - Objet
  - Impératif
  - Fonctionnel
- Le JavaScript est un langage interprété



# Les variables JavaScript

## Les variables



Typage dynamique

Déclaration :

```
var x  
let z  
const y
```

Commentaires :

```
// pour une ligne  
  
/*  
Sur plusieurs lignes  
*/
```



## Les variables

- Les principaux types primitifs (immuables) :
- Les types complexes : Date, Math, ...

*number*  
*string*  
*boolean*  
*undefined*  
*null*

# Démonstration

# Les opérateurs JavaScript

# Les opérateurs

- La principale nouveauté

===    !==

```
Console.log("1" == 1) //true
```

```
console.log("1" === 1) //false
```



Les autres opérateurs : <, <=, >=, >, ==, !=

# Les structures de code en JavaScript

## Les structures de code

- Les conditionnelles
  - if ... else
  - switch
- Les boucles
  - while
  - do ... while
  - for, for ... in, for ... of
- La gestion d'erreurs
  - try ... catch

# Les collections en JavaScript

## Les collections

- Les collections permettent de rassembler des éléments
- Il existe plusieurs types de collection dont les principaux sont :
  - Array : Fonctionne comme un tableau
  - Map : Fonctionne comme un dictionnaire avec un couple clé/valeur
  - Set : Fonctionne comme un tableau mais sans doublon



# Les tableaux

- Déclaration

```
const etablisements = ['école maternelle', 'école primaire'];  
const eleves = [];  
const profs = new Array(15);
```

- Utilisation

- Comme en Java
- Taille non fixée

- Propriétés et méthodes disponibles

- length, push(), pop(), slice(), sort(), toString()...

# Les dictionnaires

- Déclaration

```
let dictionnaire1 = new Map([[1, 'Jaune'], [2, "Bleu"], [3, "Rouge"]]);
```

- Utilisation

- Couple clé/valeur
- Taille non fixée

- Propriétés et méthodes disponibles

- set(), get(), ...

# Les sets

- Déclaration

```
let setExemple = new Set();
```

- Utilisation

- Comme en Java
- Collection sans doublon

- Propriétés et méthodes disponibles

- add(), has(), ...

# Démonstration

# Intégration du JavaScript dans une page HTML

# Intégration du JavaScript dans une page HTML

- Déclaration

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

- Emplacement :

- Il est possible de déclarer les ressources JS soit dans le **head** ou avant la balise fermante du **body**

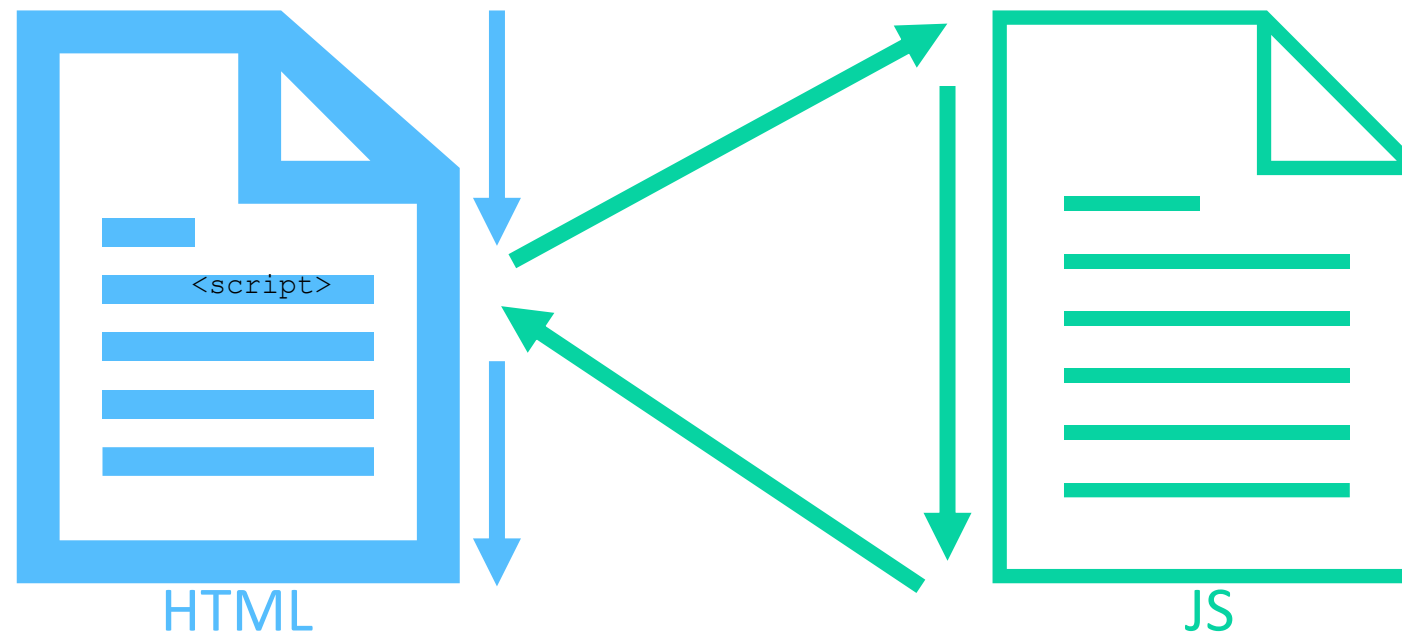
- Temps de chargement :

- La lecture par le navigateur d'un fichier HTML est réalisée de haut en bas. Quand le moteur de rendu arrive sur une balise `<script>`, il interrompt la lecture du HTML pour lire celle du script. Une fois terminée, il retourne au HTML.

- Conseil :

- Il est préférable de déclarer les scripts avant la fermeture de body

# Intégration du JavaScript dans une page HTML



# Intégration du JavaScript dans une page HTML

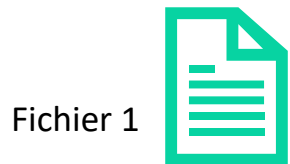
- async et defer :
  - Attributs HTML
  - async : Charge de manière asynchrone le script
  - defer : Diffère l'exécution du script à la fin du chargement du document



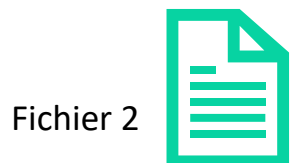
# Modules JavaScript

## Module JavaScript

- Si le code devient trop important, il peut être utile de le découper en plusieurs modules
- Pour cela, il faut utiliser les instructions `import` et `export`



```
export let value = 'Hello !';
```



```
import {value} from './module.js'
```

# Démonstration

# Outils de debug JavaScript du navigateur

## Outils de debug - Navigateur

- Quand le code est interprété par le navigateur, la seule solution pour le déboguer est d'utiliser les outils intégrés au navigateur
- L'accès se fait par **F12** et les onglets **Console** et **Sources** (pour Chrome)

# Démonstration

TP