

Le développement côté client avec JavaScript

Module 1 - Introduction au JavaScript

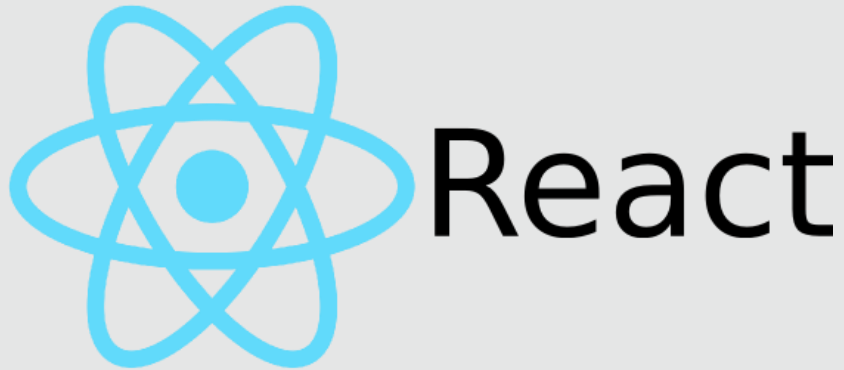


Objectifs

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

Introduction au JavaScript

L'usage du JavaScript aujourd'hui



Introduction au JavaScript

L'usage du JavaScript au tout début

JavaScript



Historique et acteurs

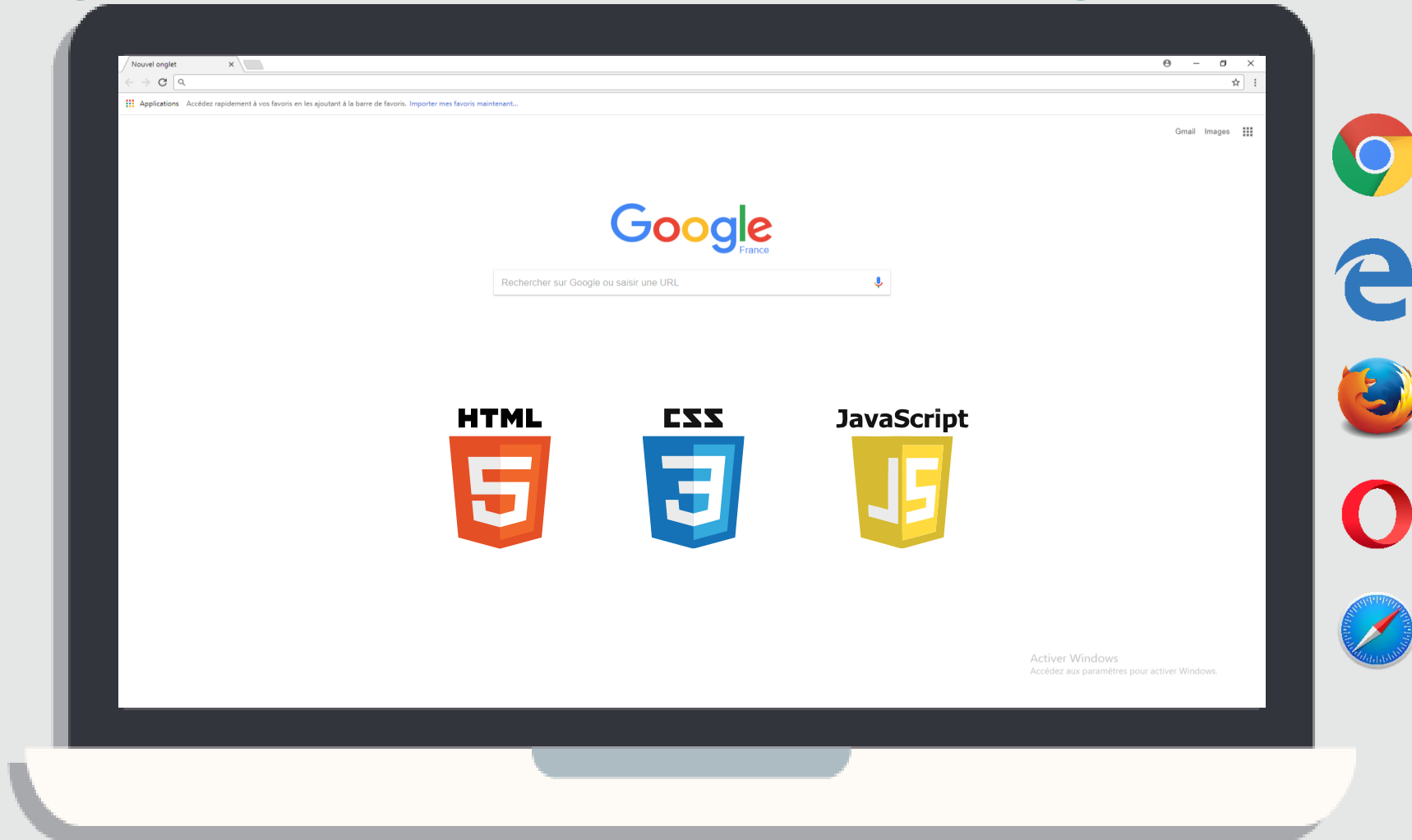


<https://www.ecma-international.org/memento/tc39-m.htm>

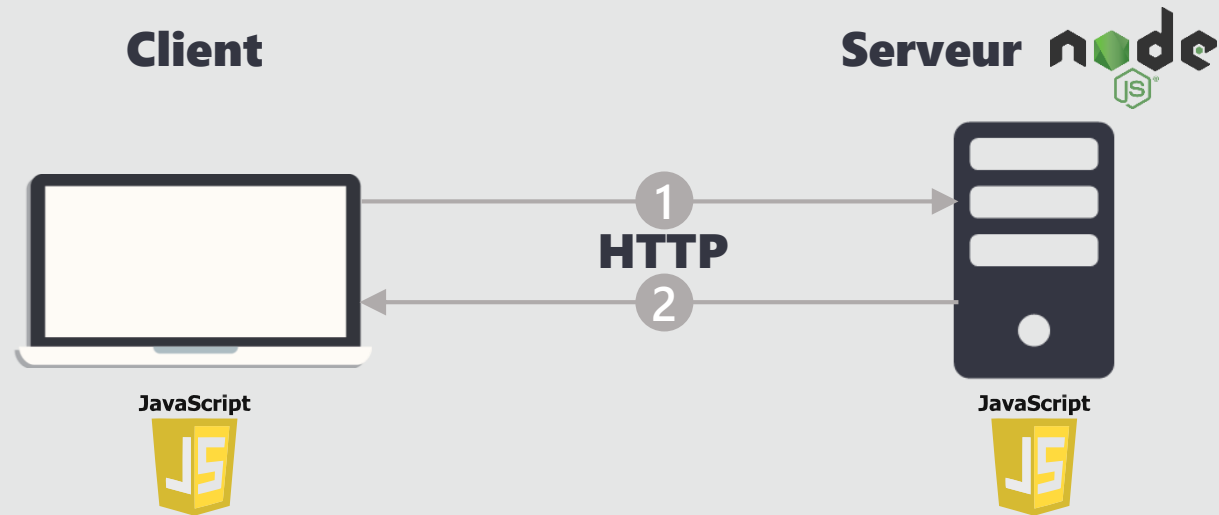
The ECMAScript™ specification has been developed by Ecma TC39 whose membership includes all major browser vendors.

Introduction au JavaScript

Un langage interprété par les navigateurs



Un langage Full Stack



Le moteur d'exécution

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

Des bibliothèques côté client


-  sert à faciliter la manipulation du DOM d'une page HTML

```
$( "a" ).click(function( event ) {  
    alert( "Vous venez de cliquer sur le lien" );  
});
```

-  sert à écrire des clients riches en utilisant une architecture MVC




```
@Component ({  
    selector:      'balise',  
    templateUrl:  './templateBalise.html',  
    providers:    [ MonService ]  
})  
export class MonComponent implements OnInit {  
    /* . . . */  
}
```

Des librairies côté serveur

-  sert à développer des applications distribuées en JavaScript
- Mise en place d'un serveur HTTP :


```
const http = require('http');  
  
http.createServer((request, response) => {  
    //Code du comportement du serveur  
}).listen(8080); // Démarrage du serveur sur le port 8080
```

Des gestionnaires de dépendances

-  est le gestionnaire de paquets officiel de 
-  est un gestionnaire de paquets orienté *front* (HTML, CSS, JS)

L'arrivée du TypeScript



- 
Microsoft
- Open source
- Surlangage à JavaScript
- Transcompilation en JavaScript

Démonstration



Le langage JavaScript

- Le JavaScript respecte les paradigmes :
 - Objet
 - Impératif
 - Fonctionnel
- Le JavaScript est un langage interprété
- Le JavaScript est déroutant pour les développeurs objet



Les variables



```
var maVariable;
```

- Les principaux types primitifs (immuables) :

```
boolean  
null  
undefined  
number  
string
```

- Les types objet : Date, Math, Array, Object, Function, NaN...

Les fonctions

```
function nom([parametre[, parametre[, ... parametre]]])  
{  
    instructions  
    [return ...]  
}
```


Les opérateurs de base

- Presque comme en Java !
- La principale nouveauté : l'égalité stricte

=== ! ==

```
console.log("1"==1);    //true
```

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

Démonstration



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 principaux objets disponibles

- String
 - `charAt()`, `concat()`, `indexOf()`, `substring()`...
- Number
 - `MAX_VALUE`, `MIN_VALUE`, `isInteger()`, `isNaN()`...
- Math
 - `random()`, `floor()`, `ceil()`...
- Date
 - `getDate()`, `getMonth()+1`, `getFullYear()`, `getHours()`... `setMinutes()`, `setSeconds()`... `toLocaleDateString()`...
- Error
 - `name`, `message`
- Propriétés et variables globales
 - `Infinity`, `NaN`, `undefined`, `isNaN()`, `Number()`, `parseFloat()`, `parseInt()`...

Les tableaux

- Déclaration

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

- Utilisation

- Comme en Java
- Taille non fixée

- Propriétés et méthodes disponibles

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

Démonstration



TP



Conclusion

- Vous savez écrire des algorithmes simples en JavaScript