INTRODUCTION

ECMASCRIPT 6

CONST ET LET

- const et let remplacent var
- les valeurs stockés dans const sont immutable, elle ne peuvent être ré-assigner | *exception faite pour les objets*
- Let est similaire a const, mais n'est pas imputable
- Elles ne sont disponible que dans leur scope

ARROW FUNCTIONS I

- Les arrow functions introduisent une syntaxe plus lisible, plus structuré :

```
// ES5

function myFunc(name) {
    return 'Hello' + name;
}

console.log(myFunc('said'));

// output
// Hello said
```

```
// ES6 Arrow function

const myFunc= name =>{
    return `Hi ${name}`;
}
console.log(myFunc('Said'))// output Hi Said

// or even without using arrow or implement `return` keyword
const myFunc= name => `Hi ${name}`;

console.log(myFunc('Said')) // output Hi Said
```

ARROW FUNCTIONS II

- Vous pouvez également utiliser les arrow functions avec les fonction native comme map, filter ou reduce.

```
// ES5
const myArray=['tony','Sara','Said',5];

let Arr1= myArray.map(function(item){

    return item;
});
console.log(Arr1);//output (4) ["tony", "Sara", "Said", 5]

//ES6 Syntax

let Arr2 = myArray.map(item => item);
console.log(Arr2); //output (4) ["tony", "Sara", "Said", 5]
```

TEMPLATE STRINGS

- Dite adieux a l'opérateur '+' pour la concaténation de chaines de caractères, les templates strings permettent une syntaxe plus élégante.

```
//ES5

function myFunc1(name,age){
 return 'Hi' + name + ' Your age is' + age + 'year old!';
}
console.log(myFunc1('Said',22))
//output -->Hi Said,Your age is 22year old!
```

```
//ES6
const myFunc= (name,age)=>{

  return `Hi ${name},Your age is ${age}year old!`;
}
console.log(myFunc('Said',22))
//output--> Hi Said,Your age is 22year old!
```

PARAMÉTRES PAR DÉFAUT

- ES6 permet d'attribuer un paramètre par défaut a ses fonctions, cela permet d'éviter des variables undefined lorsqu'on invoque la fonction

```
const myFunc=(name,age)=>{
    return `Hello ${name} your age is ${age} year old?`
}
console.log(myFunc('said'))
// output= Hello said your age is undefined year old?
```

```
const myFunc=(name,age=22)=>{
    return `Hello ${name} your age is ${age} year old?`
}
console.log(myFunc('said'))
// output= Hello said your age is 22  year old?
```

DÉSTRUCTURATION D'OBJETS ET DE TABLEAU

- ES6 introduit une nouvelle syntaxe pour accéder au variables stocker dans les tableaux et objets

```
// ES5 syntax
const contacts={
    name:'said',
    familyName:'Hayani',
    age:22
}

let name=contacts.name;
let FamillyName=contacts.familyName;
let myAge=contacts.age;
console.log(name)
console.log(FamillyName)
console.log(myAge)
// output
// said
// Hayani
// 22
```

```
//ES6
const contacts={
    name:'said',
    familyName:'Hayani',
    age:22
}

let{name,familyName,age}=contacts
console.log(name)
console.log(familyName)
console.log(age)
// output
// said
// Hayani
// 22
```

IMPORT EXPORT I

- Exemple simple d'utilisation d'import et d'export

```
//ES6

export default  function detail(name,age){
    return `Hello ${name},your age is${age} year old!`;
}
```

```
import detail  from './detailComponent'

console.log(detail('Said',20))
//output >> Hello Said,your age is 22  year old!
```

IMPORT EXPORT II

- Exemple d'importation multiple, lorsqu'on exporte plusieurs éléments

```
import {detail,userProfile,getPosts}  from './detailComponent'

console.log(detail('Said',20))
console.log(userProfile)
console.log(getPosts)
```

SPREAD OPERATOR

- Permet de déstructurer un array, le spread operator retourne toutes les data contenu dans l'array, très pratique lorsqu'on travaille avec des pure functions.

```
const doctors = ['Peter Capaldi', 'David Tennant', 'Matt Smith', 'Jodie Whittaker']

console.log(...doctors)
// logs Peter Capaldi David Tennant Matt Smith Jodie Whittaker
```