Call me number six

Les nouveatés de l'ES6

CONFIGURATION DE L'ENVIRONEMENT

Les langages, comme le JavaScript, évolues souvent plus vite que les navigateurs qui sont sensés les lire. C'est pouquoi il faut configurer votre environement de développement pour écrire du code ES6 et pouvoir l'exploiter sur le Web. Déffirents solutions éxistes, il s'agit de complilateurs de codes qui sont capable de trascrire le code pour qu'il soit interprété correctement, nous utiliserons dans nos exemple la solution proposée par le framework Babel.

CONFIGURATION DU DOSSIER DE TRAVAIL

Dans un premier temps, vous devez créer un dossier contenant deux sous-dossiers, **"typings"** et un **"js"**, ainsi qu'un fichier **"package.json"**.

```
MyES6proj
  - js
  - typings
  - package.json
```

INSTALLATION DE BABEL CLI

Ouvrez une fenêtre de terminal, rendez vous dans le dossier que vous venez de créer et taper la commande suivante :

```
npm install --save-dev babel-cli babel-preset-env
```

`--save` ..... référencie le module dans le fichier package.json
`-dev` ..... indique qu'il s'agit d'un modul de développement

Toutes les option de bable sont disponibles sur babeljs.io.

CONFIGURATION DU COMPILATEUR

Une fois Babel CLI installé, vous ajouter un script dans votre fichier package.json pour configurer Babel CLI.

```
..
  "scripts": {
    "build": "babel --no-babelrc typings -w -d js
    --presets=env"
  },
..
```

`-w` .... compliation du/des fichier/s à la vollée

COMPILATION DES FICHIERS

Tous les fichier ES6 du dossier **"typings"** seront compilés en ES5 dasn le dossier **"js"** en tapant la commande suivante dans votre terminal :

```
npm run build
```

LE TEMPLATING ES6 : LA BACKQUOTE

Tous lés développeur s'étant souvent retrouver à modifier le contenu du DOM s'est retrouvé devant la compléxité de la concatènation en JavaScript : l'enchainement de plus et de backslash à outrance demande une gymnastique complèxe. ES6 intègre le principe des apostrophes inversées - ou backquote - qui permet d'écrire des strings sur plusieurs lignes et d'y appeler des variables sans devoir couper la chaine.

LES VARIABLES LET

En ES6 pour définir une variable classique il ne faut plus utiliser var mais **let**

LE MULTILIGNE

Avec les backquote il est possible d'écrire sur plusieurs ligne sans couper la chaine

LE SYMBOLE DOLLAR

Les variables sont directement appelées ou manipulées sans couper la chaine

```
let name = `Abdel`
let age = 17

document.querySelector(`header`).innerHTML = `
<h1>Bonjour ${name}, pret à utiliser les backquotes ?</h1>
<p>Votre année de naissance est ${2017 - age}.</p>
`
```

VARIABLES NON MODIFIABLES

Nous venons de voir dans l'exemple précédent que le mot clès var a été remplacé par let, bien que les deux peuvent vivre ensemble dans le même fichier il est recommander d'utiliser let pour les variables classiques. ES6 intégre également la notion de variables non-modifiable et qui manquait à JavaScript : les constantes. Définir une constante permet de bloquer la valeur d'une variable pour qu'elle ne puisse pas être modifiée, c'est principalement une aide au développement car une fois la constante défini une erreur sera indiquée lors de la compilation.

LES CONSTANTES

Lorsqu'une constante est défini il n'est pas possible d'en modifier la valeur

```
const birthyear = 1997
birthyear = 1998

console.log(`Vous avez ${2017 - birthyear} ans`)
```

ERROR DE COMPILATION

Le compilateur bloc son exécution et indique l'endroit ou ce trouve l'erreur

```
SyntaxError: src/app.js: “birthyear” is read-only

   const birthyear = 1997
   birthyear = 1998
   ^
```

LES FONCTIONS FLÉCHÉES

Grâce à ES6, il est à présent possible d'écrire une fonction simple sur une seule ligne avec les fonctions fléchées, il suffi de créer une variable et d'écrire la fonction selon la syntaxe suivante : (attr) => result.

FONCTION FLÉCHÉE

Si une fonction n'a qu'un seul retour il est possible de l'écrire sur une ligne

```
const twice = (attr) => return attr * 2
twice(7) === 14
```

LES PARAMÊTRES OPTIONNELS

Une des grandes possibilités qu'ajoute ES6 au langage est la possibilité de définir des paramêtres optionnels directement lors de la création de la fonction.

VALEUR PAR DÉFAUT

En ajoutant une valeur à l'attribut de la fonction il devient optionnel

```
const twice = (attr = 7) => {
    return attr * 2
}
twice() === 14
twice(5) === 10
```

REST PARAMETER ET SPREAD OPERATOR

Ces deux nouveaux principes consistent à aggréger plusieurs valeurs en un seul paramêtre pour faciliter le traitement de ces valeurs. Cette technique consite à préfixer un paramêtre avec trois points et selon qu'il soit ajouter à une fonction ou une variable, les valeurs seront intégrées dans un tableau itérable ou concaténées avec la valeur de la variable qui les appel.

FUNCTION : REST PARAMETER

Dans le cadre d'une fonction lorsque les trois points sont ajoutés avant un paramêtre, la function comprendra qu'elle doit ajouter tous les paramêtres dans un tableau.

PARAMÊTRE REST

En ajoutant une valeur à l'attribut de la fonction il devient optionnel

```
const restFunction = (x, y, ...rest) => return (x + y) + rest.length
restFunction(3, 2, true, 14, `Sophia`, ) === 8
```

VARIABLE : SPREAD OPERATOR

Dans le cadre d'une fonction lorsque les trois points sont ajoutés avant un paramêtre, la function comprendra qu'elle doit ajouter tous les paramêtres dans un tableau.

OPÉRATEUR SPREAD

Les valeur sont ajoutées dans le tableau

```
let spread = [true, 14]
let myArray = [`Sophia`, ...spread]
console.log(myArray) // [`Sophia`, true, 14]
```