ES6/7-CHEAT-SHEET

Jean-François Le Foll (@JeffLeFoll)

Assignation de variable

```
> const pi = 3.14;
> pi = 3.15
x TypeError: invalid assignment to const pi

> let foo = 'bar';
> console.log(foo)
< "bar"
> foo = 'foo'
> console.log(foo)
< "foo"
> let foo = 'toto'
x SyntaxError: redeclaration of let foo
```

Manipulation des tableaux (https://repl.it/@JeffLeFoll/Array5)

```
> ['foo', 'bar', 'flop'].filter(item => item.startsWith('f'));
< [ "foo", "flop" ]

> [1, 2, 3].reduce((total, valeur) => total + valeur);
< 6

> ['toto', 'pop', 'start'].map(item => item.split('').reverse().join(''));
> [ "otot", "pop", "trats" ]

> ['foo', 'polf', 'flop'].map(item => item.split('').reverse().join(''))
.filter(item => item.startsWith('f'));
< ['flop']
```

Fonction fléchée (https://repl.it/@JeffLeFoll/FonctionFlechee)

```
> let addition = (x, y) => x + y;
> addition(4,5);
< 9

Equivaut à :

> let addition = (x, y) => {
    return x + y;
};

> let addition = function(x,y) {
    return x + y;
}
```

Classe (https://repl.it/@JeffLeFoll/ES6class)

```
> class Vehicule {
    constructor(couleur, puissance) {
        this._couleur = couleur;
        this._puissance = puissance;
    }

    get puissance() {return this._puissance}

    get couleur() {return this._couleur}

    set couleur(nouvelleCouleur) {this._couleur = nouvelleCouleur}
}

> let unVehicule = new Vehicule('rouge', '7cv');
> console.log(unVehicule.couleur);
< 'rouge'
> unVehicule.couleur = 'vert';
> console.log(unVehicule.couleur);
< 'vert'
> console.log(unVehicule.puissance);
< '7cv'
> unVehicule.puissance = '4cv';
> console.log(unVehicule.puissance);
< '7cv'
```

Gestion des paramètres (https://repl.it/@JeffLeFoll/Parametres)

```
> let additionAvecMinimum1 = (x, y=1) => x + y;
> additionAvecMinimum1(4);
< 5

> let addition = (...valeurs) => valeurs.reduce((total, valeur) => total + valeur );
> addition(5,6,7);
< 18

> let soustraction = (...[a, b, c]) => a - b - c;
> soustraction(20, 5, 2, 45);
< 13
```

Classe - Extension (https://repl.it/@JeffLeFoll/ES6ClassExt)

```
> class Moto extends Vehicule {
    constructor(couleur, puissance, type) {
        super(couleur, puissance);
        this._type = type;
    }

    debridage(nouvellePuissance) {this._puissance = nouvellePuissance}

    static warning() {return 'N\'oubliez pas les équipements de sécurités'}
}

> let gsr600 = new Moto('gris', '98cv', 'roadster');
> console.log(gsr600.couleur);
< 'gris'
> console.log(gsr600.puissance);
< '98cv'
> gsr600.debridage('110cv');
> console.log(gsr600.puissance);
< '110cv'
> Moto.warning();
< "N'oubliez pas les équipements de sécurités"
```

Scoped Function (https://repl.it/@JeffLeFoll/ScopedFunction)

```
> (function() {
  let texte = 'Je suis une fonction auto-appelante !';
  console.log(texte);
})();
< Je suis une fonction auto-appelante !

Devient :

> {
  let texte = 'Moi pareil mais avec une syntaxe plus simple ! :)';
  console.log(texte);
}
< Moi pareil mais avec une syntaxe plus simple ! :)
```

Template Literals (https://repl.it/@JeffLeFoll/TemplateLiterals)

```
> let personne = {prenom:'James', nom:'Bond'};
> let exclam = 'AH AH !';

> console.log(`My name is ${persone.nom},
            ${personne.prenom}
      ${personne.nom}
-- ${exclam} --`);
< My name is Bond,
            James
      Bond
-- Ah Ah ! --
```

Destructuration (https://repl.it/@JeffLeFoll/Destructuration)

```
> let tableauSource = [1, 2, 3, 4];
> let dest1, dest2, reste;

> [dest1, dest2, ...reste] = tableauSource;
> console.log(dest1); // 1
> console.log(dest2); // 2
> console.log(reste); // [3, 4]

> let personne = {nom: 'Bond', prenom: 'James'};
> let {nom, prenom} = personne;
> console.log(nom); // Bond
> console.log(prenom); // James

> let url = 'https://developer.mozilla.org/en-US/Web/JavaScript';
> let parsedURL = /^(\w+)\:\/\/([^\/]+)\/(.*)$/.exec(url);
> console.log(parsedURL);
< ['https://developer.mozilla.org/en-US/Web/JavaScript', 'https',
< 'developer.mozilla.org','en-US/Web/JavaScript']

> let [source, protocol, fullhost, fullpath] = parsedURL;
> console.log(protocol); // https
```

Promesses (https://repl.it/@JeffLeFoll/Promesse)

```
> let promesse = new Promise((resolve, reject) => {
  setTimeout(() => resolve('Success!'), 250);
});

> promesse.then(message => console.log(message));
< Success!

> promesse
  .then(message => message + ' bingo')
  .then(messageModifie => console.log(messageModifie))
  .catch(reason => console.log('Error : ' + reason));
< Success! bingo
```

Fetch API (https://repl.it/@JeffLeFoll/HttpRequest)

```
> fetch('https://swapi.co/api/starships/10/')
  .then(reponse => {
    if (reponse.ok) {
      return reponse.json();
    }
    throw new Error('Network response was not ok.');
  })
  .then(data => console.log('fetch: ' + data.name))
  .catch(error => console.log('Problem : ' + error));
< fetch: Millennium Falcon

> let options = {
  method: 'GET',
  headers: new Headers(),
  mode: 'cors',
  cache: 'default',
};
> fetch('https://swapi.co/api/starships/10/', options);
```

HttpRequest (https://repl.it/@JeffLeFoll/HttpRequest)

```
> let xhr = new XMLHttpRequest();
> xhr.open('get', 'https://swapi.co/api/starships/10/', true);
> xhr.responseType = 'json';
> xhr.onload = function() {
  let status = xhr.status;
  if (status == 200) {
    console.log('xhr: ' + xhr.response.name);
  } else {
    console.log('Network response was not ok.');
  }
};
> xhr.send();
< xhr: Millennium Falcon
```

Async / Await (https://repl.it/@JeffLeFoll/AsyncAwait)

```
> async function loadData(url) {
    let response = await fetch(url);
    let data = await response.json();
    return data;
  }

> let data = await loadData('https://swapi.co/api/starships/10/');
> console.log(data.name);
< Millennium Falcon
```

JSON to ES Class (https://repl.it/@JeffLeFoll/JSON2Class)

```
> let jsonData = { nom: 'Bond', prenom: 'James' };

> jsonData.presentation = () => jsonData.nom + ', ' + jsonData.prenom;
> console.log(jsonData.presentation());
< Bond, James

> let autreJsonData = { nom: 'Trevelyan', prenom: 'Alec' };
> console.log(autreJsonData.presentation());
< TypeError: autreJsonData.presentation is not a function

> class Personne {
  constructor({ nom, prenom }) {
    this.nom = nom;
    this.prenom = prenom;
  }

  presentation() {
    return this.nom + ', ' + this.prenom;
  }
}
```

JavaScript Module (http://jeff.lefoll.info/es6-cheat-sheet/exemples/imports/index.html)

```
 -- A tester avec Chrome ou Firefox (activer le flag :
 -- dom.moduleScripts.enabled dans la page about:config )
 -- Logger.js
 export class Logger {
  static log(logMessage) {
    console.log('From Logger : ' + logMessage);
  }
}
-- Main.js
import { Logger } from './Logger.js';

export class Main {
  constructor(message) {
    Logger.log(message);
  }
}
-- index.html
<html><body>
    <script type="module">
        import { Main } from './Main.js';
        let main = new main('Bingo !!');
    </script>
</body></html>

< From Logger : Bingo !!
```

Test unitaire (http://jeff.lefoll.info/es6-cheat-sheet/exemples/tests/index.html)

```
-- Agent.js
class Agent {
  constructor(nom, prenom, code) {
    this.nom = nom;
    this.prenom = prenom;
    this._code = code;
  }

  presentation() {
    return this.nom + ', ' + this.prenom;
  }

  get code() {
    return this._code;
  }
}

-- AgentSpec.js
chai.should(); // ou let expect = chai.expect;

describe("Le comportement d'un Agent est", () => {
```

Prototype (https://repl.it/@JeffLeFoll/Prototype)

```
> function Moto(couleur) {this.couleur = couleur;}

> let motoRouge = new Moto('rouge');
> console.log(motoRouge.type); // undefined

> Moto.prototype.type = 'sportive';

> let motoBleu = new Moto('bleu');
> motoBleu.type = 'roadster';

> console.log(motoRouge.type); // "sportive"
> console.log(motoBleu.type); // "roadster"

> motoBleu.carburan = 'essence';
> console.log(motoRouge.carburan); // "undefined"
> console.log(motoBleu.carburan); // "essence"
```

Dans le doute

https://developer.mozilla.org
https://caniuse.com/
https://github.com/airbnb/javascript

```
> let agent006 = new Personne(autreJsonData);
> console.log(agent006.presentation());
< Trevelyan, Alec
```

```
  let monAgent;

  beforeEach(() => {
    monAgent = new Agent('Bond', 'James', '007');
  });

  it('doit se presenter', () => {
    monAgent.presentation().should.equal('Bond, James');
  });
});
```