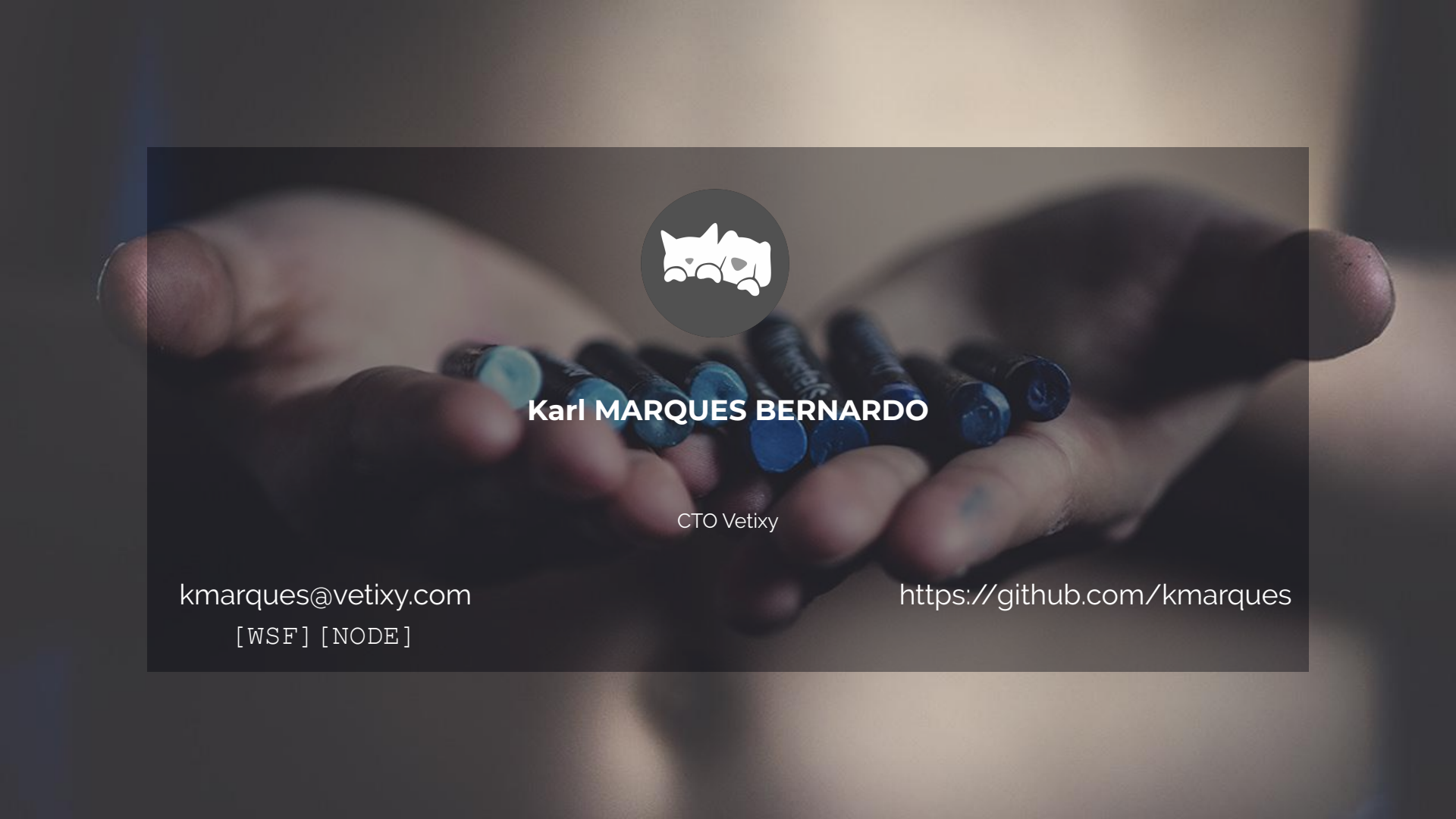

NodeJS



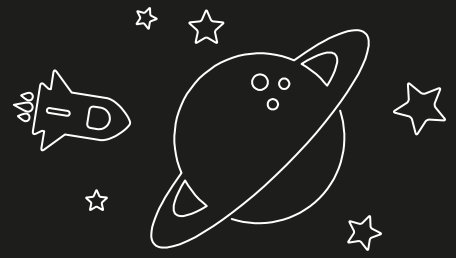
Karl MARQUES BERNARDO

CTO Vetixy

kmarques@vetixy.com

https://github.com/kmarques

[WSF][NODE]

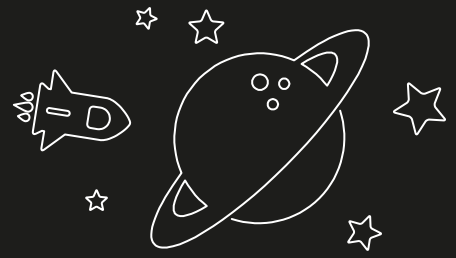


EVALUATION

3

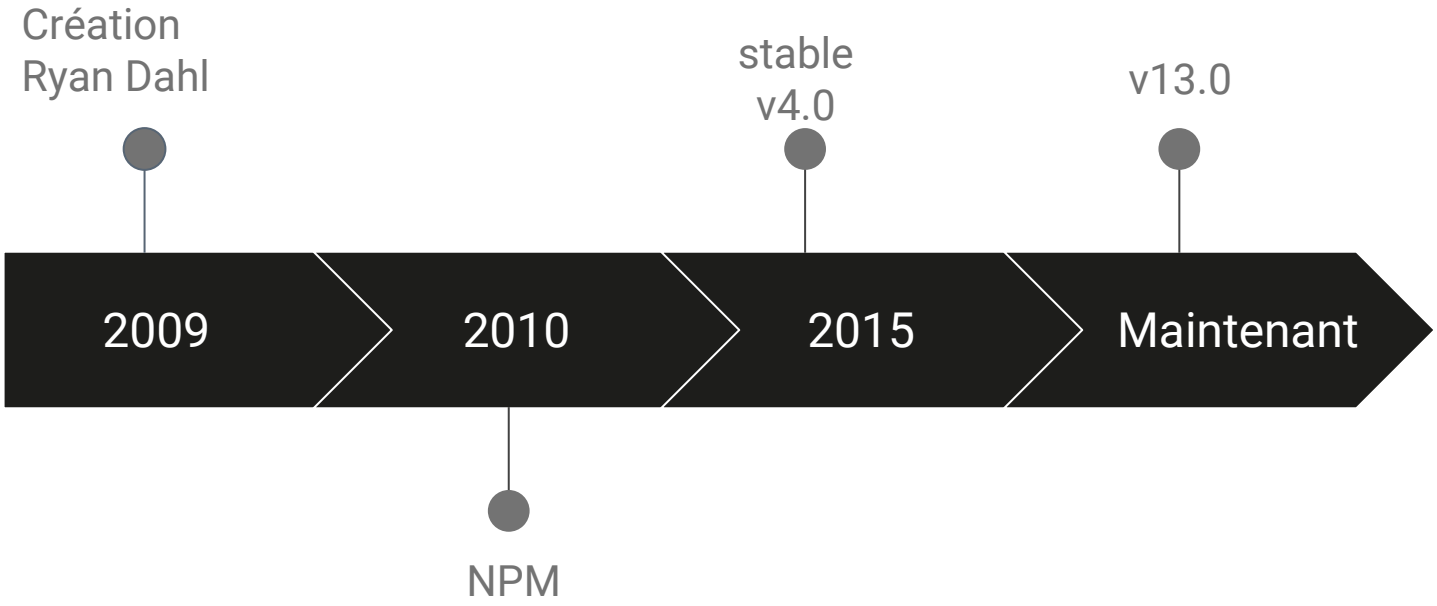
Evaluation finale

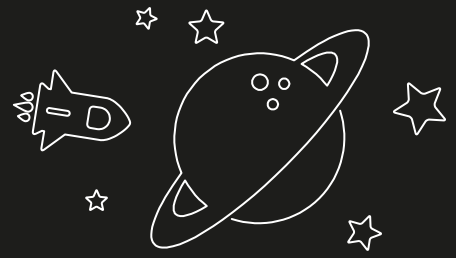
Projet de groupe



HISTORIQUE

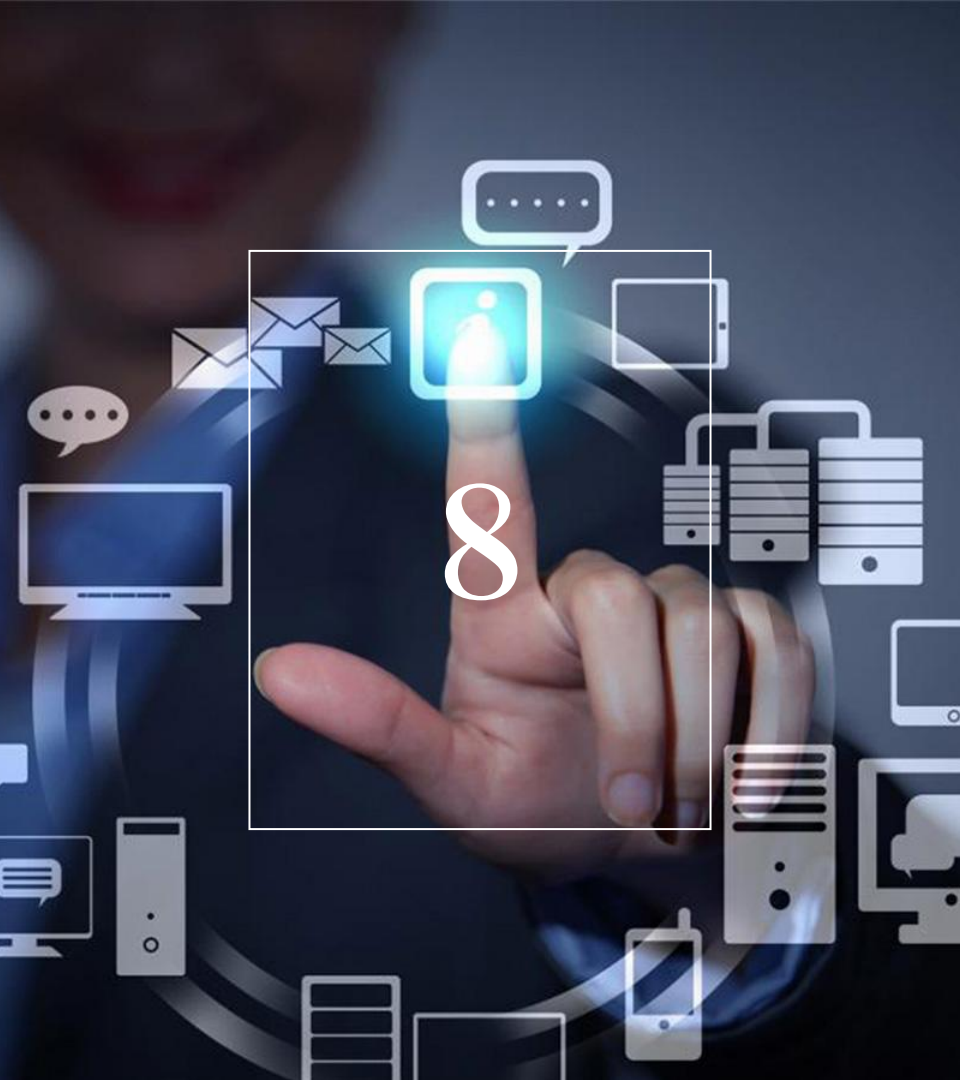
5


Création
Ryan Dahl
stable
v4.0
v13.0
2009
2010
2015
Maintenant
NPM



ECO-SYSTEME

7



Où exécuter son script ?

Terminal client/serveur

Linux: Bash, Sh, zsh

Windows: WSL, Batch, PowerShell

Base de données

MongoDB

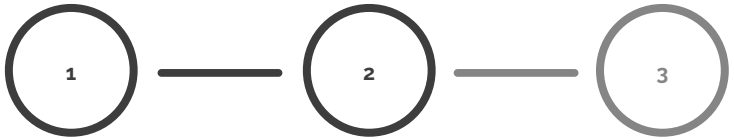
Comment exécuter de l'ES6 ou ultérieur ?

LES TRANSPILLERS ou "compilateurs"

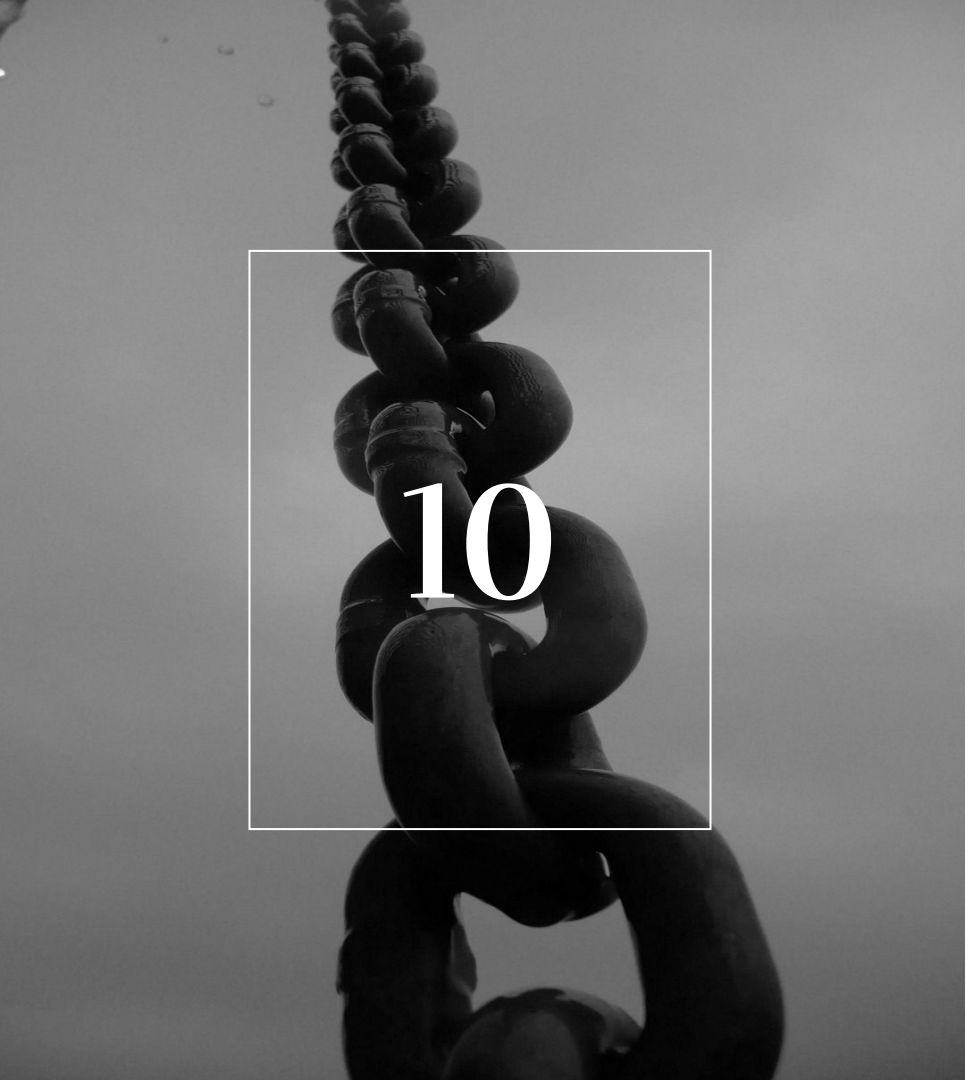
(Babel / Browserify / ...)

Transformer du code Javascript

Next-Gen en javascript multi-plateformes



Parser	Transformers	Generator
Code -> AST	AST -> AST	AST -> Code
	(plugins/presets)	



Les toolchains ou task-runners

Même concept que pour les TRANSPILLERS mais appliqué à l'ensemble des assets d'une application

(Gulp / Grunt / WebPack)

1 — 2 — 3

Inputs

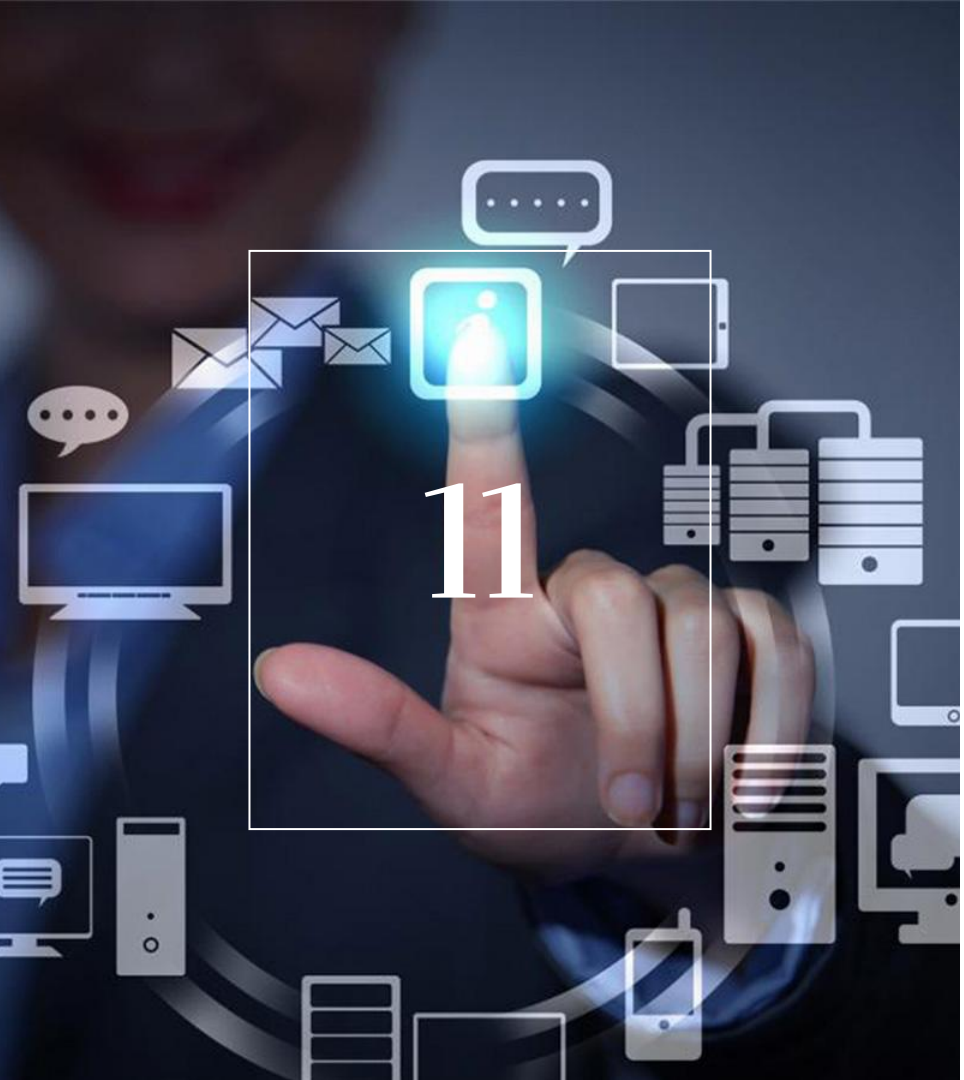
Collecte des entrées
fichiers
var Env
string

Transformers

Modification des entrées
plugins

Generator

Sorties



Les gestionnaires de paquets

NPM

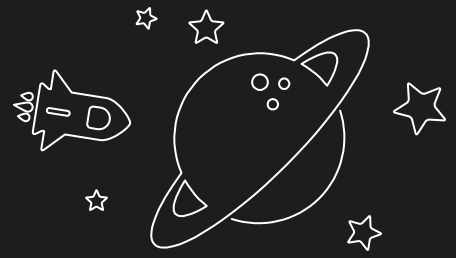
Gestion automatique des dépendances

YARN

Mode Hors-ligne
Meilleur gestion des dépendances
Plus sécurisé

BOWER

Gestion manuelle des dépendances
Plateforme annexe de paquets
Mort et enterré



INSTALLATION

12

1. **Docker : docker-compose.yml**

```
version: '3'
services:
  node:
    image: 'node:13-alpine'
    volumes:
      - './:/home/node/app'
    working_dir: '/home/node/app'
    ports:
      - '3000:3000'
    command: 'npm install && npm start'
```

2. **Mac : Homebrew**

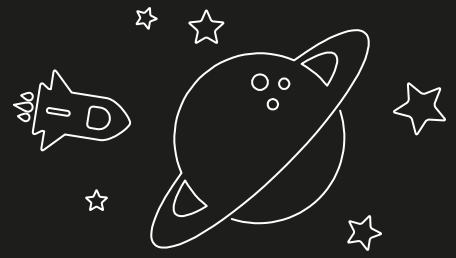
   brew install nodejs

3. **Linux**

   curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
   sudo apt-get install -y nodejs

4. **Windows : Télécharger l'installeur**

   WSL : utiliser la méthode Linux
   Autres:
   https://nodejs.org/en/download/
   Pensez à ajouter au PATH !!!



Rappel des bases

14

Les types de données

Types primitifs

String => “message”

Number => 2 2.3

Boolean => true false

Undefined => undefined

Types complexes

Object

object => { prop1: true, prop2: “yes” }

null => null

array => [1, 2, 3]

Function

Portées des variables

foo = 3

Déclaration globale

var foo = 3

Déclaration locale

let foo = 3

Déclaration au block

const foo = 3

Let + constante de référence

Attention !!! Phénomène de Hoisting

Javascript remonte automatiquement les déclarations de variables sans les initialiser en haut du scope courant pour le keyword **var**

Exemples

```
var foo = 1;
(function() {
  console.log(foo);
  var foo = 2;
  var baz = 3;
  bar = 4;
})();

console.log(foo);
console.log(bar);
console.log(baz);
```

Undefined

1

4

erreur

Les closures

Fonction dans une fonction

Javascript se souvient de l'environnement de chaque fonction.

Une variable qui ne devrait plus exister peut donc continuer à être appelée.

```
function creerFonction() {
  var nom = "Mozilla";
  function afficheNom() {
    console.log(nom);
  }
  return afficheNom;
}

var maFonction = creerFonction();
maFonction();
```

Démonstration

capitalize
1ère lettre de chaque mot en MAJ
hello world => Hello World

camelCase
Capitalize + coller les mots
hello world => HelloWorld

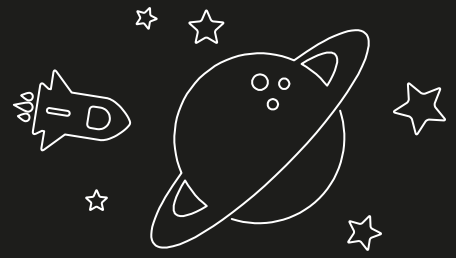
Exercices

snake_case
Joindre les mots par des underscores en MIN
Hello World, i am john => hello-word-i-am-john

leet - Cryptage (uniquement les voyelles)
anaconda => 4n4c0nd4
A=>4, E=>3, I=>1, O=>0 ,U=> (_), Y=>7

yoda
Inverser la position des mots d'une phrase
Hello world => world Hello

Renvoyer une chaîne vide en cas de valeur non String



LES OBJETS

20

- Tout type est “wrappé” par un objet
- Tout objet contient un propriété ***prototype***
- ***prototype*** contient l’ensemble des méthodes et propriétés accessibles de l’objet hérité
- Héritage

var Object2 = Object.create(Object1);

Déclaration d'un objet

- Accolade (A)

```
var myObject = {
    p1: "foo"
};
```

- Constructeur (C)

```
function MyObject() {
    this.p1 = "foo"
}
var myObject = new MyObject()
```

Publique

Propriétés

- **(A)**

```
{
  p1: “foo”
};
```

- **(C)**

```
function MyObject() {
  this.p1 = “foo”;
}
```

Méthodes

- **(A)**

```
{
  m1: function() {}
};
```

- **(C)**

```
function MyObject() {
  this.m1 =function() {};
}
```

Prototype

```
myObject.prototype.p1 = “foo”;
myObject.prototype.m1 = function() {};
```

Privé

Propriétés

- **(A)**

Impossible

- **(C)**

```
function MyObject() {
  var pr1 = “foo”;
}
```

Méthodes

- **(A)**

Impossible

- **(C)**

```
function MyObject() {
  var mr1 =function() {};
}
```

Les classes

```
class Student extends Person {
        constructor (name, prom) {
                super(name);
                This.prom = prom;
        }
        monAttr = 4;
        display () {
                return
                        super.display()
                        + " de la promo " + this.prom;
        }
}
Student.prototype.say = function() {
   "Bonjour, mon nom est "+this.name
};
```

```
const toto = new Student("toto", 2019);

console.log(toto.monAttr)   => 4

console.log(toto.display()) => "Je suis toto de la promo 2019"

toto.say() => "Bonjour, mon nom est toto"
```

- Reprendre l'exercice 1, rendre toutes les fonctions accessibles pour chaque String

  Exemples:

  ucfirst("ma chaine") => "ma chaine".ucfirst()

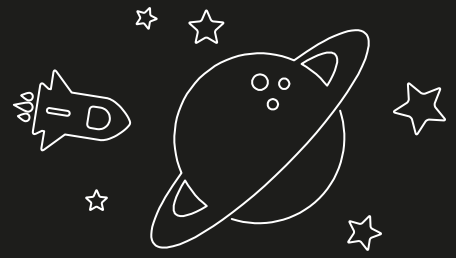
  vig("ma chaine", "ma clé") => "ma chaine".vig("ma clé")

- Ecrire la fonction round pour le type Number, celle-ci utilise un argument facultatif afin de savoir à quelle précision le nombre doit être arrondi, par défaut 0.

  Exemples:

  12.45678.round(2) => 12.46

  12.45678.round() => 12



Les Exceptions

27

- 6 types d'exceptions

EvalError : erreur dans un *eval()*

RangeError : utilisation d'un nombre en dehors des valeurs possibles

ReferenceError : utilisation d'une variable non déclarée ou hors scope

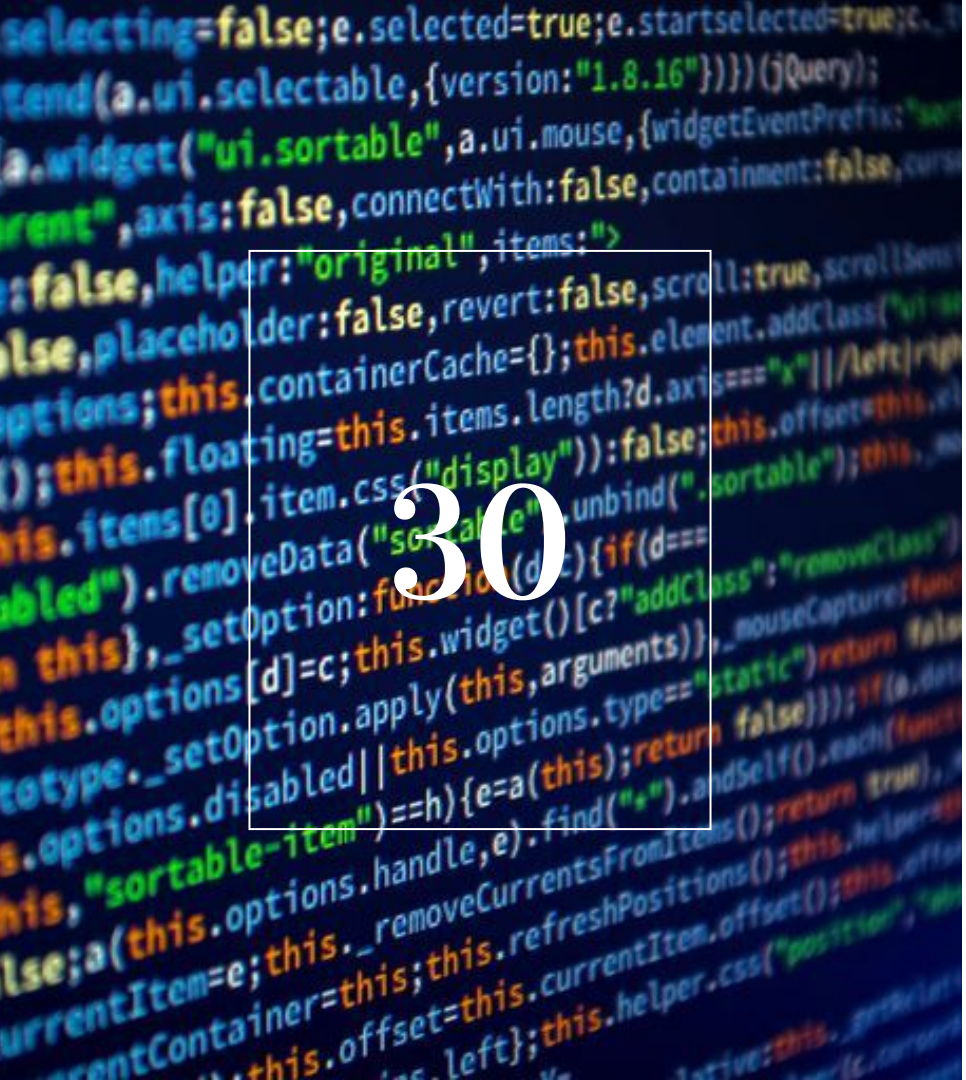
SyntaxError : erreur de syntaxe dans le code soumis à un *eval()*

TypeError : utilisation d'une fonction n'appartenant pas au type

URIError : utilisation de caractères illégaux dans une fonction URI

- Possibilité de créer ses propres exceptions

```
function MyError(param1, param2, ...) {
   var instance = new Error("custom message");
   Object.setPrototypeOf(
          instance, Object.getPrototypeOf(this)
   );
   If (Error.captureStackTrace) {
     Error.captureStackTrace(instance, MyError);
   }
   return instance;
}
```



Gestion des exceptions

try {

Permet d'exécuter des instructions à risque

throw *"exc" | new Error("exc");*

} catch (error) {

Catch les exceptions levées

If (error instanceOf MyError)

ou

If (error.name === "MyError")

} finally {

Permet d'exécuter des instructions même si des exceptions sont levées

}

Exercice

- Créer une exception **TooFewArgumentError**(funcName, expectedCount, gotCount)

  Message: “Too few arguments for function ‘{funcName}’ expected ‘{expectedCount}’ got ‘{gotCount}’”

  EX: throw new TooFewArgumentError(“myfunc”, 3, 2)

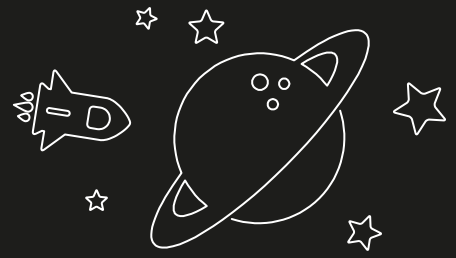
  => “Too few arguments for function ‘myfunc’ expected ‘3’ got ‘2’”

- Créer une fonction **test** qui catch l’exception et affiche

  Si exception, **“Exception caught”**

  Sinon, **“Alright!”**

Commit : [DONE] exercice 5



LES MODULES

32

Les modules

- Permet de générer des libs JS
- Les variables sont scopées aux modules, hors variables globales
- Système d'export/import
- <script type="module" src="./main.js"></script>

Exemple

Library.js

Méthode multi-export

```
export const myVar = 10;
export function myFunc() {};
export default function myDefault() {};
```

Méthode single-export (CommonJS)

```
const myVar = 10;
function myFunc() {};

module.exports = {
        myVar: myVar,
        func1: myFunc
};
```

Main.js

Méthode multi-export

```
import {myVar, myFunc, default as func2} from "./library.js";
```
=> importe toutes les fonctions nommées de la lib

```
import func2 from "./library.js";
```
=> importe la fonction par défaut de la lib

```
import * as lib from "./library.js";
```
=> importe tous les exports dans un objet lib

Méthode single-export (CommonJS)

```
const func1 = require('./library.js').func1;
const var1 = require('./library.js').myVar;
```