

REPL

NodeJS

REPL

NodeJs vient avec un environnement virtuel appelé

Read

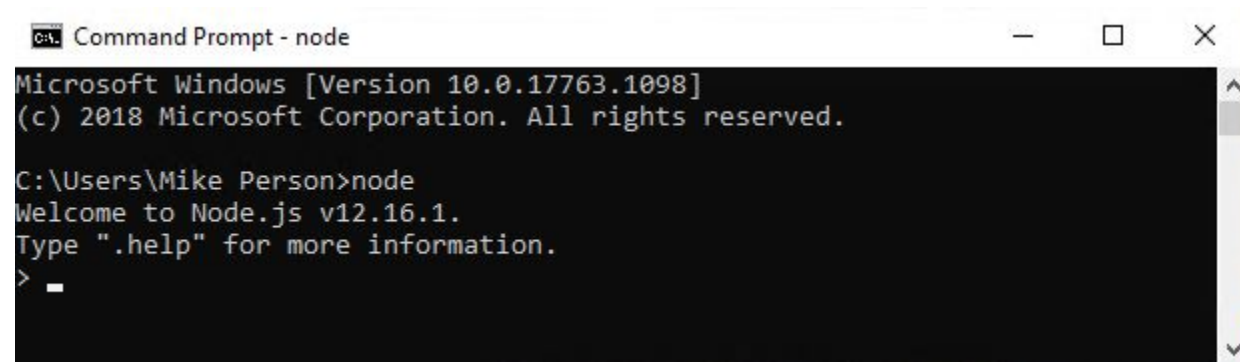
Eval

Print

Loop.

C'est la façon la plus simple et rapide de tester du NodeJS.

Pour lancer l'environnement, il suffit de se rendre dans une console (command prompt sous Windows ou Terminal sous Linux) et ensuite taper node.



```
C:\> Command Prompt - node
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

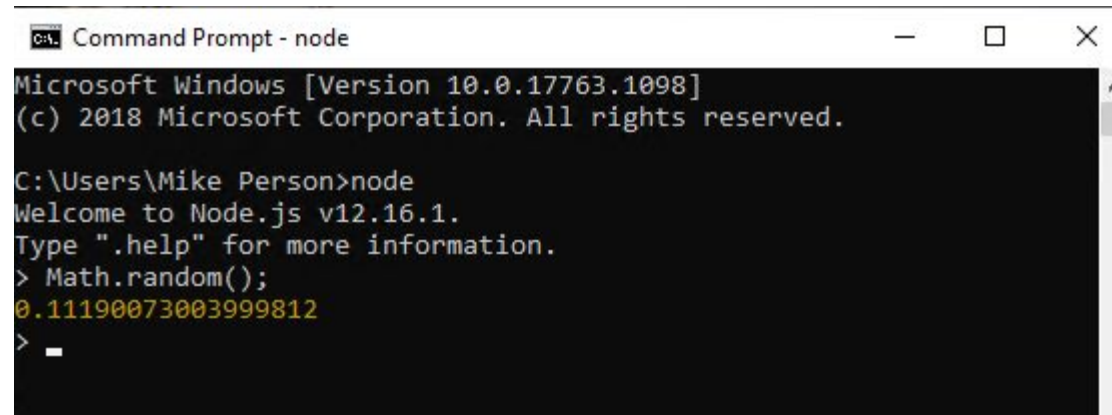
C:\Users\Mike Person>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> _
```

REPL

Commandes	Description
.help	Affiche la page d'aide des commandes
tab Keys	Affiche la liste des commandes
Up/Down Keys	Permet de remonter à la commande précédente ou aller vers la suivante
.save filename	Sauvegarde la session actuelle dans un fichier
.load filename	Charge un fichier contenant une sessions REPL
ctrl + c	Permet de terminer la session
ctrl + c (twice)	Permet de quitter l'environnement
ctrl + d	Permet de quitter l'environnement
.break	Permet de sortir d'une expression multi-ligne
.clear	Permet de sortir d'une expression multi-ligne

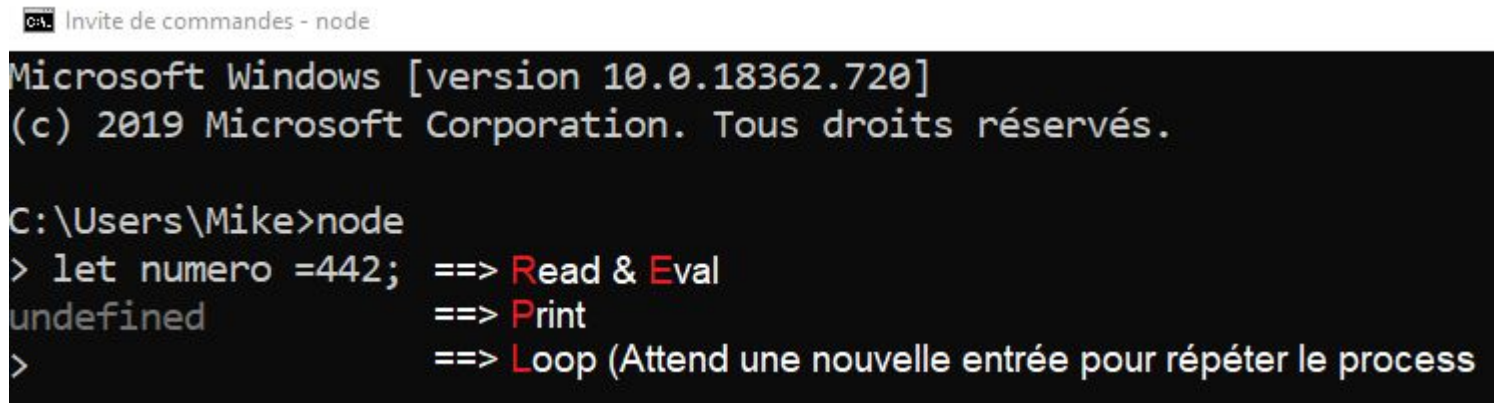
REPL

Grâce à cet outils, vous pouvez écrire du javascript et tester facilement celui-ci.



```
Command Prompt - node
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Mike Person>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> Math.random();
0.11190073003999812
> _
```

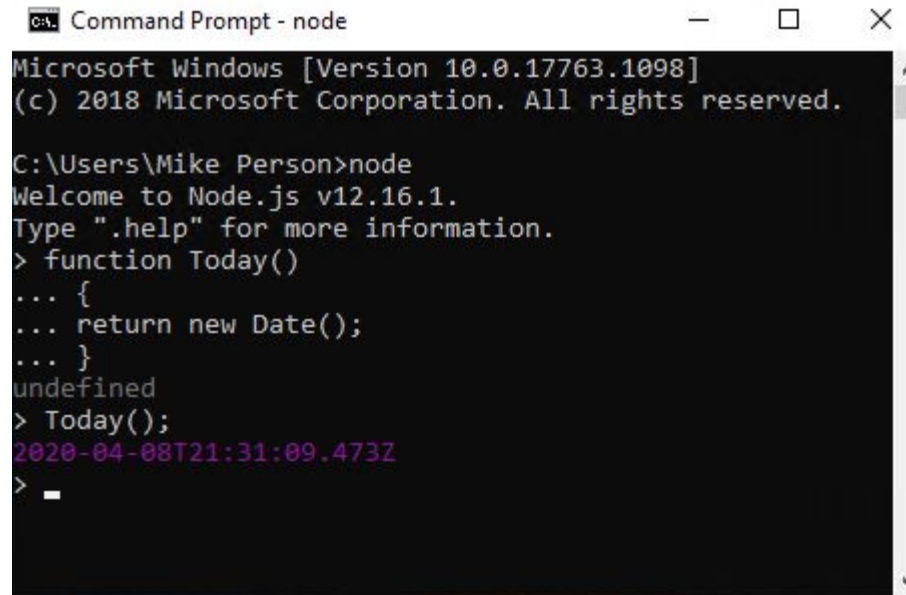


```
Invite de commandes - node
Microsoft Windows [version 10.0.18362.720]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\Mike>node
> let numero =442;
undefined
==> Read & Eval
==> Print
==> Loop (Attend une nouvelle entrée pour répéter le process)
```

REPL

L'outil est capable de détecter si nous sommes dans une instruction multilignes



```
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Mike Person>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> function Today()
... {
...   return new Date();
... }
undefined
> Today();
2020-04-08T21:31:09.473Z
> _
```

REPL

Le mode *editor* permet d'écrire de multiples instructions pour ensuite pouvoir les utiliser.

La commande `.editor` ouvre un éditeur et on utilise CTRL+D pour le fermer.

```
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Mike Person>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> .editor
// Entering editor mode (^D to finish, ^C to cancel)
function Add(x,y)
{
    return x+y;
}

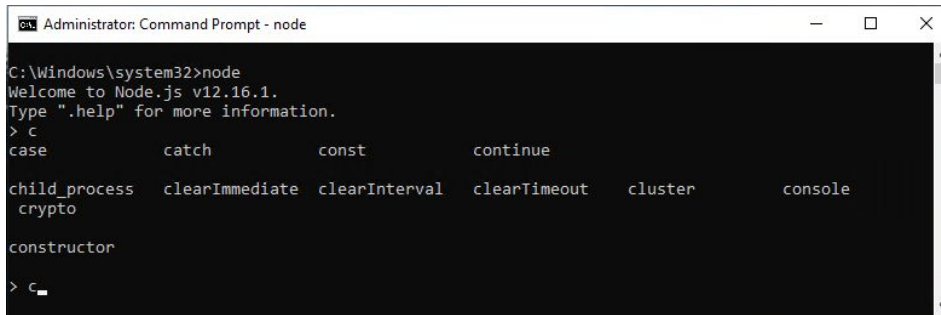
function Today()
{
    return new Date();
}

undefined
> Add(2,4)
6
> Today()
2020-04-08T21:34:33.106Z
> _
```

REPL

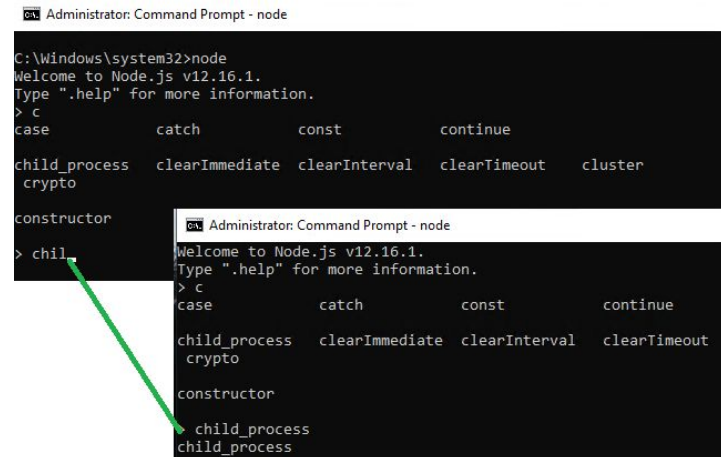
La touche Tabulation : la touche magique.

La touche Tab dans l'outil permet de proposer des instructions (2 press)



```
Administrator: Command Prompt - node
C:\Windows\system32>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> c
case          catch          const          continue
child_process clearImmediate clearInterval clearTimeout cluster    console
crypto
constructor
> c_
```

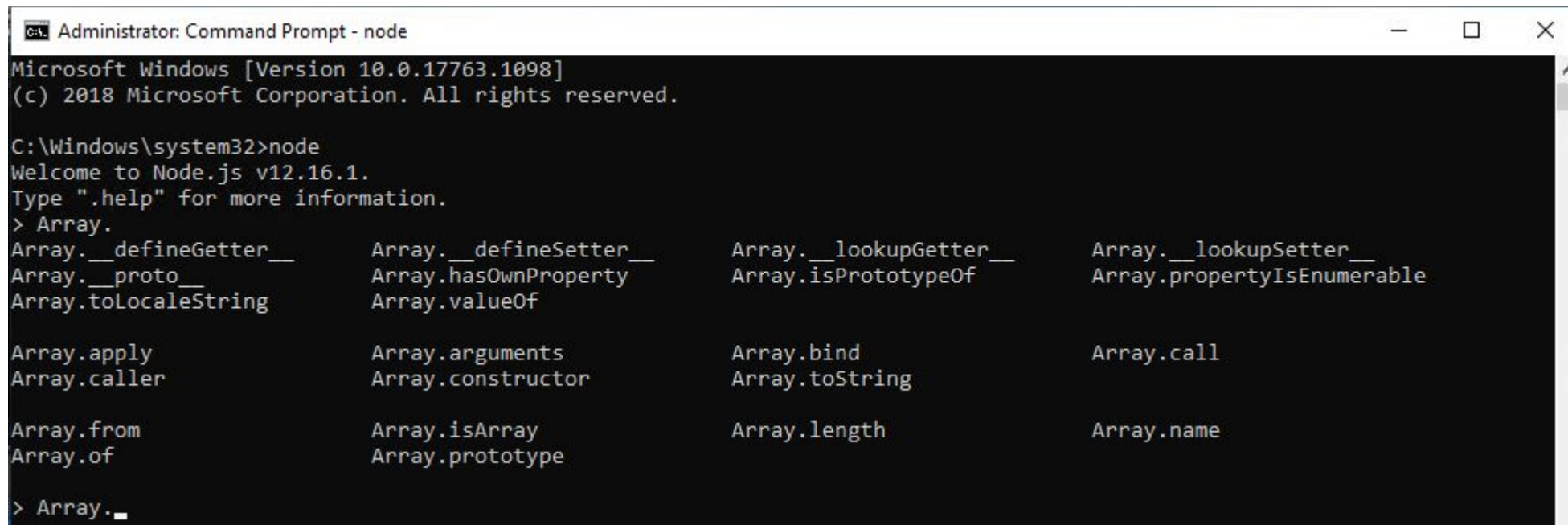
ou de compléter une instruction (1 press)



```
Administrator: Command Prompt - node
C:\Windows\system32>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> c
case          catch          const          continue
child_process clearImmediate clearInterval clearTimeout cluster    co
crypto
constructor
> chil_
child_process
child_process
```

REPL

Ce principe de tabulation fonctionne également pour avoir un « intellisense » à partir d'une classe



```
Administrator: Command Prompt - node
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> Array.
Array.__defineGetter__      Array.__defineSetter__    Array.__lookupGetter__    Array.__lookupSetter__
Array.__proto__            Array.hasOwnProperty      Array.isPrototypeOf       Array.propertyIsEnumerable
Array.toLocaleString       Array.valueOf
Array.apply                Array.arguments           Array.bind                Array.call
Array.caller              Array.constructor        Array.toString
Array.from                Array.isArray            Array.length              Array.name
Array.of                  Array.prototype
> Array._
```


REPL

Les Timers

Les fonctions existantes en JS se retrouvent bien entendu en NodeJs.

- Si on désire retarder l'exécution d'une fonction : **setTimeout(référence de la fonction à exécuter, temps en ms[, *arg1,arg2,arg3,...*])**

Exemple :

```
Administrator: Command Prompt - node
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> const func = audience =>{ console.log(audience + ' is listening');}
undefined
> setTimeout(func,4*1000,'WebApps');
Timeout {
  _idleTimeout: 4000,
  _idlePrev: [TimersList],
  _idleNext: [TimersList],
  _idleStart: 90525,
  _onTimeout: [Function: func],
  _timerArgs: [Array],
  _repeat: null,
  _destroyed: false,
  [Symbol(refed)]: true,
  [Symbol(asyncId)]: 270,
  [Symbol(triggerId)]: 5
}
> WebApps is listening
```

REPL

- Ou si on désire répéter une instruction dans le temps : **setInterval**

```
C:\Windows\system32>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> const func = audience =>{ console.log('Coucou - ' + new Date().toString());}
undefined
> setInterval(func, 3*1000);
Timeout {
  _idleTimeout: 3000,
  _idlePrev: [TimersList],
  _idleNext: [TimersList],
  _idleStart: 27550,
  _onTimeout: [Function: func],
  _timerArgs: undefined,
  _repeat: 3000,
  _destroyed: false,
  [Symbol(refed)]: true,
  [Symbol(asyncId)]: 87,
  [Symbol(triggerId)]: 5
}
> Coucou - 10:48:44 GMT+0200 (Central European Summer Time)
Coucou - 10:48:47 GMT+0200 (Central European Summer Time)
Coucou - 10:48:50 GMT+0200 (Central European Summer Time)
Coucou - 10:48:53 GMT+0200 (Central European Summer Time)
```

REPL

Il existe également les fonctions suivantes :

- **setImmediate** : équivalent à l'appel du setTimeout avec un délai de 0
- **clearTimeout** : permet d'annuler l'appel au setTimeout dont l'id est passé en paramètre
- **clearInterval** : permet d'annuler l'appel au setInterval dont l'id est passé en paramètre
- **clearImmediate** : permet d'annuler l'appel au setImmediate dont l'id est passé en paramètre

```
const timerId = setTimeout(  
  () => console.log('You will not see this one!'),  
  0  
);  
  
// setImmediate  
  
clearTimeout(timerId);  
// clearInterval  
// clearImmediate
```

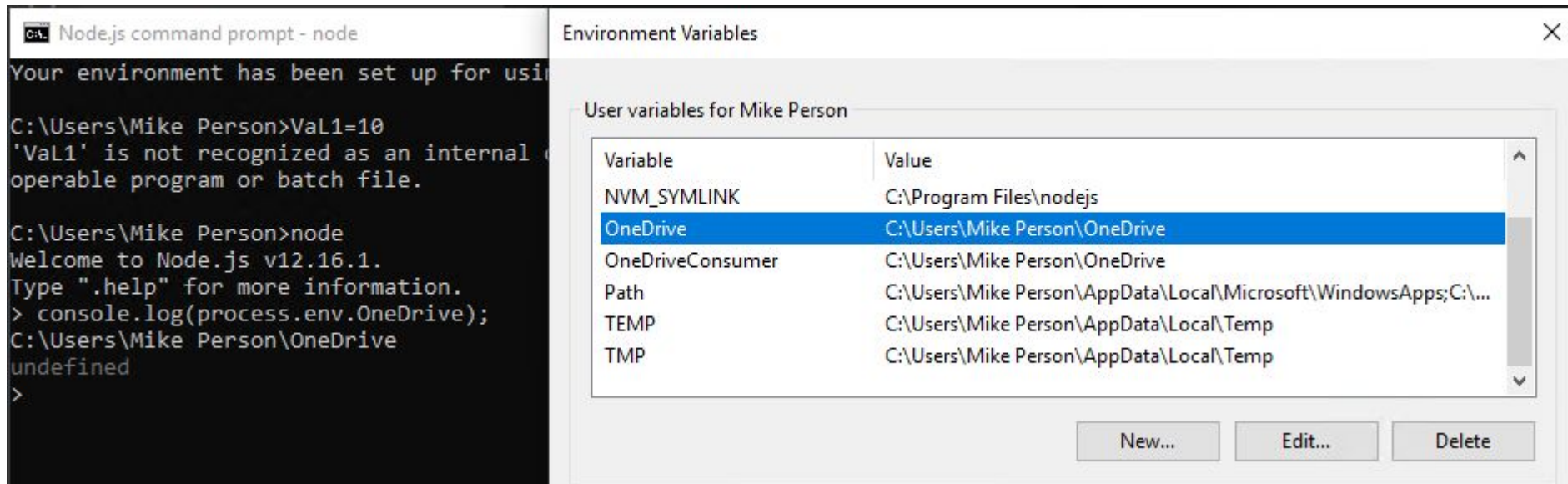
REPL

L'objet Process

Avec node, vous pouvez utiliser dans vos scripts les variables d'environnements définies en dehors du script.

Cela se fait via l'objet process et sa propriété env.

Exemple :



REPL

L'objet process contient également un attribut stdin qui nous permet de récupérer au clavier les infos encodées par l'utilisateur et stdout qui fait l'affichage dans la console

Exemple :

```
1 process.stdin.on('readable',()=>
2 {
3   const info = process.stdin.read();
4   if(info!==null)
5   {
6     console.log("via le log : " + info);
7     //ou
8     process.stdout.write("via process :" + info);
9   }
10  });
```

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL 1: powershell v

```
PS C:\Cours\NodeJs\Exemples\REPL\ProcessObject> node .\app.js
Test
via le log : Test

via process :Test
PS C:\Cours\NodeJs\Exemples\REPL\ProcessObject> |
```