

Analyse syntaxique avec Tree-sitter

15 nov. 2023

Analyse syntaxique

```
#test
require ["imapflags","fileinto"];

if anyof (
    header :contains "subject" "test"
) {
    stop;
    addflag "\\Seen";
    fileinto "INBOX";
}
```

script sieve

Analyse syntaxique

```
#test
require ["imapflags","fileinto"];

if anyof (
  header :contains "subject" "test"
) {
  stop;
  addflag "\\Seen";
  fileinto "INBOX";
}
```

script sieve

```
{
  actions: [
    { method: "stop" },
    { argument: "seen", method: "addflag" },
    { argument: "INBOX", method: "fileinto" }
  ],
  name: "test",
  rules: [
    {
      field: "subject",
      operator: "contains",
      value: "test"
    }
  ]
}
```

filtre sogo

Analyse syntaxique

```
#test
require ["imapflags","fileinto"];

if anyof (
  header :contains "subject" "test"
) {
  stop;
  addflag "\\Seen";
  fileinto "INBOX";
}
```

script sieve

```
{
  actions: [
    { method: "stop" },
    { argument: "seen", method: "addflag" },
    { argument: "INBOX", method: "fileinto" }
  ],
  name: "test",
  rules: [
    {
      field: "subject",
      operator: "contains",
      value: "test"
    }
  ]
}
```

filtre sogo

Analyse syntaxique

```
#test
require ["imapflags","fileinto"];

if anyof (
  header :contains "subject" "test"
) {
  stop;
  addflag "\\Seen";
  fileinto "INBOX";
}
```

script sieve

```
{
  actions: [
    { method: "stop" },
    { argument: "seen", method: "addflag" },
    { argument: "INBOX", method: "fileinto" }
  ],
  name: "test",
  rules: [
    {
      field: "subject",
      operator: "contains",
      value: "test"
    }
  ]
}
```

filtre sogo

Analyse syntaxique

```
#test
require ["imapflags","fileinto"];

if anyof (
  header :contains "subject" "test"
) {
  stop;
  addflag "\\Seen";
  fileinto "INBOX";
}
```

script sieve

```
{
  actions: [
    { method: "stop" },
    { argument: "seen", method: "addflag" },
    { argument: "INBOX", method: "fileinto" }
  ],
  name: "test",
  rules: [
    {
      field: "subject",
      operator: "contains",
      value: "test"
    }
  ]
}
```

filtre sogo

Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```

Analyse syntaxique

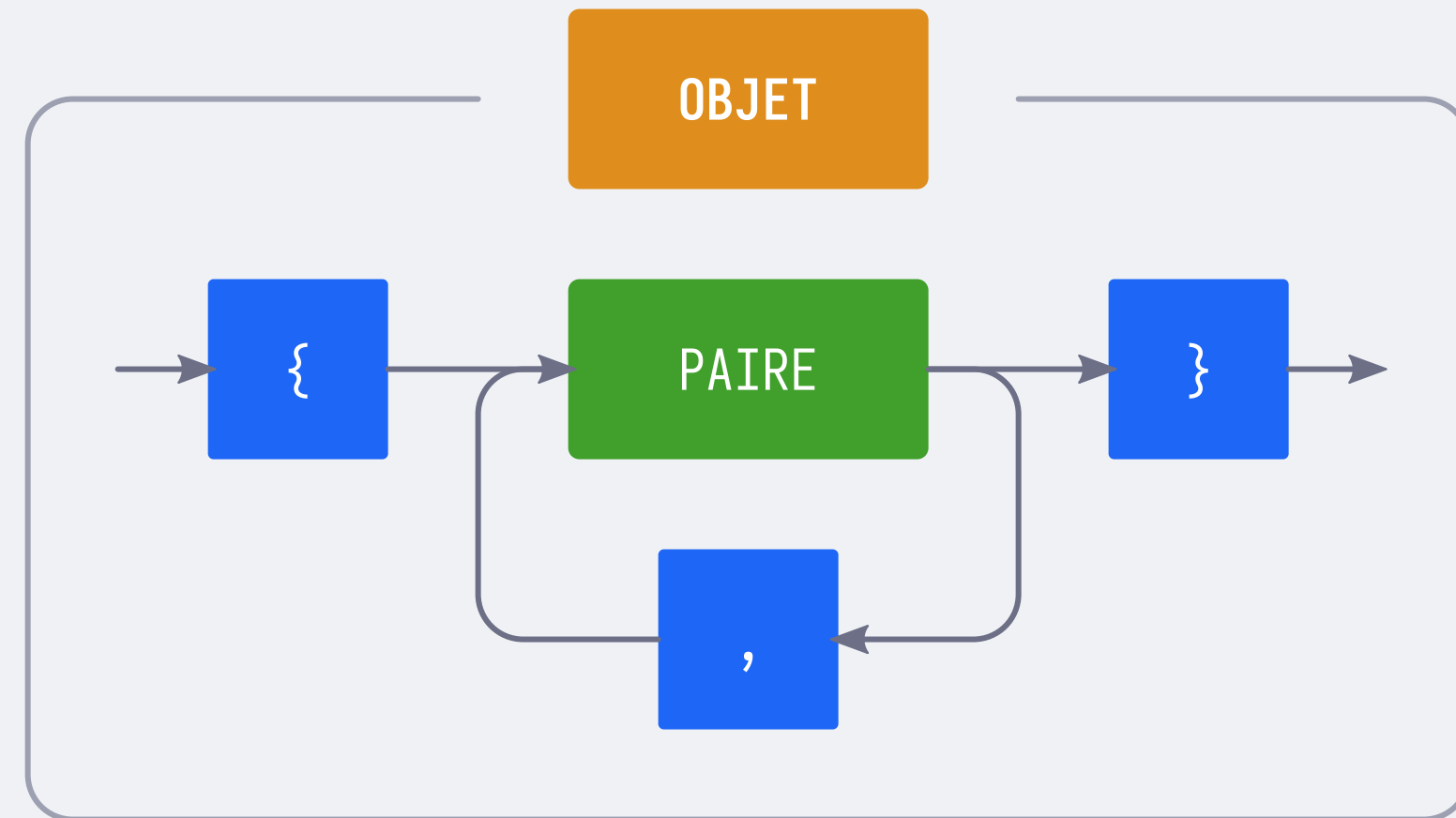
```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```


Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```

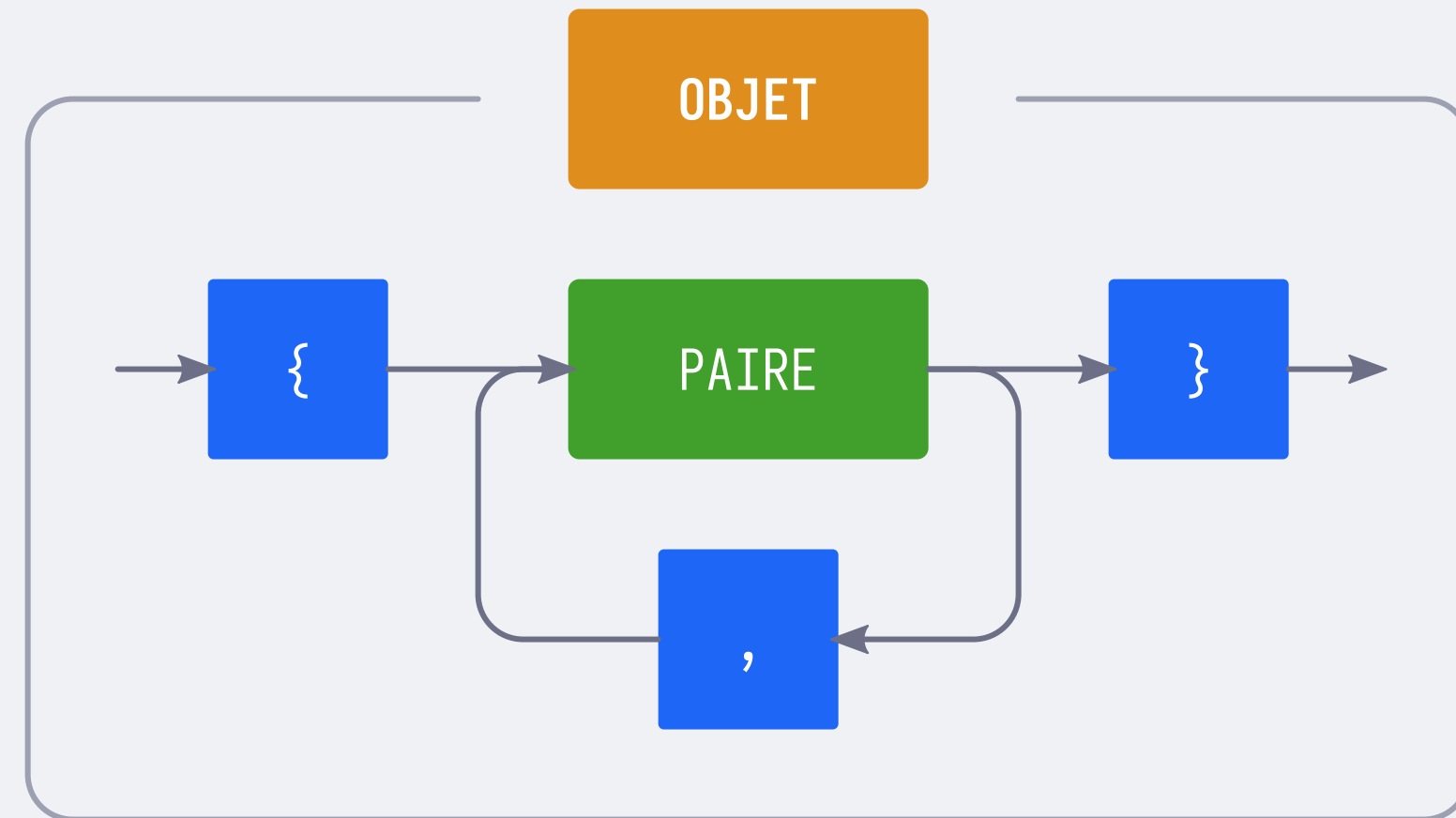
Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```



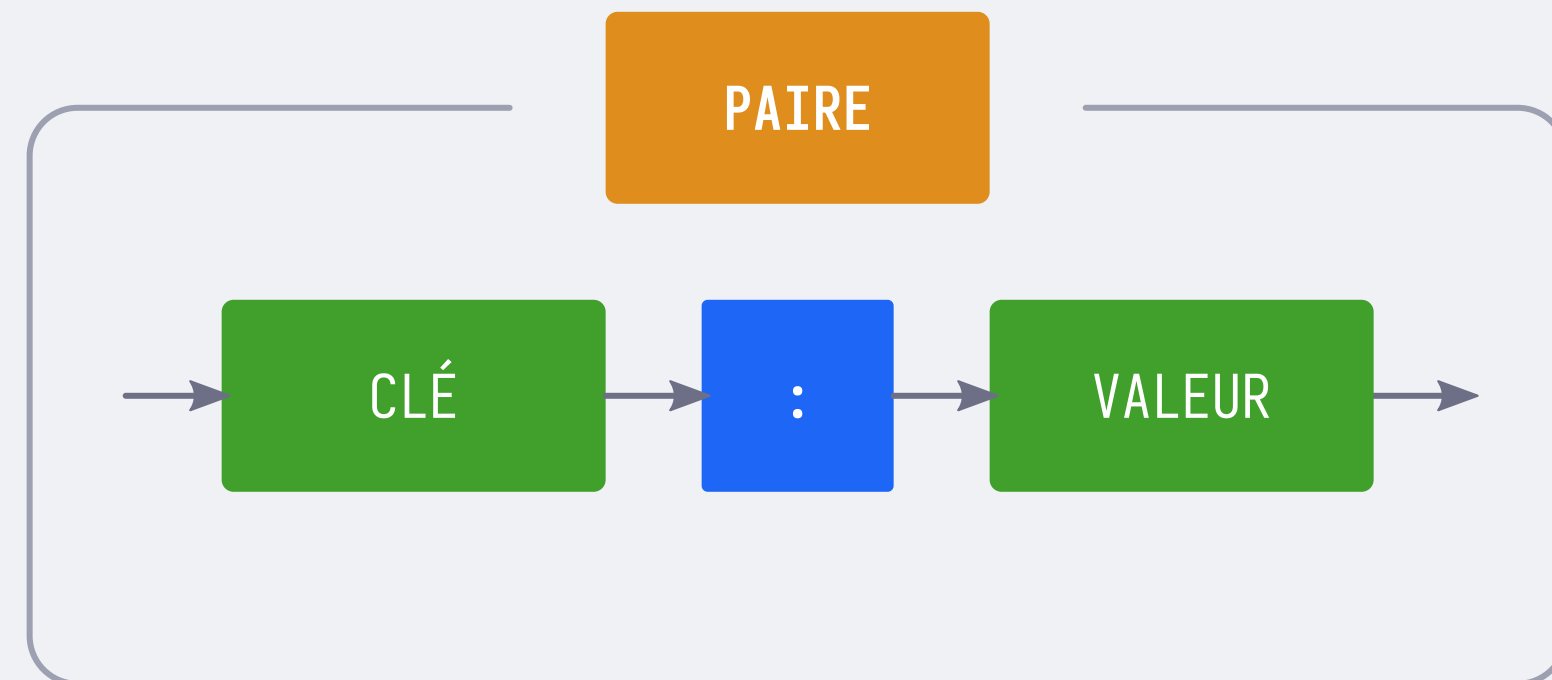
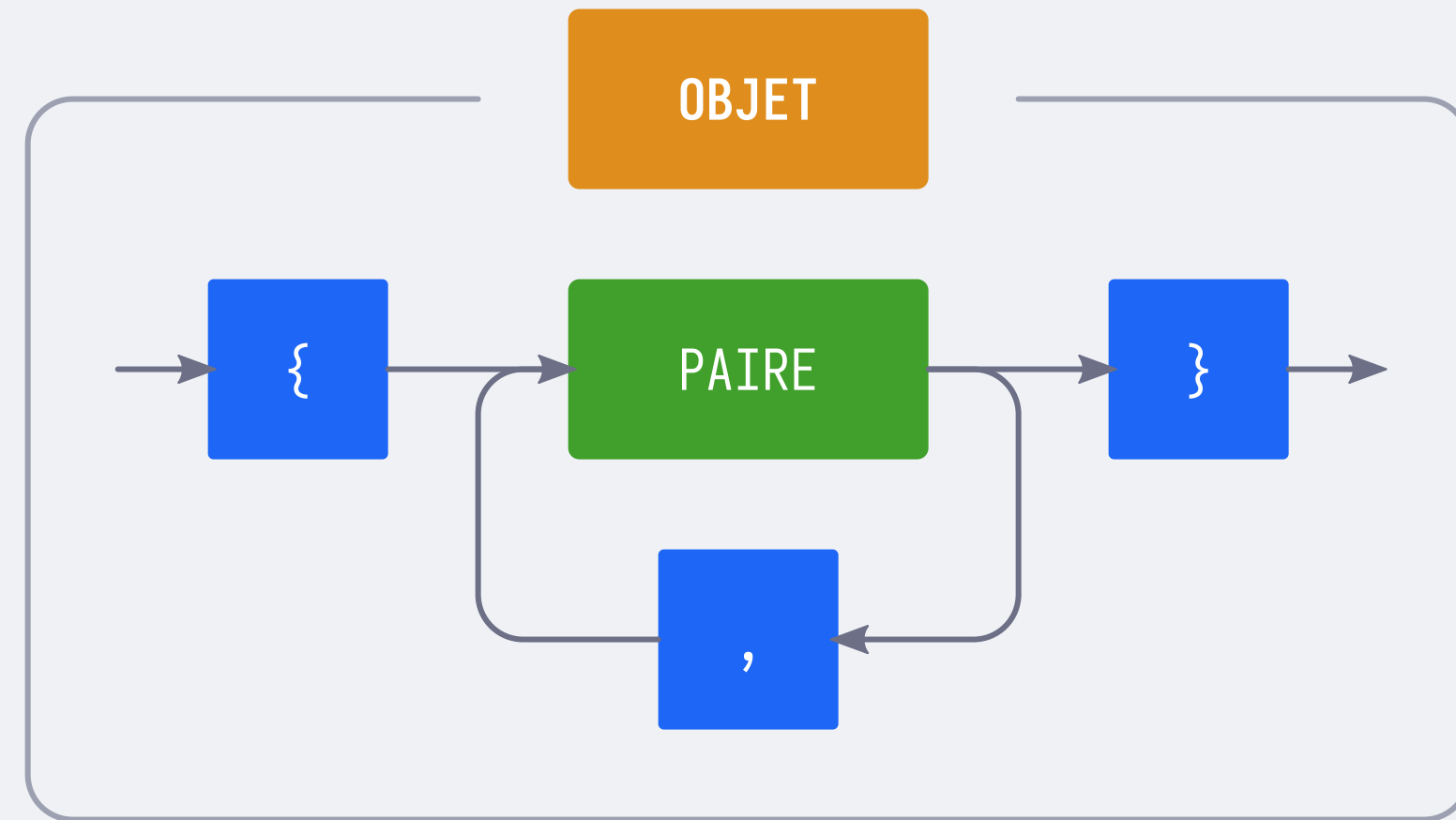
Analyse syntaxique

```
1  
{  
  2      3      4      5      6      7  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```



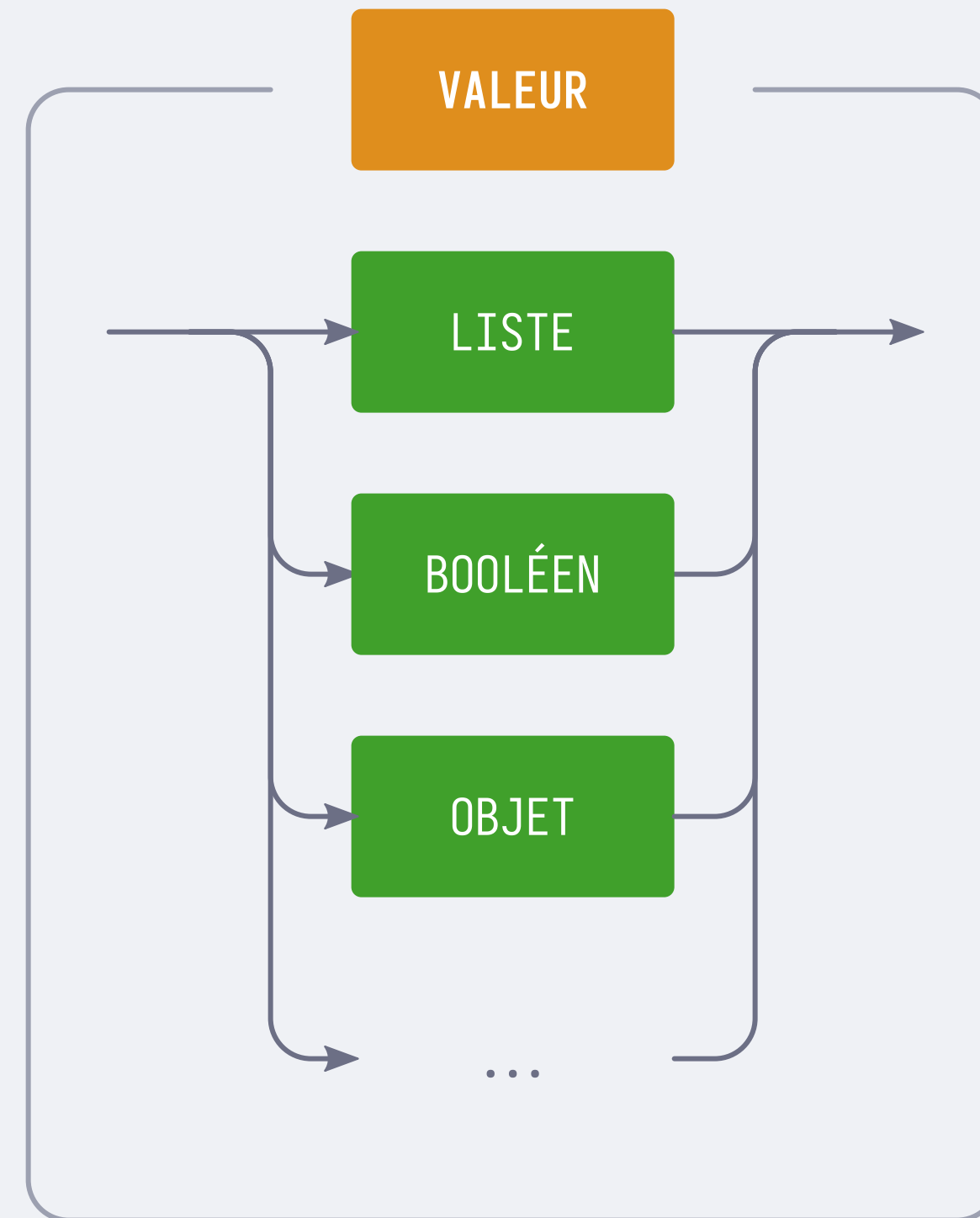
Analyse syntaxique

```
1  
{  
  2      3      4      5      6      7  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```



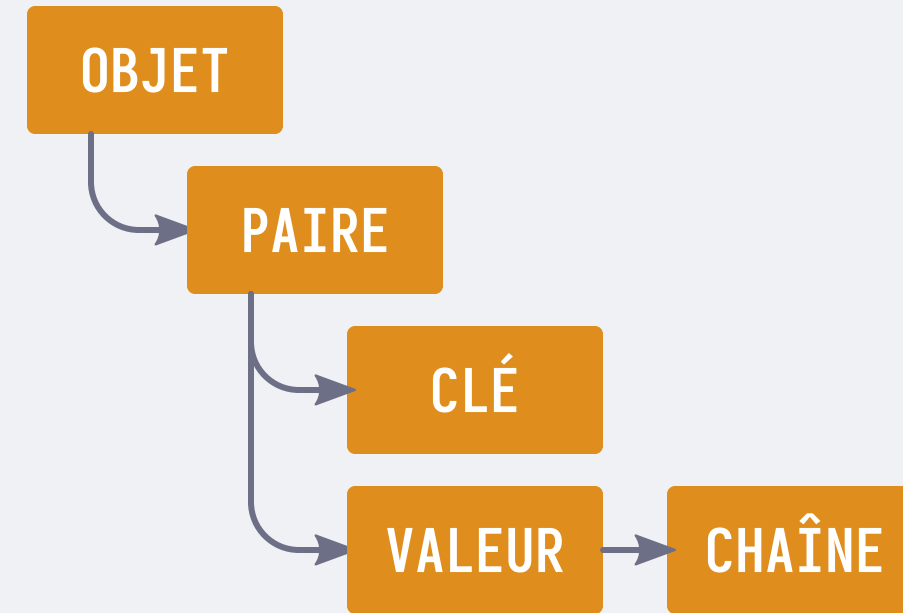
Analyse syntaxique

```
1  
{  
  2      3      4      5      6      7  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```



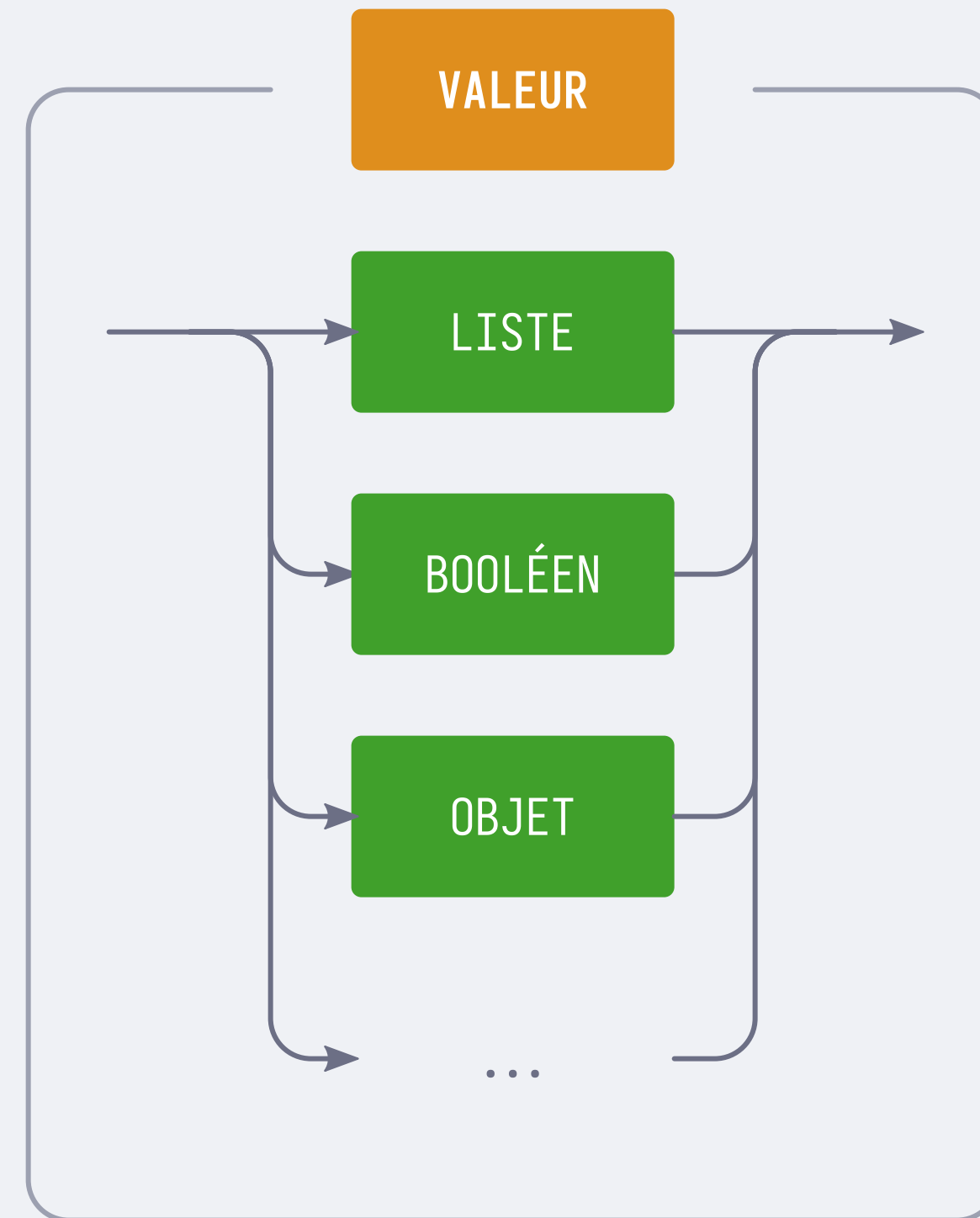
Analyse syntaxique

```
1  
{  
  2      3      4      5      6      7  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```



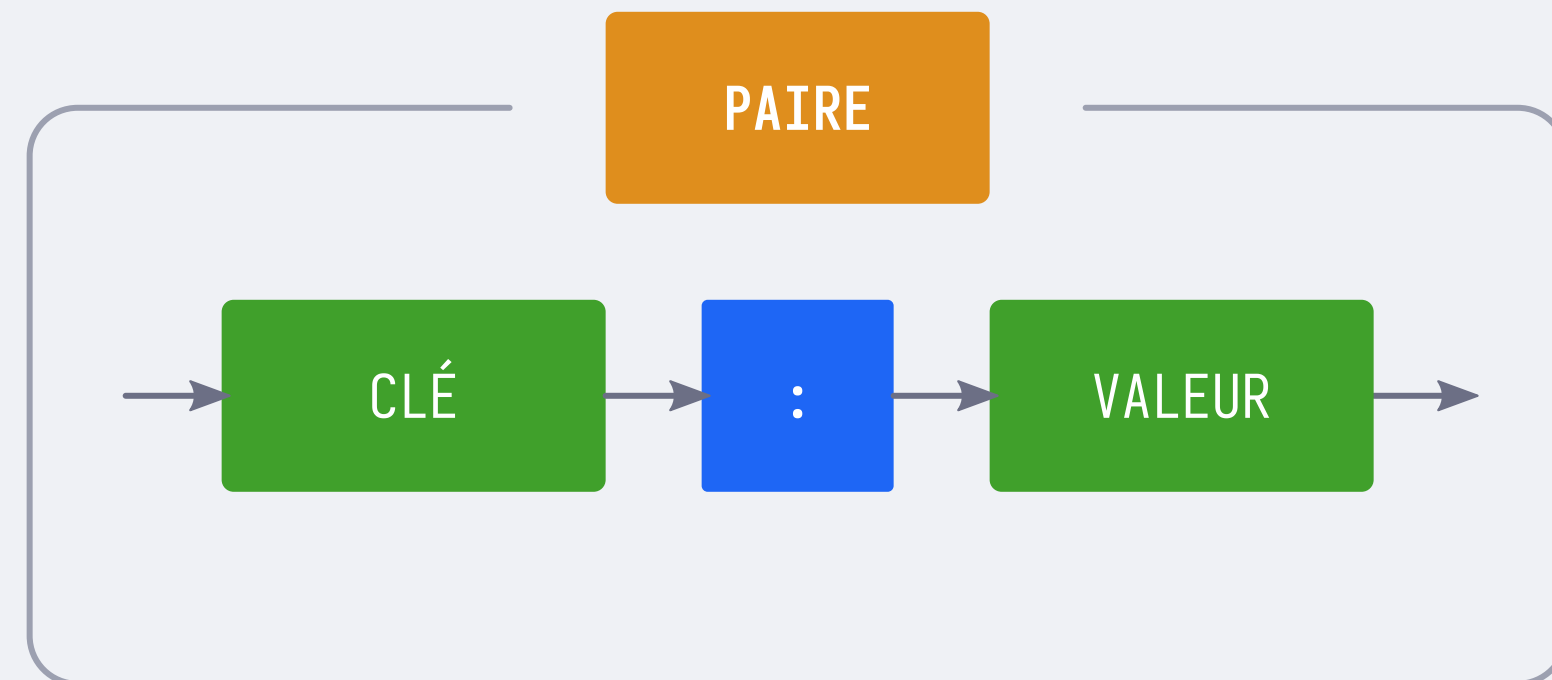
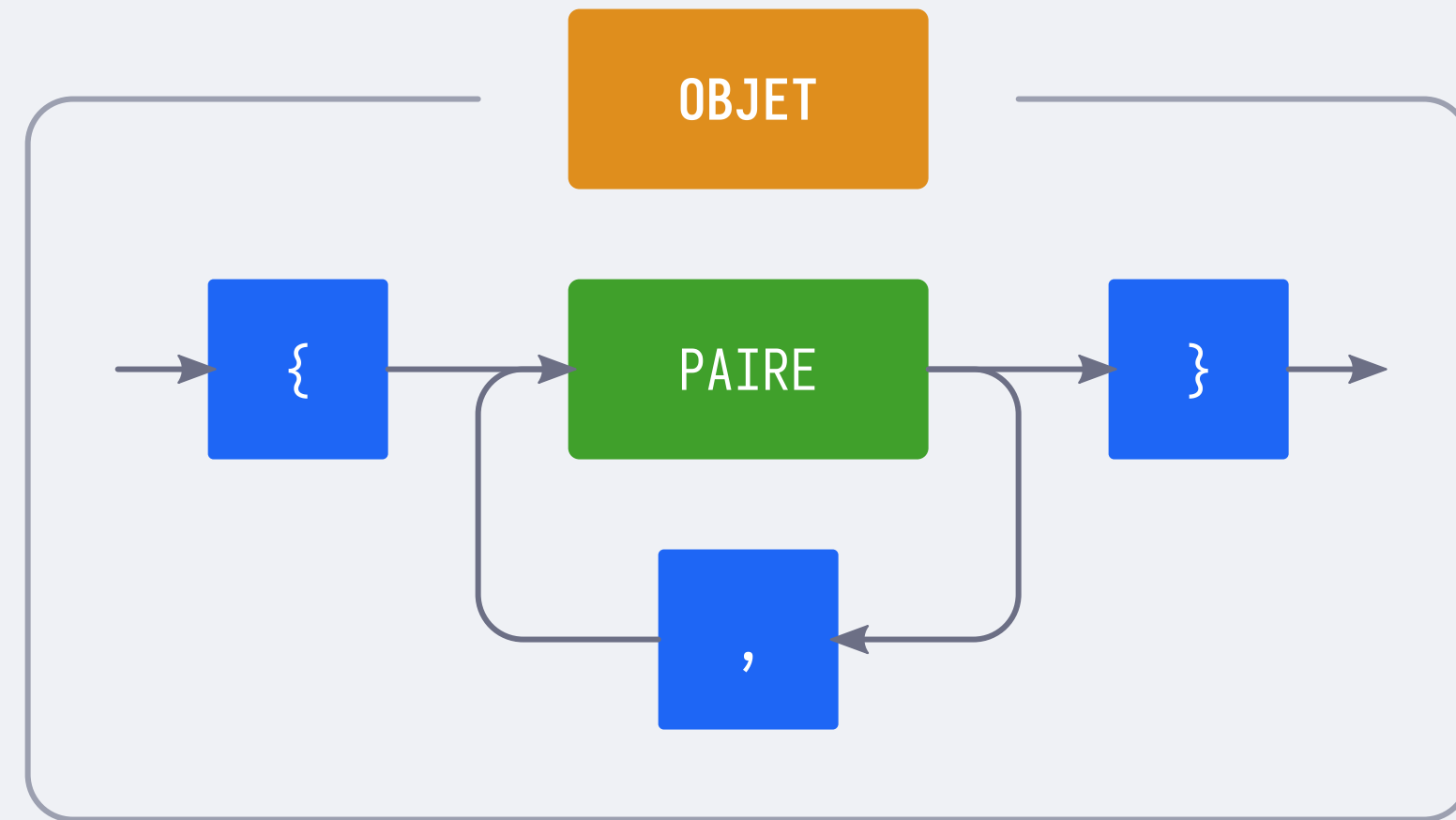
Analyse syntaxique

```
1 {  
  2 Lorem : "Ipsum",  
  3 Dolor : {  
    4 sit : "amet"  
  5 }  
6 }  
7 }
```



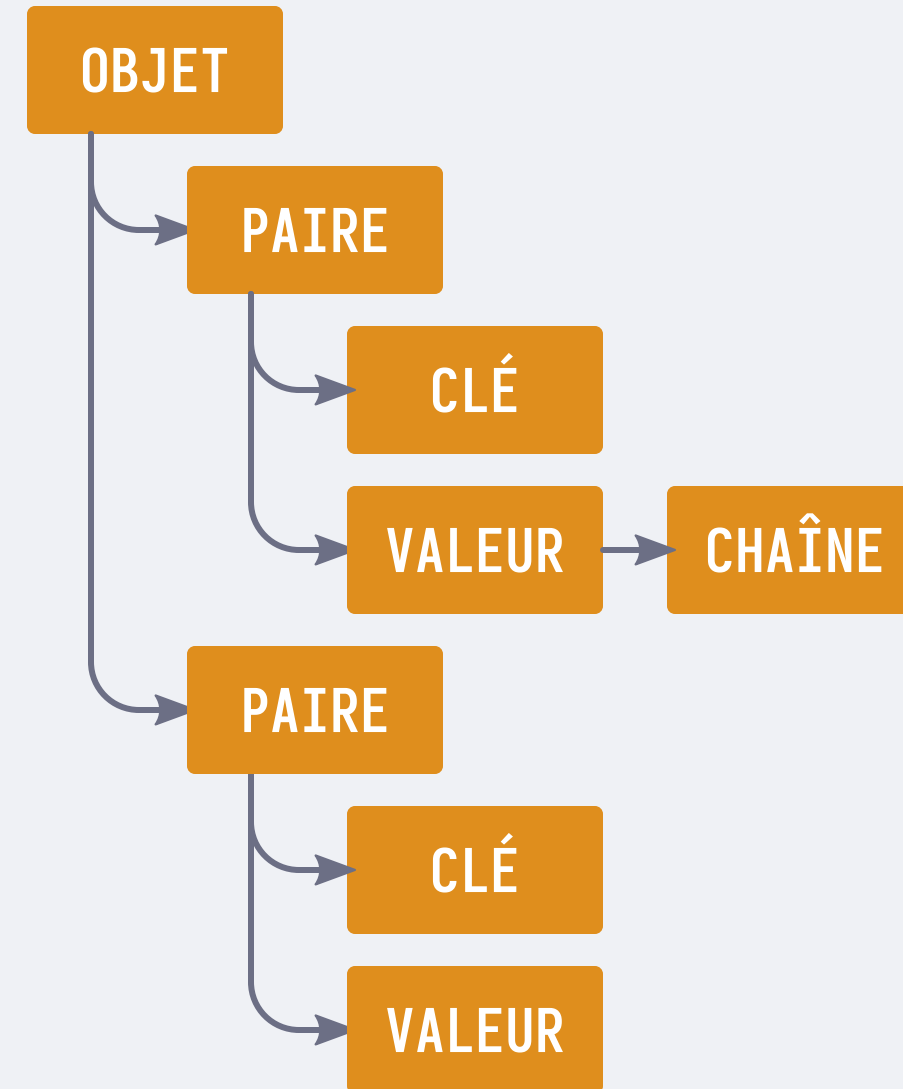
Analyse syntaxique

```
1  
{  
  2      3      4      5      6      7  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```



Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```

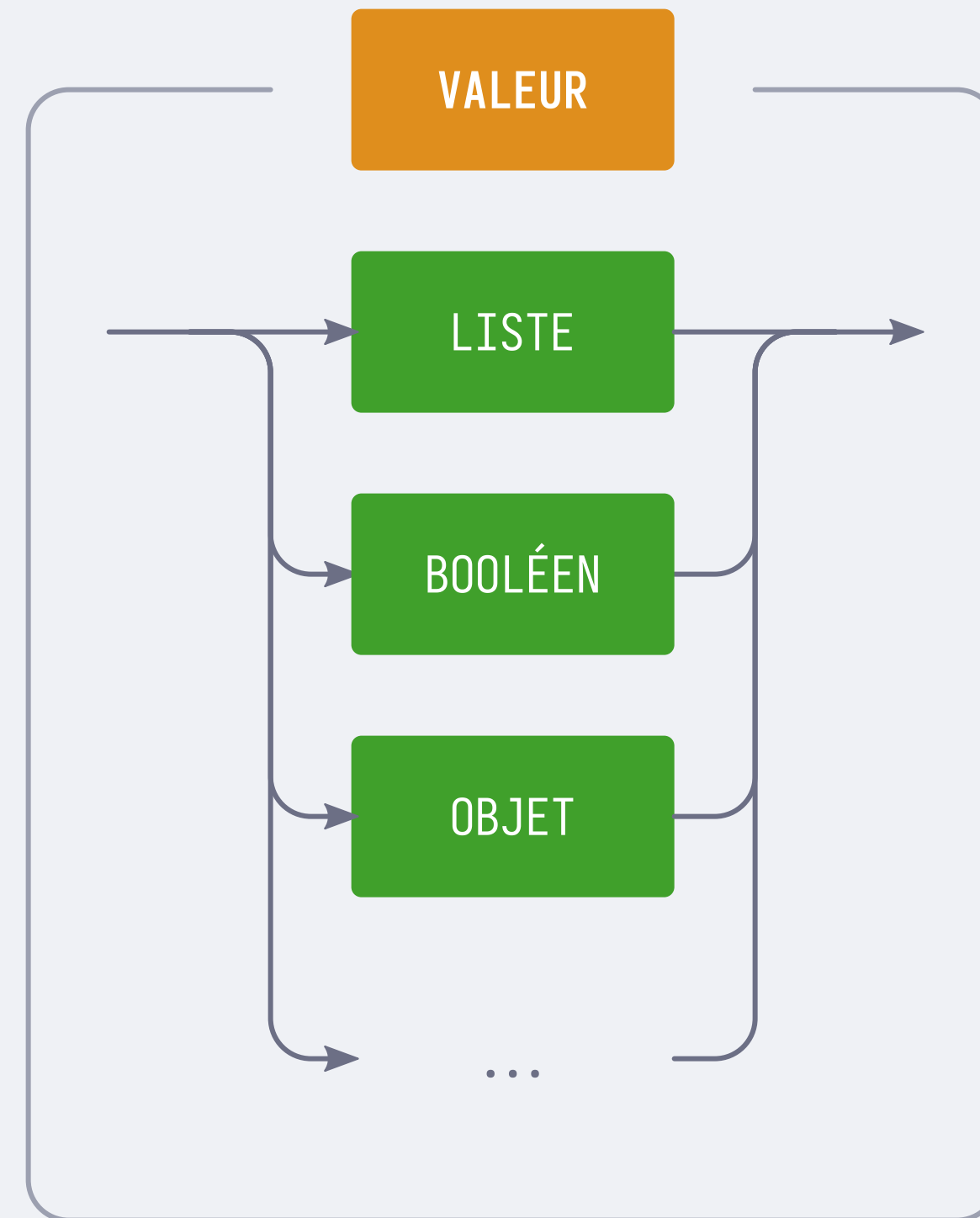


Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```

Diagram illustrating a JSON object structure with highlighted tokens and indices:

- Token 8: `8` (under `8`)
- Token 9: `9` (under `:`)
- Token 10: `10` (under `"`)

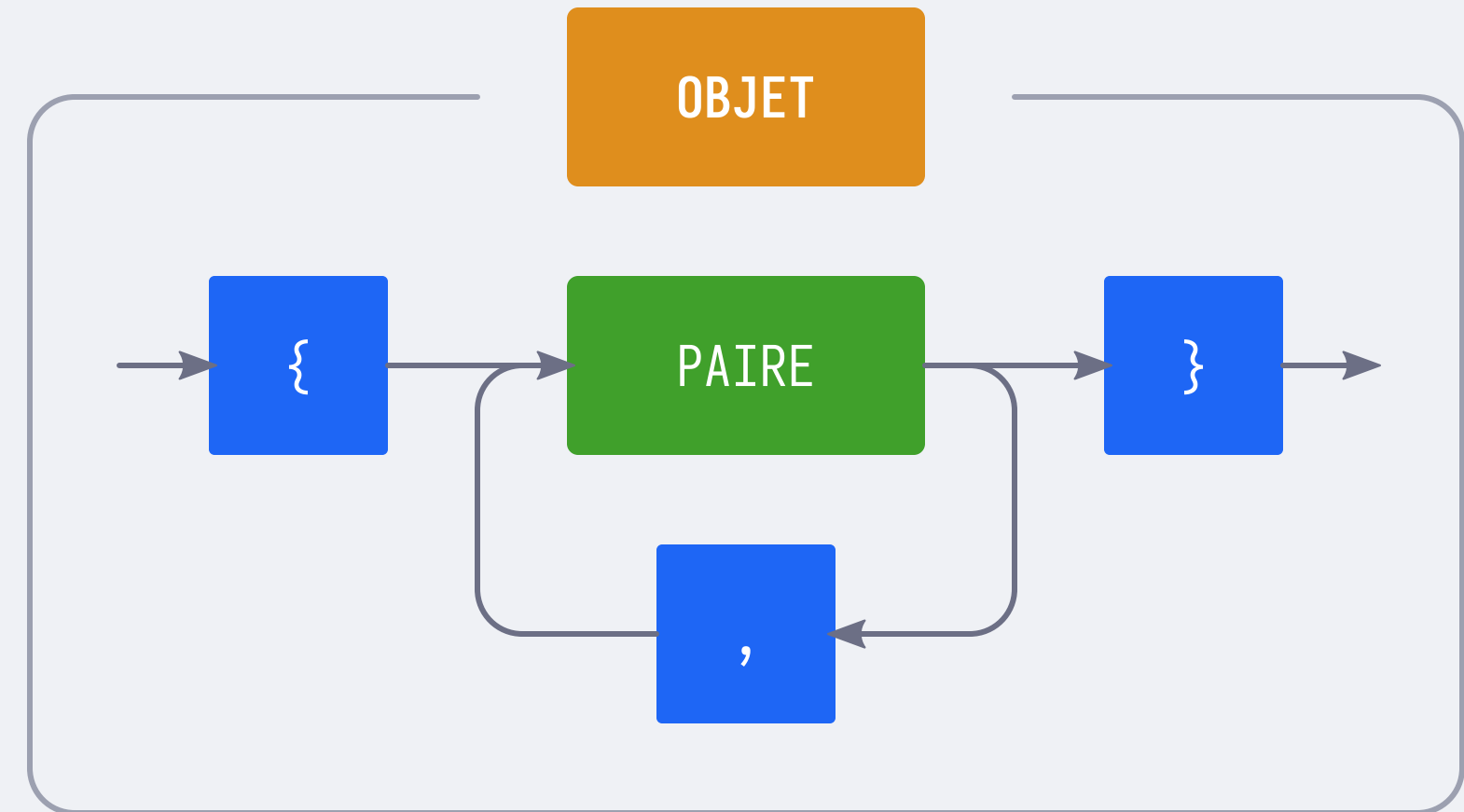


Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```

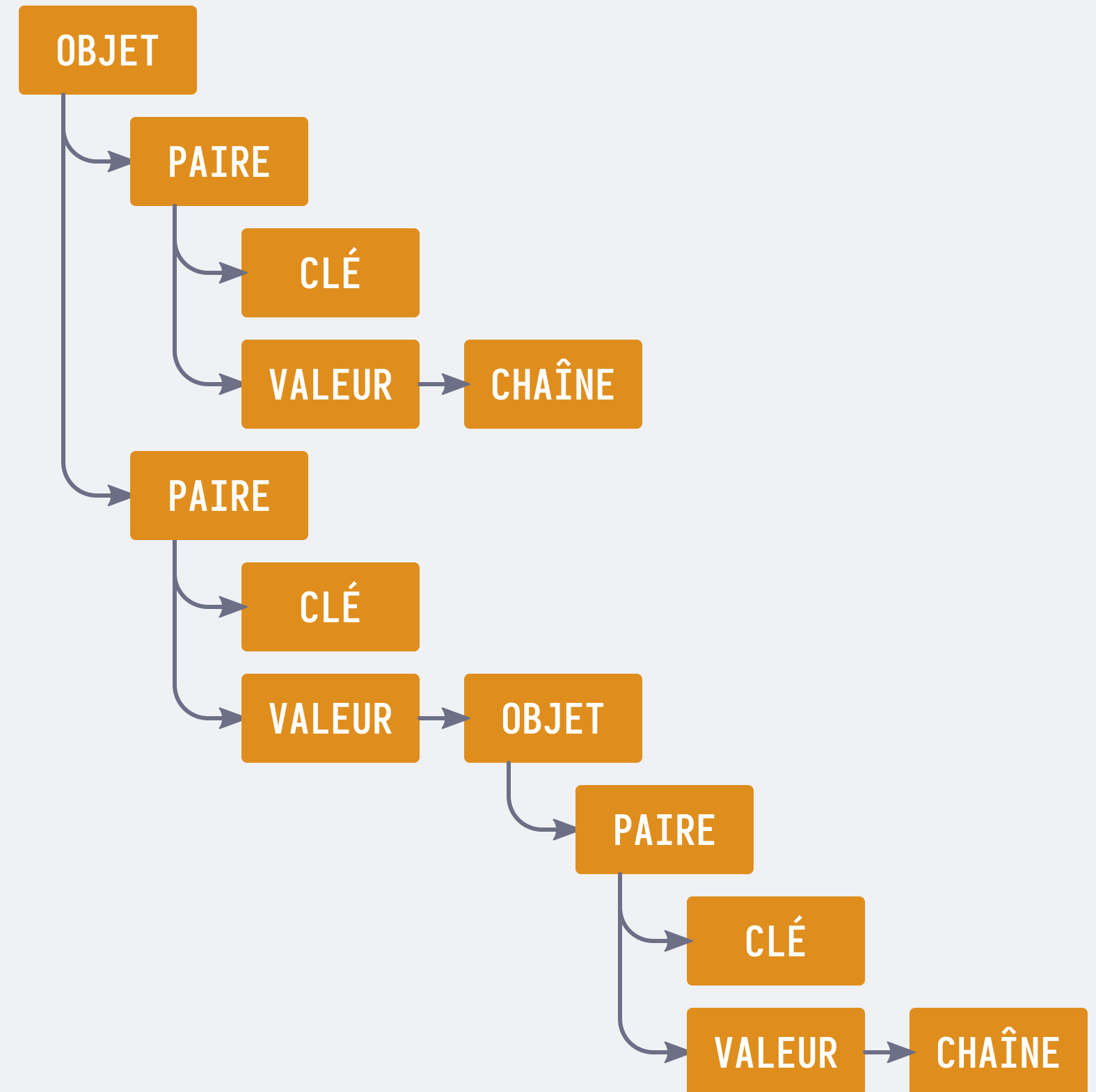
Diagram illustrating the parsing of a JSON object. The text is highlighted with colored circles and dots indicating the current state of the parser:

- 8: Under "Lorem"
- 9: Under ":"
- 10: Under "Ipsum"



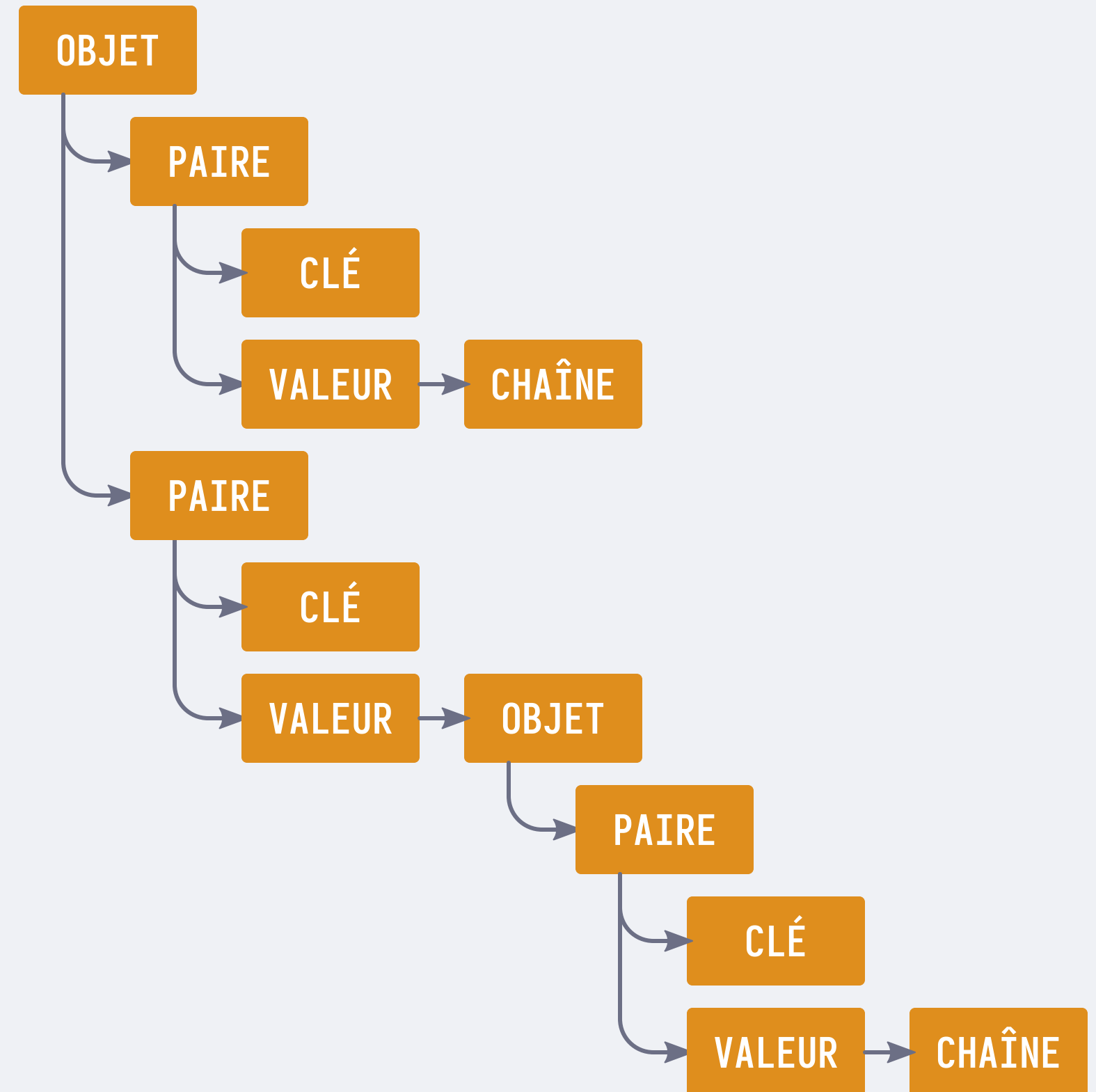
Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```

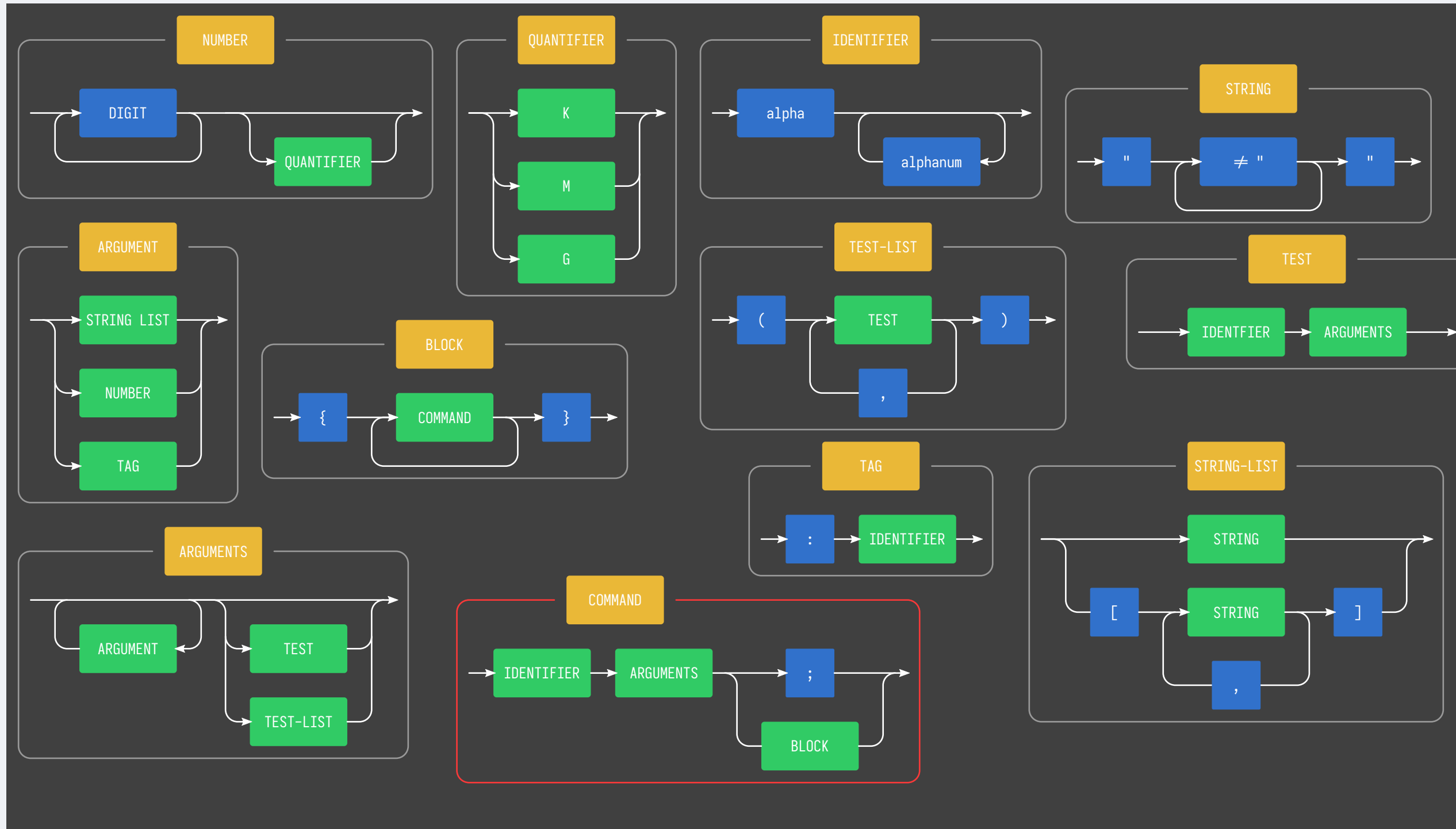


Analyse syntaxique

```
{  
  Lorem : "Ipsum",  
  Dolor : {  
    sit : "amet"  
  }  
}
```



Analyse syntaxique



Analyse syntaxique

Product Solutions Open Source Pricing

tonioo / sievelib Public

Code Issues 9 Pull requests 4 Actions Projects Wiki Security Insights

master 3 branches 13 tags Go to file Code

tonioo Merge pull request #102 from tonioo/fix/control_args cc76519 on Dec 7, 2020 215 commits

folder	sievelib	Don't forget command completion here.	3 years ago
file	.gitignore	Travis integration.	10 years ago
file	.travis.yml	Removed support for outdated Python versions.	3 years ago
file	COPYING	License file updated (MIT).	8 years ago
file	MANIFEST.in	Include the utf-8 test file in the source tarball.	8 years ago
file	README.rst	Better support of relational and date extensions.	4 years ago
file	requirements.txt	Put the list of requirements in setup.py	5 years ago
file	setup.cfg	Added setup.cfg file.	6 years ago
file	setup.py	Removed support for outdated Python versions.	3 years ago

README.rst

sievelib

build passing codecov 85% pypi package 1.2.1

Client-side Sieve and Managesieve library written in Python.

- Sieve : An Email Filtering Language ([RFC 5228](#))

Tree-sitter



[GitHub repository](#) 

[Introduction](#)

[Language Bindings](#)

[Parsers](#)

[Talks on Tree-sitter](#)

[Underlying Research](#)

[Using Parsers](#)

[Creating Parsers](#)

[Syntax Highlighting](#)

[Implementation](#)


[Contributing](#)

[Playground](#)

[Code Navigation Systems](#)












Introduction

Tree-sitter is a parser generator tool and an incremental parsing library. It can build a concrete syntax tree for a source file and efficiently update the syntax tree as the source file is edited. Tree-sitter aims to be:

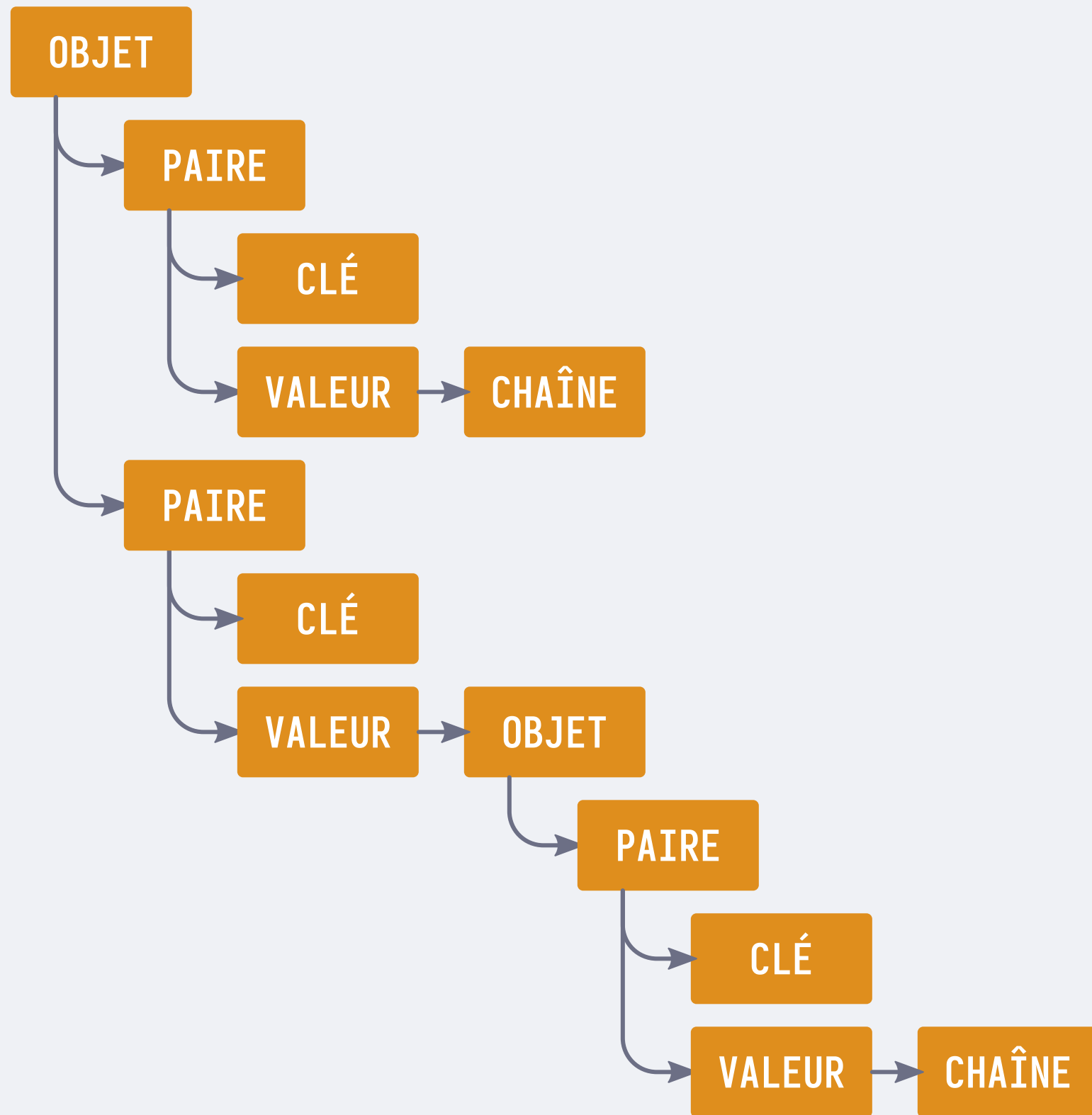
- **General** enough to parse any programming language
- **Fast** enough to parse on every keystroke in a text editor
- **Robust** enough to provide useful results even in the presence of syntax errors
- **Dependency-free** so that the runtime library (which is written in pure C ) can be embedded in any application

Language Bindings

There are currently bindings that allow Tree-sitter to be used from the following languages:

- [Go](#) 
- [Haskell](#) 
- [Java](#) 
- [JavaScript \(Node.js\)](#) 
- [JavaScript \(Wasm\)](#) 
- [Kotlin](#) 
- [Lua](#) 
- [OCaml](#) 
- [Perl](#) 
- [Python](#) 
- [Ruby](#) 

Tree-sitter



```
( OBJET
  ( PAIRE
    ( CLÉ )
    ( VALEUR ( CHAÎNE )))
  ( PAIRE
    ( CLÉ )
    ( VALEUR ( OBJET
      ( PAIRE
        ( CLÉ )
        ( VALEUR ( CHAÎNE ))))))))
```

Tree-sitter – Query

```
( OBJET
  ( PAIRE
    ( CLÉ )
    ( VALEUR ( CHAÎNE )))
  ( PAIRE
    ( CLÉ )
    ( VALEUR
      ( OBJET
        ( PAIRE
          ( CLÉ )
          ( VALEUR ( CHAÎNE ))))))))
```

```
( VALEUR ( CHAÎNE ))
```

Tree-sitter – Query

```
( OBJET
  ( PAIRE
    ( CLÉ )
    ( VALEUR ( CHAÎNE )))
  ( PAIRE
    ( CLÉ )
    ( VALEUR
      ( OBJET
        ( PAIRE
          ( CLÉ )
          ( VALEUR ( CHAÎNE ))))))))
```

```
( VALEUR ( CHAÎNE )) @MOT
```

Tree-sitter – Query

```
( OBJET
  ( PAIRE
    ( CLÉ )
    ( VALEUR ( CHAÎNE )))
  ( PAIRE
    ( CLÉ )
    ( VALEUR
      ( OBJET
        ( PAIRE
          ( CLÉ )
          ( VALEUR ( CHAÎNE ))))))))
```

```
( VALEUR ( CHAÎNE )) @MOT

( OBJET
  ( PAIRE
    ( VALEUR ( OBJET ))))
```

Tree-sitter – Query

```
( OBJET
  ( PAIRE
    ( CLÉ )
    ( VALEUR ( CHAÎNE )))
  ( PAIRE
    ( CLÉ )
    ( VALEUR
      ( OBJET
        ( PAIRE
          ( CLÉ )
          ( VALEUR ( CHAÎNE ))))))))
```

```
( VALEUR ( CHAÎNE )) @MOT

( OBJET
  ( PAIRE
    ( VALEUR ( OBJET ) @ENFANT2 )))
```

Tree-sitter – Query

```
func add(x int, y int) int {  
  return x + y  
}
```

[code source](#)

Tree-sitter – Query

```
func add(x int, y int) int {  
  return x + y  
}
```

code source

```
(function_declaration  
  name: (identifier)  
  parameters: (parameter_list)  
    (parameter_declaration  
      name: (identifier)  
      type: (type_identifier))  
    (parameter_declaration  
      name: (identifier)  
      type: (type_identifier))  
  result : (type_identifier)  
  ...
```

arbre de syntaxe

Tree-sitter – Query

```
func add(x int, y int) int {  
  return x + y  
}
```

code source

```
(function_declaration  
  name : (identifier) @nom  
  parameters : (parameter_list  
    (parameter_declaration  
      name : (identifier) @parametre )))
```

requête

```
(function_declaration  
  name: (identifier)  
  parameters: (parameter_list  
    (parameter_declaration  
      name: (identifier)  
      type: (type_identifier))  
    (parameter_declaration  
      name: (identifier)  
      type: (type_identifier))  
  result : (type_identifier)  
  ...
```

arbre de syntaxe

Tree-sitter – Query

```
function add1(a, b) { return a+b; }  
let add2 = (a, b) => { return a+b; };
```

[code source](#)

Tree-sitter – Query

```
function add1(a, b) { return a+b; }  
let add2 = (a, b) => { return a+b; };
```

code source

```
(function_declaration  
  name: (identifier)  
  ...  
  
(variable_declarator  
  name : (identifier)  
  value : (arrow_function)  
  ...
```

arbre de syntaxe

Tree-sitter – Query

```
function add1(a, b) { return a+b; }  
let add2 = (a, b) => { return a+b; };
```

code source

```
(function_declaration  
  name : (identifier) @nom )  
  
(variable_declarator  
  name : (identifier) @nom  
  value : (arrow_function))
```

requête

```
(function_declaration  
  name: (identifier)  
  ...  
  
(variable_declarator  
  name : (identifier)  
  value : (arrow_function)  
  ...
```

arbre de syntaxe

Tree-sitter – Coloration syntaxique

```
type Node struct {  
    data string  
    left * Node  
    right * Node  
}  
  
func (self * Node) GetData() string {  
    return self.data  
}  
  
func (self * Node) GetLeft() * Node {  
    return self.left  
}
```

Regex

Tree-sitter – Coloration syntaxique

```
type Node struct {  
    data string  
    left * Node  
    right * Node  
}  
  
func (self * Node) GetData() string {  
    return self.data  
}  
  
func (self * Node) GetLeft() * Node {  
    return self.left  
}
```

Regex

```
type Node struct {  
    data string  
    left * Node  
    right * Node  
}  
  
func (self * Node) GetData() string {  
    return self.data  
}  
  
func (self * Node) GetLeft() * Node {  
    return self.left  
}
```

Tree-sitter

Tree-sitter – Coloration syntaxique

```
function test(t)
  for k, v in pairs(t) do
    print(k, v)
  end
end
```

Tree-sitter – Coloration syntaxique

```
for k, v in pairs(t) do
  for i, d in ipairs(v) do
    for j, e in ipairs(d) do
      print(d)
    end
  end
end
end
```

Tree-sitter – Coloration syntaxique

```
\begin{document}
```

```
\begin{center}
```

```
\begin{align}
```

```
\end{align}
```

```
\end{center}
```

```
\end{document}
```


Tree-sitter – Query

```
( OBJET
  ( PAIRE
    ( CLÉ )
    ( VALEUR ( CHAÎNE )))
  ( PAIRE
    ( CLÉ )
    ( VALEUR
      ( OBJET
        ( PAIRE
          ( CLÉ )
          ( VALEUR ( CHAÎNE ))))))))
```


Tree-sitter – Coloration syntaxique

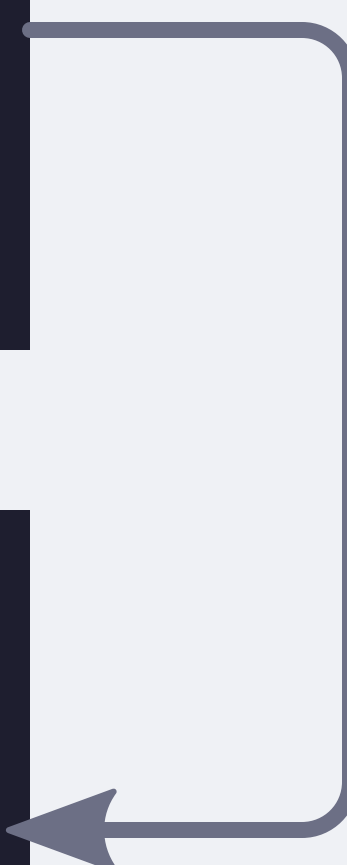
```
<div class="cards">  
<% cards.forEach( card => { %>  
  <%- include("card", card); %>  
<% }); %>  
</div>
```

Tree-sitter – Coloration syntaxique

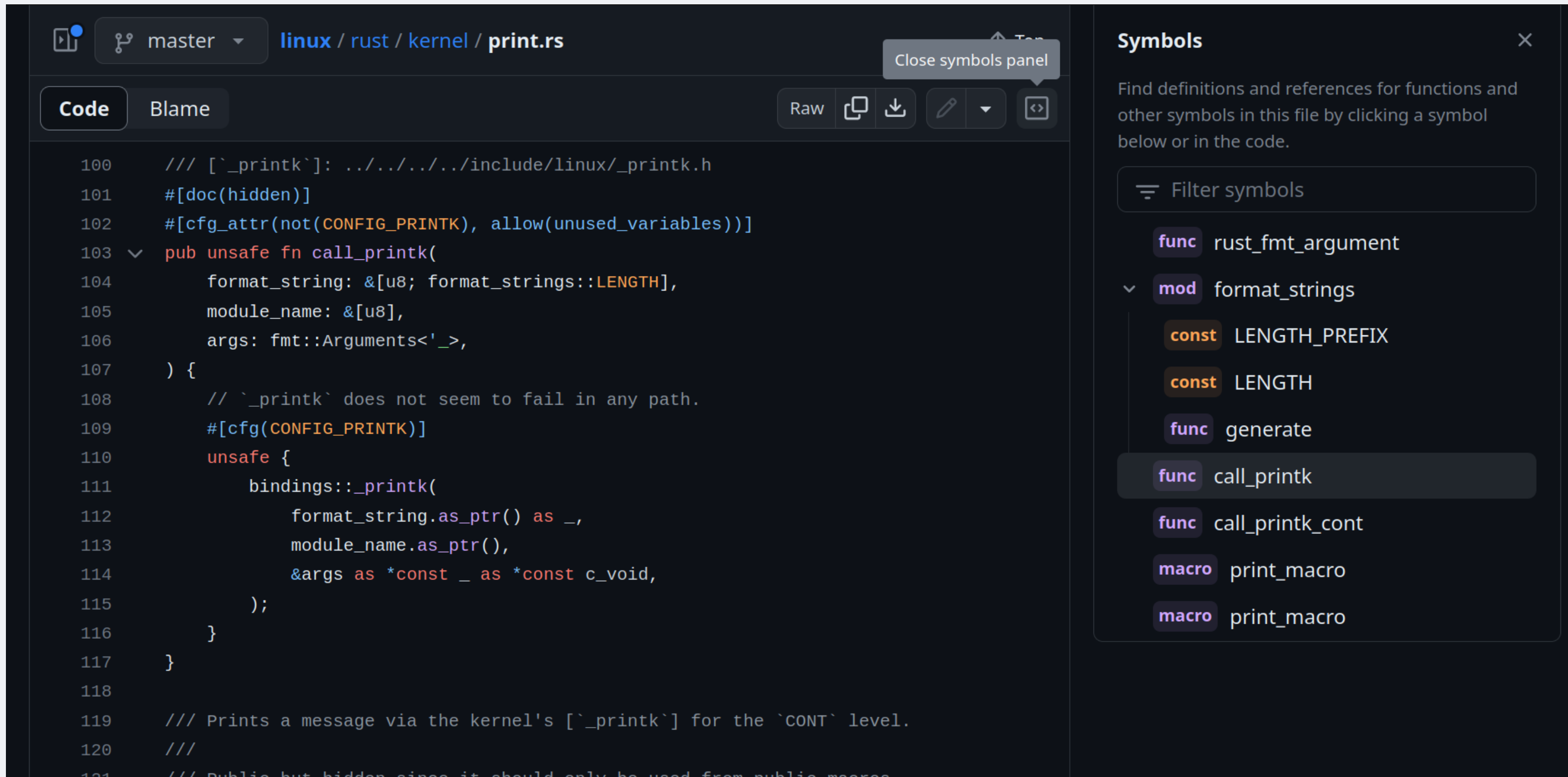
```
<div class="cards">
<% cards.forEach( card => { %>
  <%- include("card", card); %>
<% }); %>
</div>
```

```
<div class="cards">
<% cards.forEach( card => { %>
  <%- include("card", card); %>
<% }); %>
</div>
```

Injection de langage entre
<% et %>



GitHub



The screenshot shows a GitHub code editor interface. The top navigation bar includes a repository path: `linux / rust / kernel / print.rs`. Below the path are tabs for `Code` and `Blame`. The main editor area displays Rust code for the `call_printk` function, with line numbers 100 through 121. The code includes comments, doc attributes, and a function definition. On the right side, a `Symbols` panel is open, listing symbols found in the file: `rust_fmt_argument` (func), `format_strings` (mod), `LENGTH_PREFIX` (const), `LENGTH` (const), `generate` (func), `call_printk` (func), `call_printk_cont` (func), and two instances of `print_macro` (macro). The `call_printk` symbol is highlighted.

```
100  /// [`_printk`]: ../../../../include/linux/_printk.h
101  #[doc(hidden)]
102  #[cfg_attr(not(CONFIG_PRINTK), allow(unused_variables))]
103  pub unsafe fn call_printk(
104      format_string: &[u8; format_strings::LENGTH],
105      module_name: &[u8],
106      args: fmt::Arguments<'_>,
107  ) {
108      // `_printk` does not seem to fail in any path.
109      #[cfg(CONFIG_PRINTK)]
110      unsafe {
111          bindings::_printk(
112              format_string.as_ptr() as _,
113              module_name.as_ptr(),
114              &args as *const _ as *const c_void,
115          );
116      }
117  }
118
119  /// Prints a message via the kernel's [`_printk`] for the `CONT` level.
120  ///
121  /// Public but hidden since it should only be used from public macros
```

GitHub

The screenshot shows a GitHub code editor interface. The top bar indicates the repository path: `linux / rust / kernel / print.rs`. The code editor displays the following Rust code:

```
100  /// [`_printk`]: ../../../../../../include/linux/_printk.h
101  #[doc(hidden)]
102  #[cfg_attr(not(CONFIG_PRINTK), allow(unused_variables))]
103  pub unsafe fn call_printk(
104      format_string: &[u8; format_strings::LENGTH],
105      module_name: &[u8],
106      args: fmt::Arguments<'_>,
107  ) {
108      // `_printk` does not seem to fail in any path.
109      #[cfg(CONFIG_PRINTK)]
110      unsafe {
111          bindings::_printk(
112              format_string.as_ptr() as _,
113              module_name.as_ptr(),
114              &args as *const _ as *const c_void,
115          );
116      }
117  }
118
119  /// Prints a message via the kernel's [`_printk`] for the `CONT` level.
120  ///
121  /// Public but hidden since it should only be used from public macros
```

On the right side, a sidebar titled "All Symbols" is open, showing the function `function call_printk`. It includes a "Definition Search" section with "In this file" and a "2 References Search" section with "In this file". The references listed are:

- 103 `pub unsafe fn call_printk(`
- 139 `the string to [call_printk].`
- 161 `$crate::print::call_printk(`

A search bar at the bottom of the sidebar prompts "Search for this symbol".

Tree-sitter – Nvim



Coloration syntaxique

Folding

Navigation

Text-objects

...

Tree-sitter – Nvim

```
def fib(n) :  
    a, b = 0, 1  
    while a < n :  
        print(a, end=' ')  
        a, b = b, a+b  
    print()
```

```
fib(1000)
```


Tree-sitter – Nvim

```
def fib(n) :  
    a, b = 0, 1  
    while a < n :  
        print(a, end=' ')  
        a, b = b, a+b  
    print()
```

```
fib(1000)
```

@function.inner

```
(function_definition  
  body : (block) @function.inner  
) @function.outer
```

Tree-sitter – Nvim

```
def fib(n) :  
    a, b = 0, 1  
    while a < n :  
        print(a, end=' ')  
        a, b = b, a+b  
    print()
```

```
fib(1000)
```

@function.outer

```
(function_definition  
  body : (block) @function.inner  
) @function.outer
```

Tree-sitter



[GitHub repository](#)

[Introduction](#)

[Using Parsers](#)

[Creating Parsers](#)

[Syntax Highlighting](#)

[Implementation](#)

[Contributing](#)

[Playground](#)

[Code Navigation Systems](#)

Code Log Query

```
1 int main() {  
2     return 0;  
3 }
```

Query

```
1 (function_definition  
2   (function_declarator  
3     declarator : (identifier) @function))  
4  
5 (number_literal) @number  
6  
7 "return" @keyword
```

Tree 1.0 ms

```
translation_unit [0, 0] - [3, 0]  
  function_definition [0, 0] - [2, 1]  
    type: primitive_type [0, 0] - [0, 3]  
    declarator: function_declarator [0, 4] - [0, 10]  
      declarator: identifier [0, 4] - [0, 8]  
      parameters: parameter_list [0, 8] - [0, 10]  
    body: compound_statement [0, 11] - [2, 1]  
      return_statement [1, 1] - [1, 10]  
        number_literal [1, 8] - [1, 9]
```