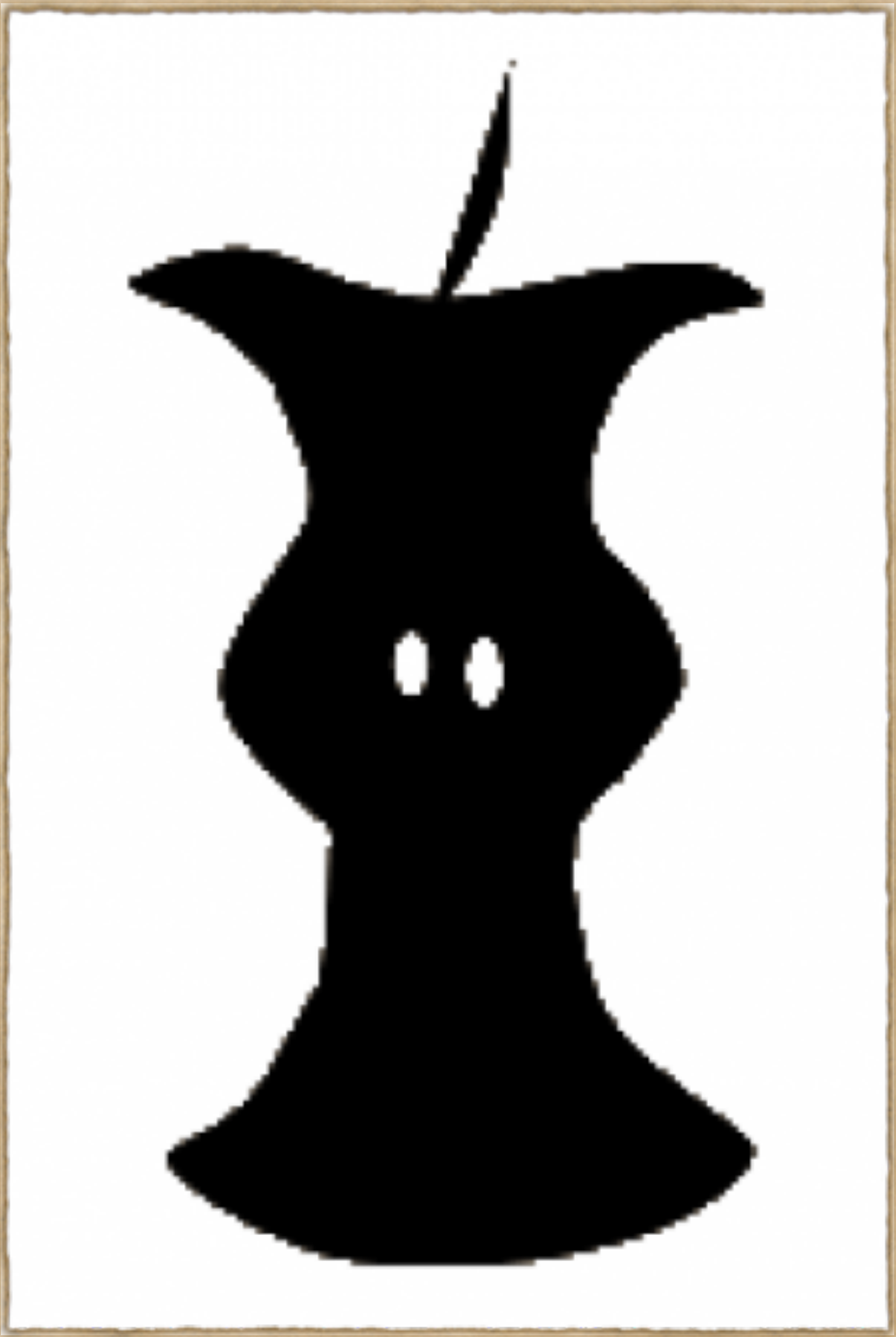
Парсеры - это спарта

CAUTION
THIS IS SPARTA

Алексей Охрименко

Twitter: @Ai_boy



{}
BACKEND
DEVELOPER

</>
FRONTEND
DEVELOPER

[redacted]

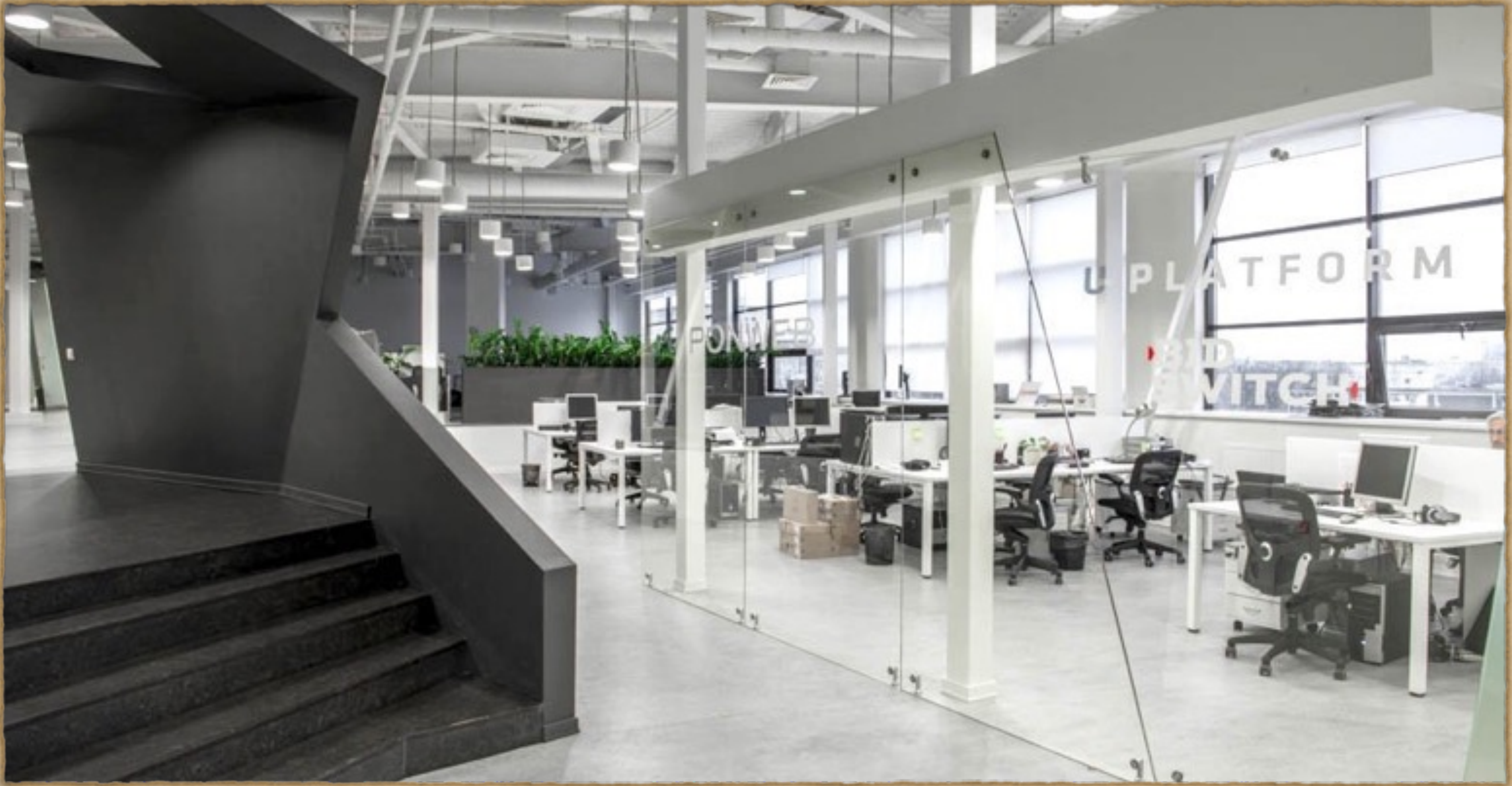
ПРОСТИ, ЧЕБУРАШКА

ты родился в Спарте

Lionidus & HIGHLOAD 2015

Тимур

infosystems jet

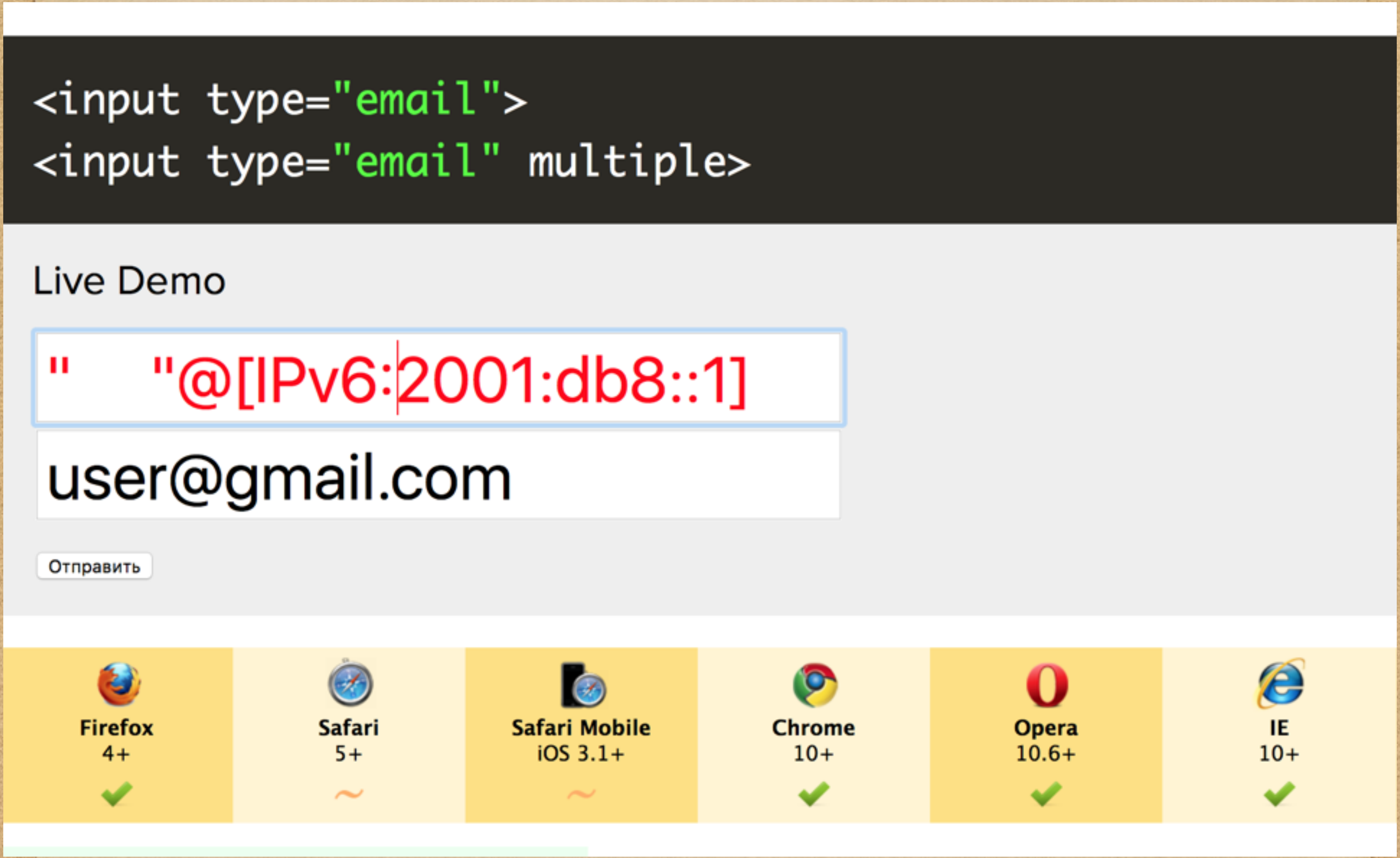


IPONWEB

RTB

IPONWEB
IPONWEB
IPONWEB
ВСЕ ОЧЕНЬ СЛОЖНО

Вы готовы?



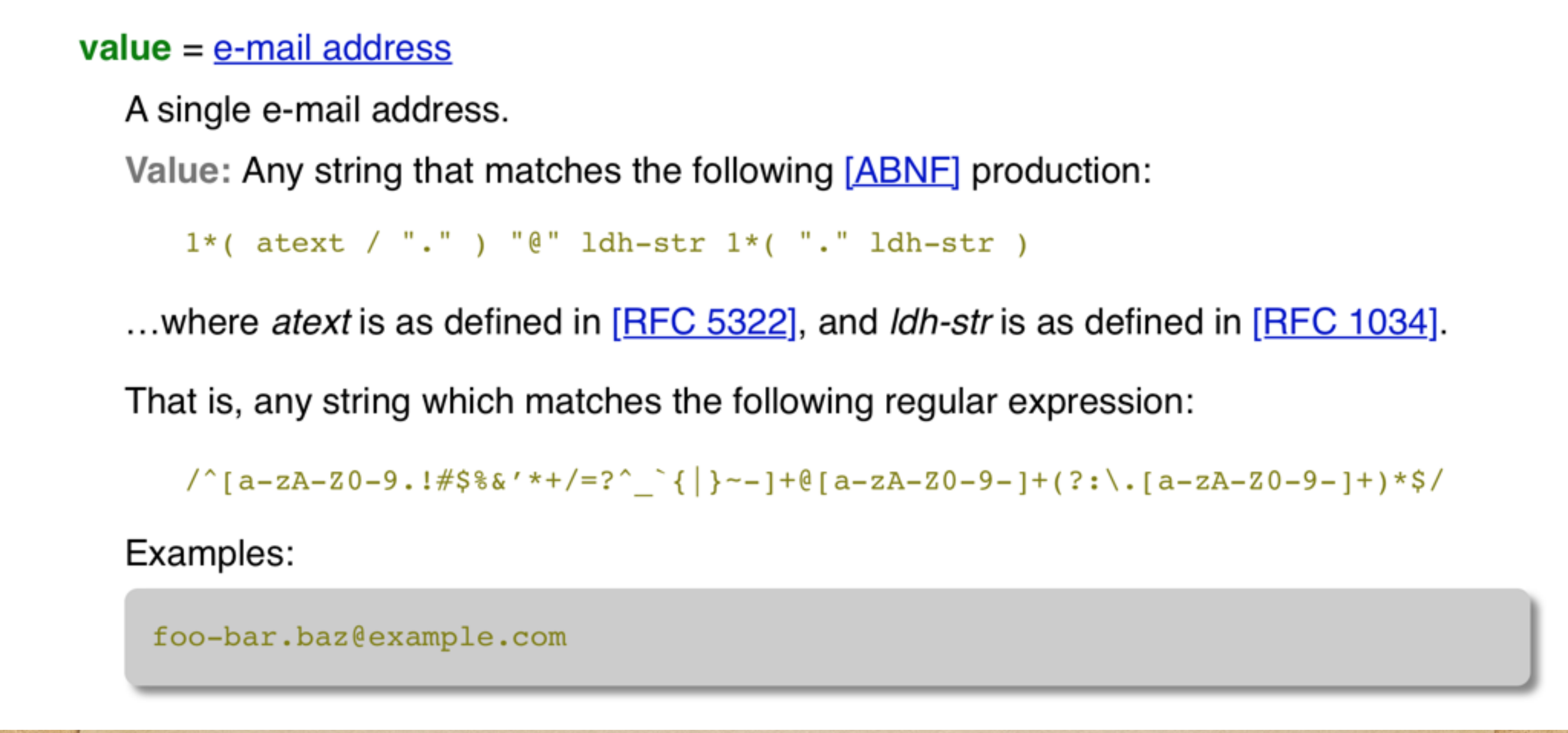
```
<input type="email">
<input type="email" multiple>
```

Live Demo

"   "@[IPv6:2001:db8::1]

user@gmail.com

Firefox	Safari	Safari Mobile	Chrome	Opera	IE
4+	5+	iOS 3.1+	10+	10.6+	10+
✓	~	~	✓	✓	✓



value = e-mail address

A single e-mail address.

Value: Any string that matches the following [ABNF] production:

```
1*( atext / "." ) "@" ldh-str 1*( "." ldh-str )
```

…where *atext* is as defined in [RFC 5322], and *ldh-str* is as defined in [RFC 1034].

That is, any string which matches the following regular expression:

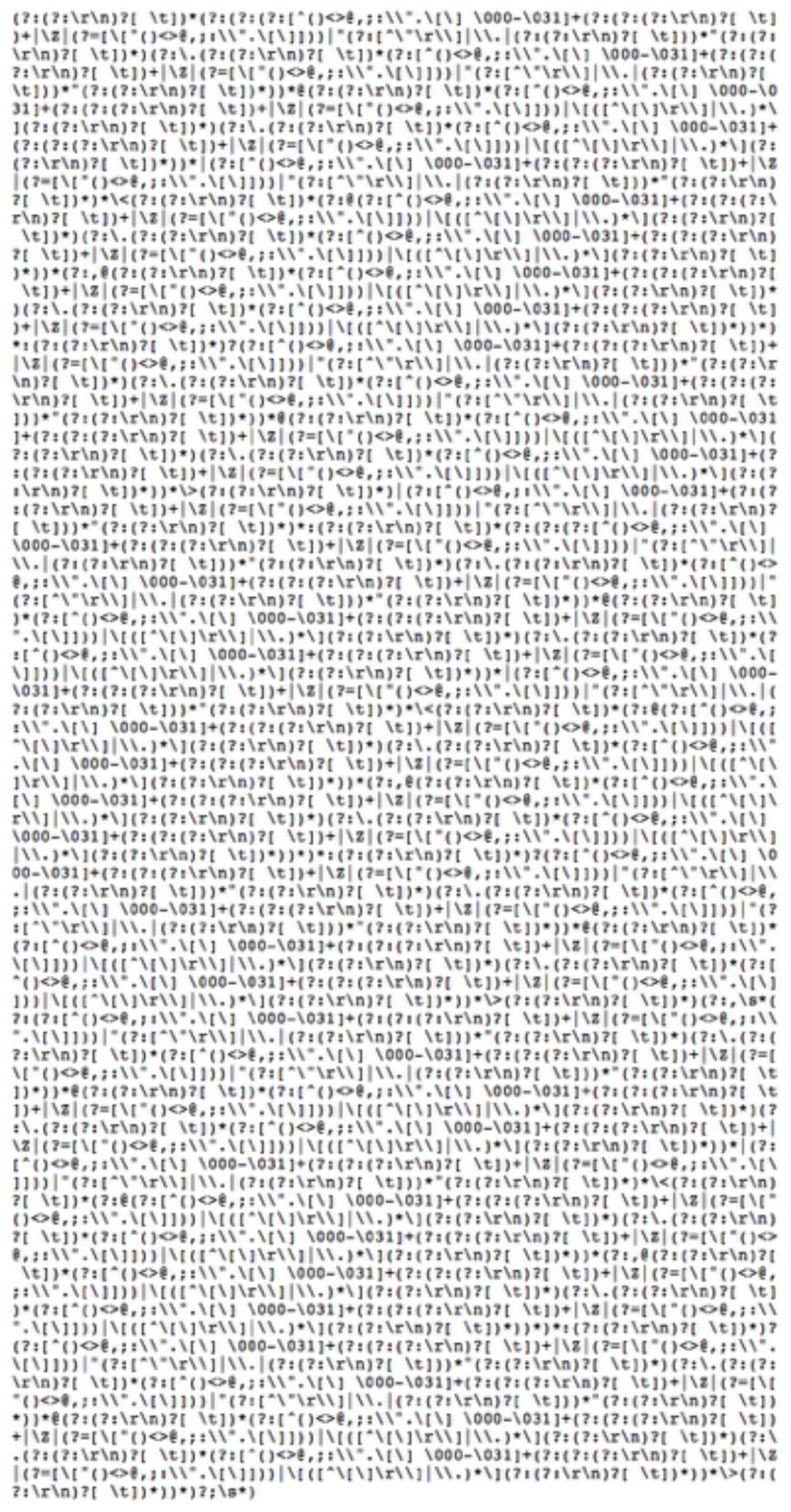
```
/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/
```

Examples:

```
foo-bar.baz@example.com
```

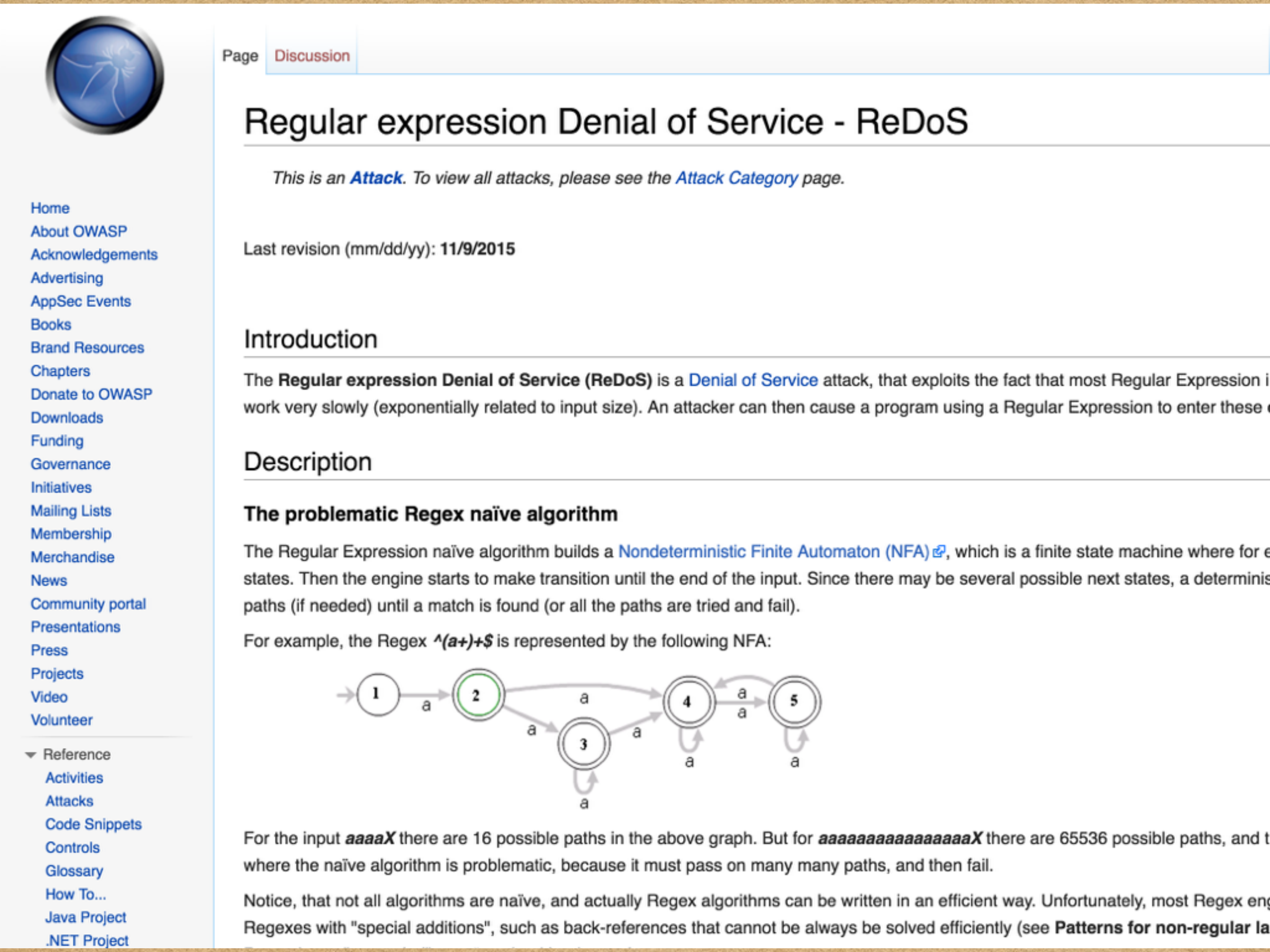
RFC 822, 2822, 5322

- https://tools.ietf.org/html/rfc822 (1982 год)
- https://tools.ietf.org/html/rfc2822 (2001 год)
- https://tools.ietf.org/html/rfc5322 (2008 год)



```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:
\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(
?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[
\t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\0
31]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\
](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:
(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z
|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)
?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\
r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
 \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)
?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t]
)*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
 \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*
)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)
*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+
|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r
\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:
\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t
]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031
]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](
?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?
:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?
:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?
:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?
[ \t]))*"(?:(?:\r\n)?[ \t])*)*:(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\]
\000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|
\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>
@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"
(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t]
)*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?
:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[
\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-
\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(
?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;
:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([
^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\"
.\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\
]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\
[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\
r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\]
\000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]
|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \0
00-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\
.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,
;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?
:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*
(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[
^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]
]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)(?:,\s*(
?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(
?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[
\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t
])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t
])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?
:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|
\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:
[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\
]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)
?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["
()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)
?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>
@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[
 \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,
;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t]
)*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?
(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:
\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\[
"()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])
*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])
+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\
.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z
|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(
?:\r\n)?[ \t])*))*)?;\s*)
```

http://www.ex-parrot.com/~pdw/Mail-RFC822-Address.html



Home
About OWASP
Acknowledgements
Advertising
AppSec Events
Books
Brand Resources
Chapters
Donate to OWASP
Downloads
Funding
Governance
Initiatives
Mailing Lists
Membership
Merchandise
News
Community portal
Presentations
Press
Projects
Video
Volunteer
Reference
Activities
Attacks
Code Snippets
Controls
Glossary
How To...
Java Project
.NET Project

Regular expression Denial of Service - ReDoS

*This is an **Attack**. To view all attacks, please see the Attack Category page.*

Last revision (mm/dd/yy): **11/9/2015**

Introduction

The **Regular expression Denial of Service (ReDoS)** is a Denial of Service attack, that exploits the fact that most Regular Expression i
work very slowly (exponentially related to input size). An attacker can then cause a program using a Regular Expression to enter these

Description

The problematic Regex naïve algorithm

The Regular Expression naïve algorithm builds a Nondeterministic Finite Automaton (NFA), which is a finite state machine where for e
states. Then the engine starts to make transition until the end of the input. Since there may be several possible next states, a determinis
paths (if needed) until a match is found (or all the paths are tried and fail).

For example, the Regex ***^(a+)+$*** is represented by the following NFA:

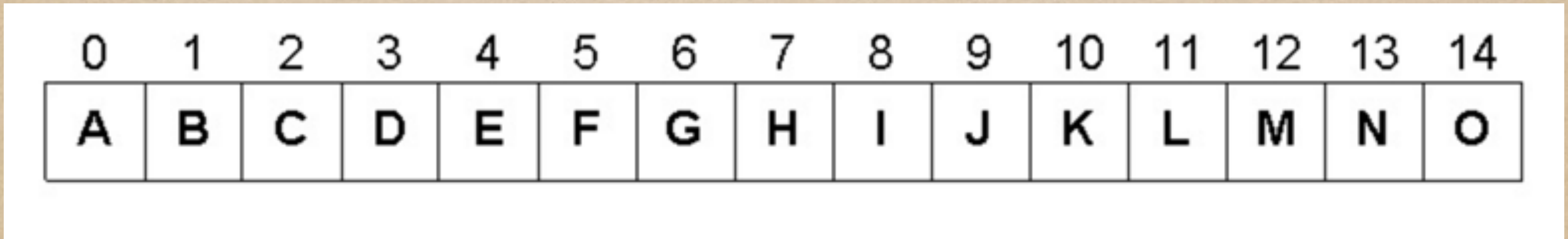
For the input ***aaaaX*** there are 16 possible paths in the above graph. But for ***aaaaaaaaaaaaaaaaX*** there are 65536 possible paths, and t
where the naïve algorithm is problematic, because it must pass on many many paths, and then fail.

Notice, that not all algorithms are naïve, and actually Regex algorithms can be written in an efficient way. Unfortunately, most Regex en
Regexes with "special additions", such as back-references that cannot be always be solved efficiently (see **Patterns for non-regular la**

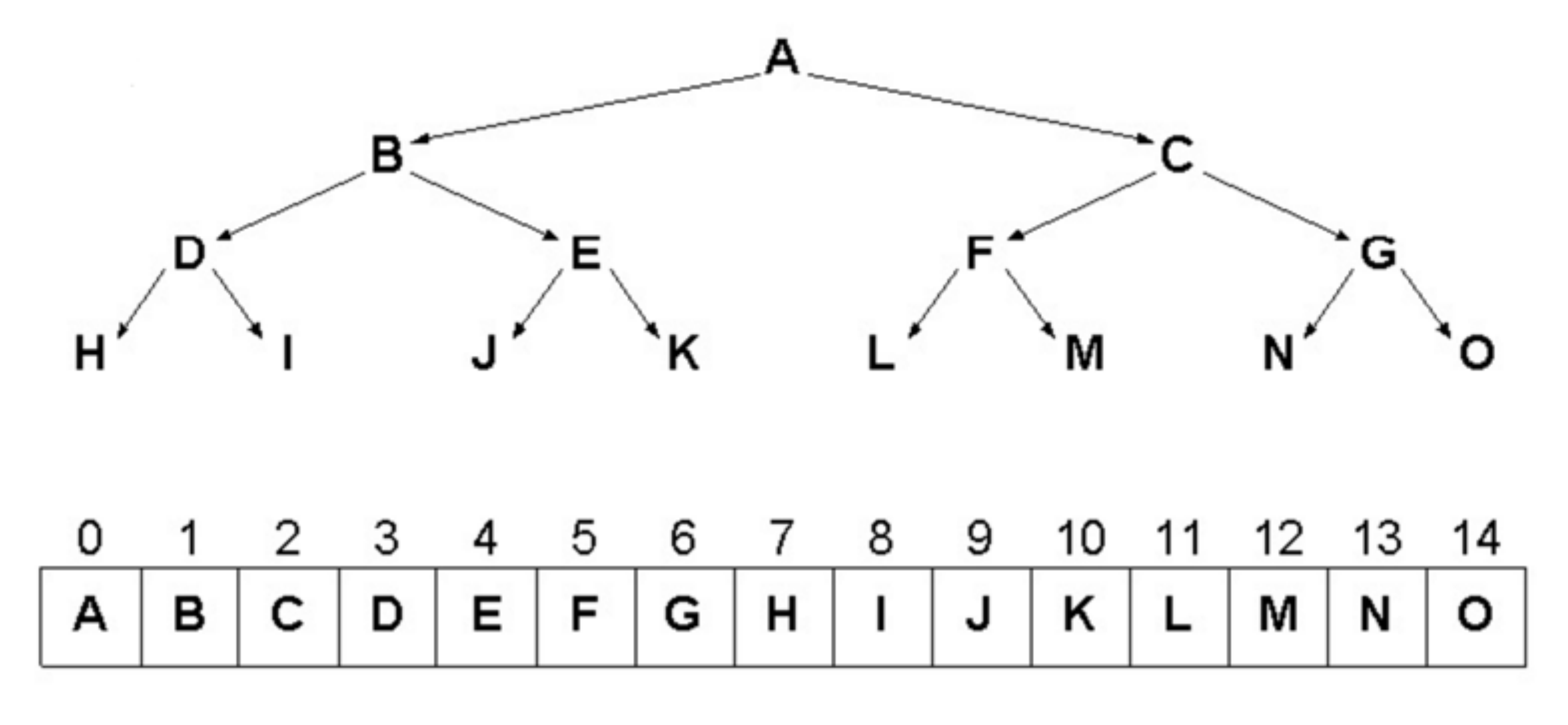
DEMO TIME

Parsing or syntactic analysis is the process of analysing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar

В основе любой сложной задачи стоит неправильно заданный вопрос.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O



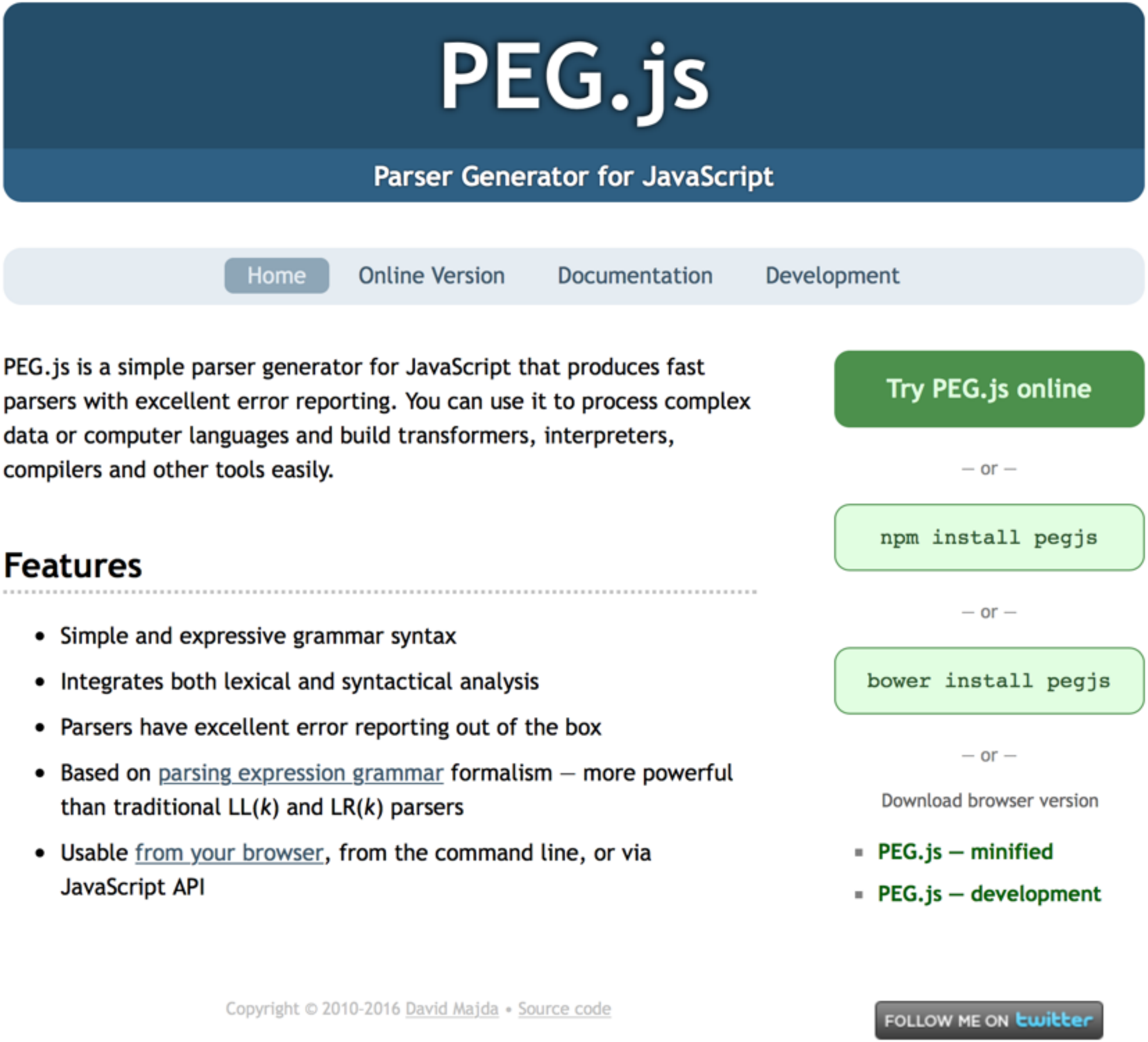
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Что нам нужно?

- Парсер
- Грамматика
- Абстрактная формальная грамматика
- Система описания синтаксиса

CODE TIME

Что дальше?



PEG.js

Parser Generator for JavaScript

PEG.js is a simple parser generator for JavaScript that produces fast parsers with excellent error reporting. You can use it to process complex data or computer languages and build transformers, interpreters, compilers and other tools easily.

Features

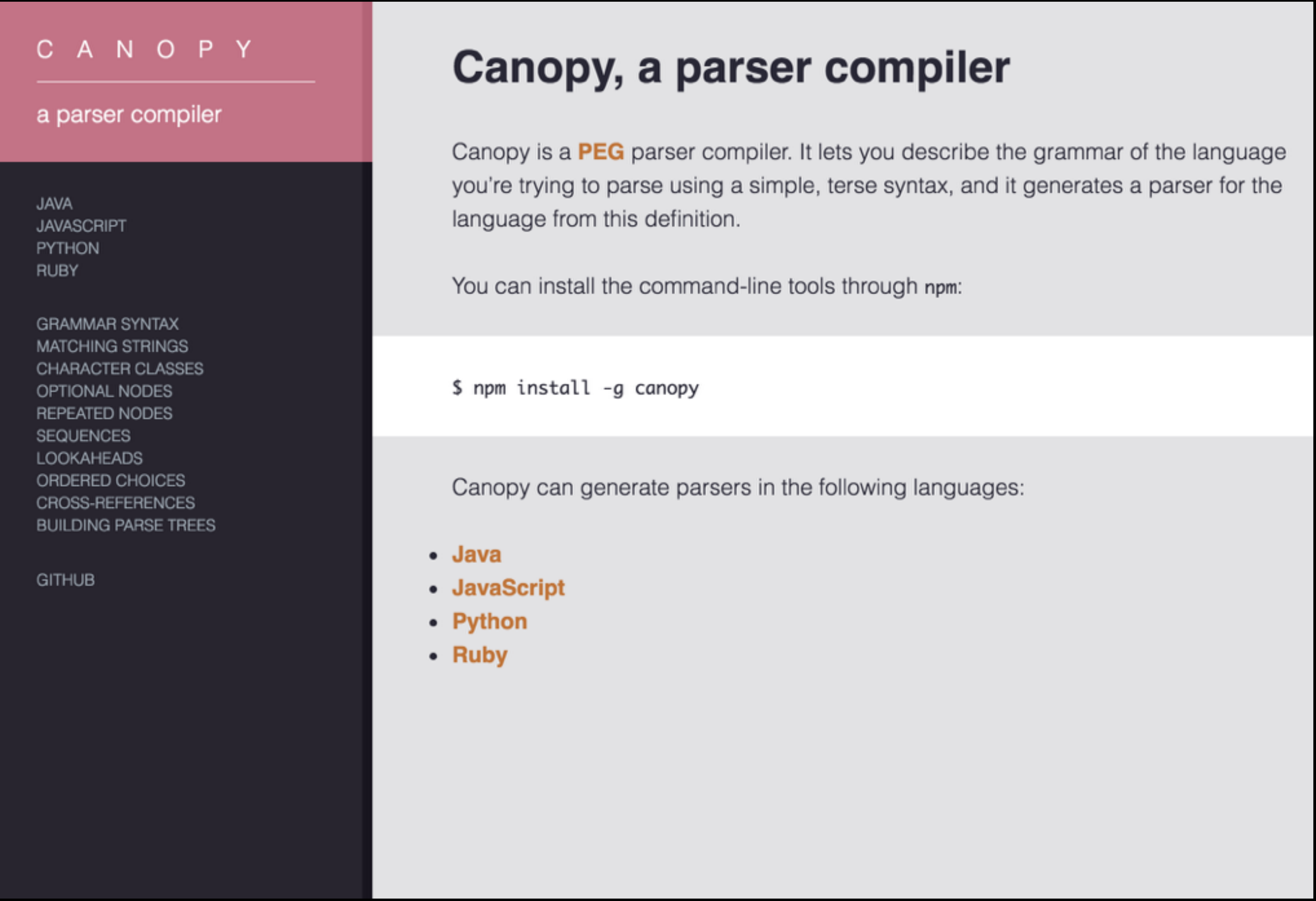
- Simple and expressive grammar syntax
- Integrates both lexical and syntactical analysis
- Parsers have excellent error reporting out of the box
- Based on parsing expression grammar formalism — more powerful than traditional LL(*k*) and LR(*k*) parsers
- Usable from your browser, from the command line, or via JavaScript API

```
npm install pegjs
```

```
bower install pegjs
```

Download browser version

- PEG.js — minified
- PEG.js — development



Canopy, a parser compiler

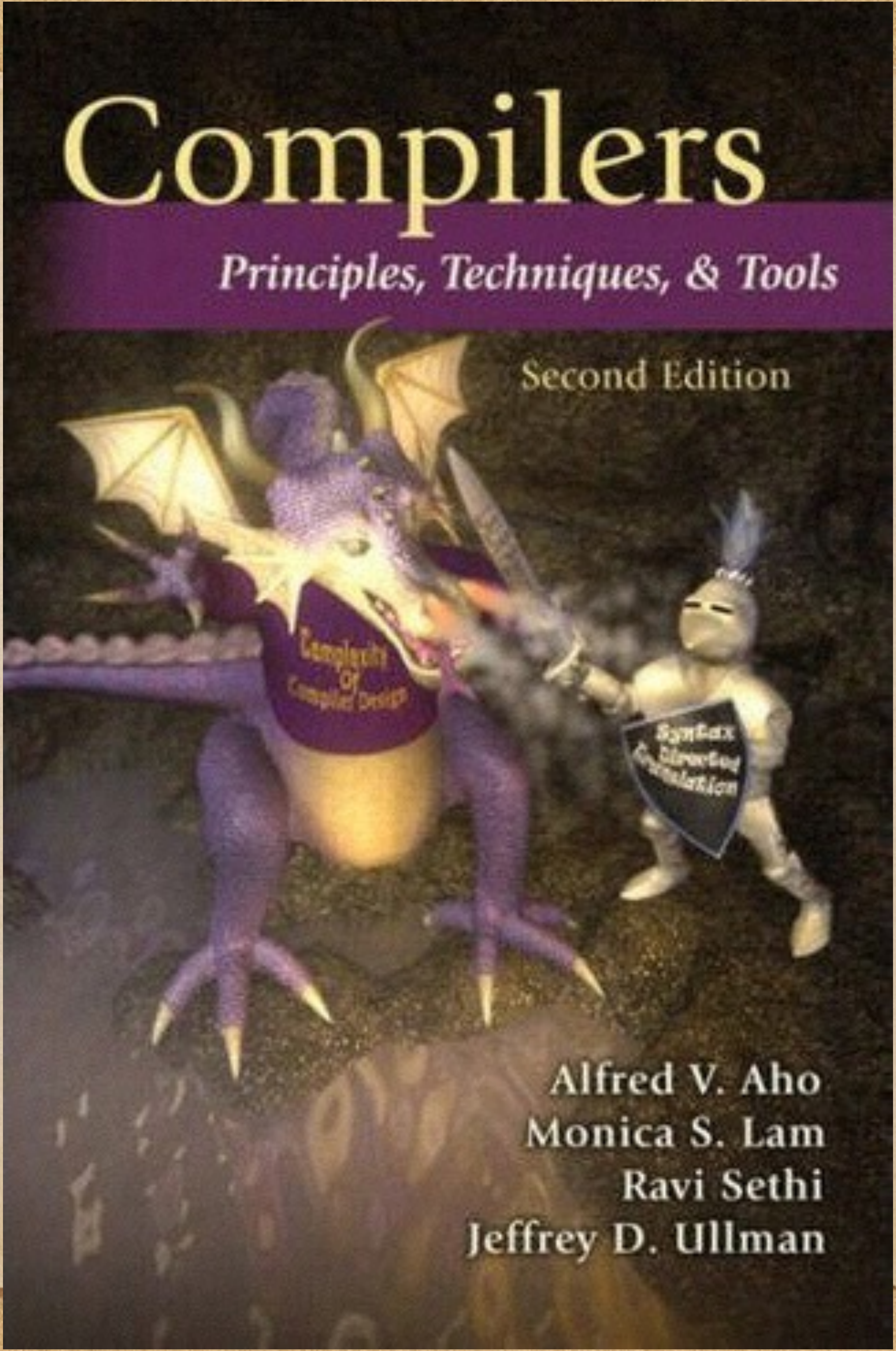
Canopy is a **PEG** parser compiler. It lets you describe the grammar of the language you're trying to parse using a simple, terse syntax, and it generates a parser for the language from this definition.

You can install the command-line tools through `npm`:

```
$ npm install -g canopy
```

Canopy can generate parsers in the following languages:

- **Java**
- **JavaScript**
- **Python**
- **Ruby**

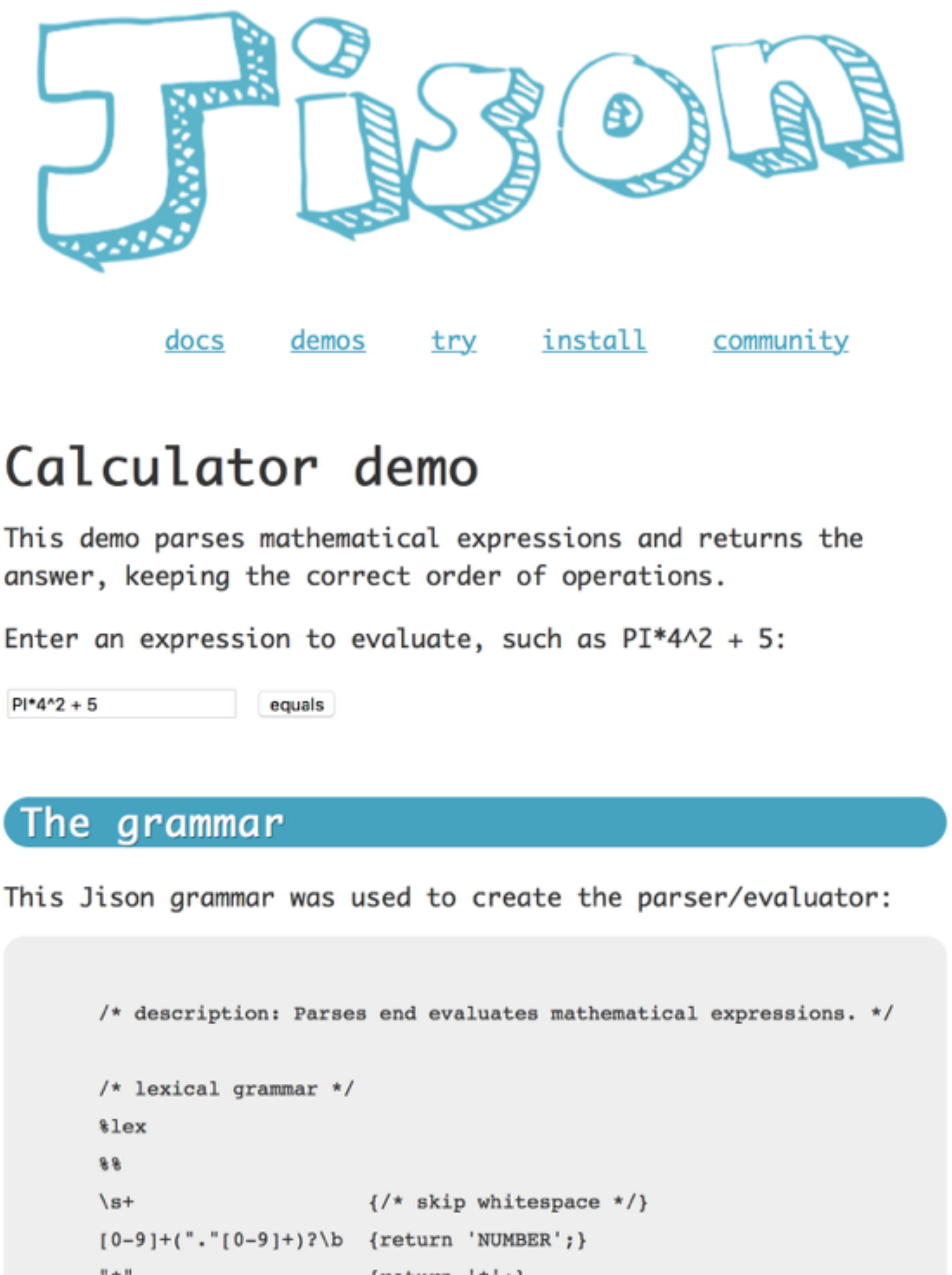


Compilers

Principles, Techniques, & Tools

Second Edition

Alfred V. Aho
Monica S. Lam
Ravi Sethi
Jeffrey D. Ullman



Jison

Calculator demo

This demo parses mathematical expressions and returns the answer, keeping the correct order of operations.

Enter an expression to evaluate, such as PI*4^2 + 5:

PI*4^2 + 5

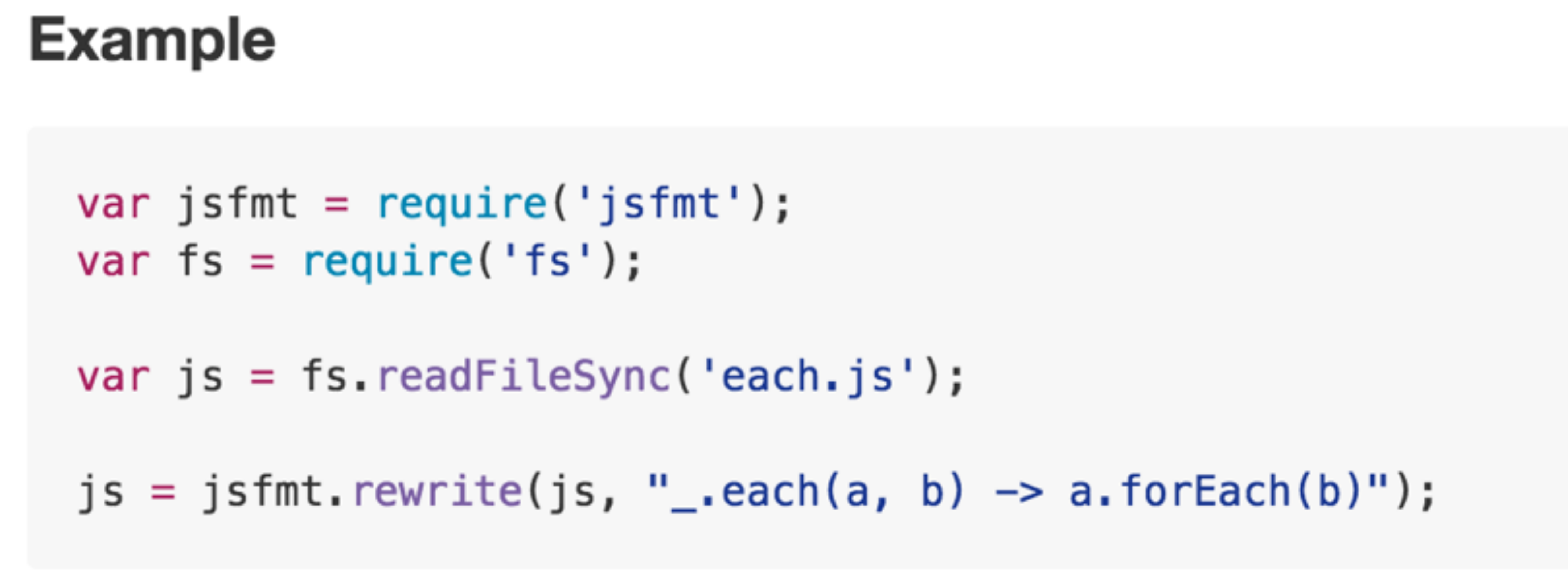
The grammar

This Jison grammar was used to create the parser/evaluator:

```
/* description: Parses end evaluates mathematical expressions. */

/* lexical grammar */
%lex
%%
\s+                   {/* skip whitespace */}
[0-9]+("."[0-9]+)?\b  {return 'NUMBER';}
"*"                   {return '*';}
```

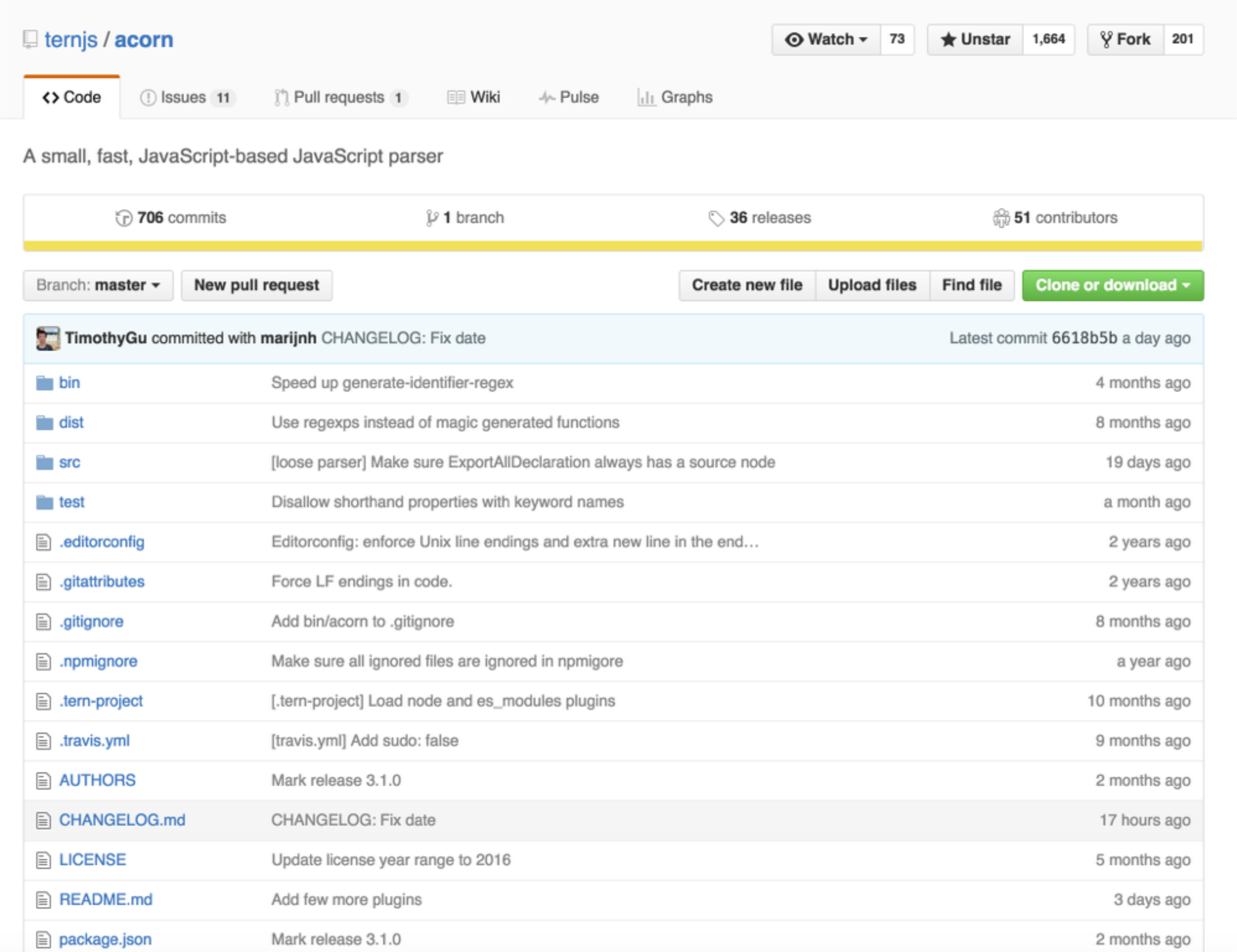
- EsLint
- JSCS
- CSSO

https://github.com/rdio/jsfmt



Example

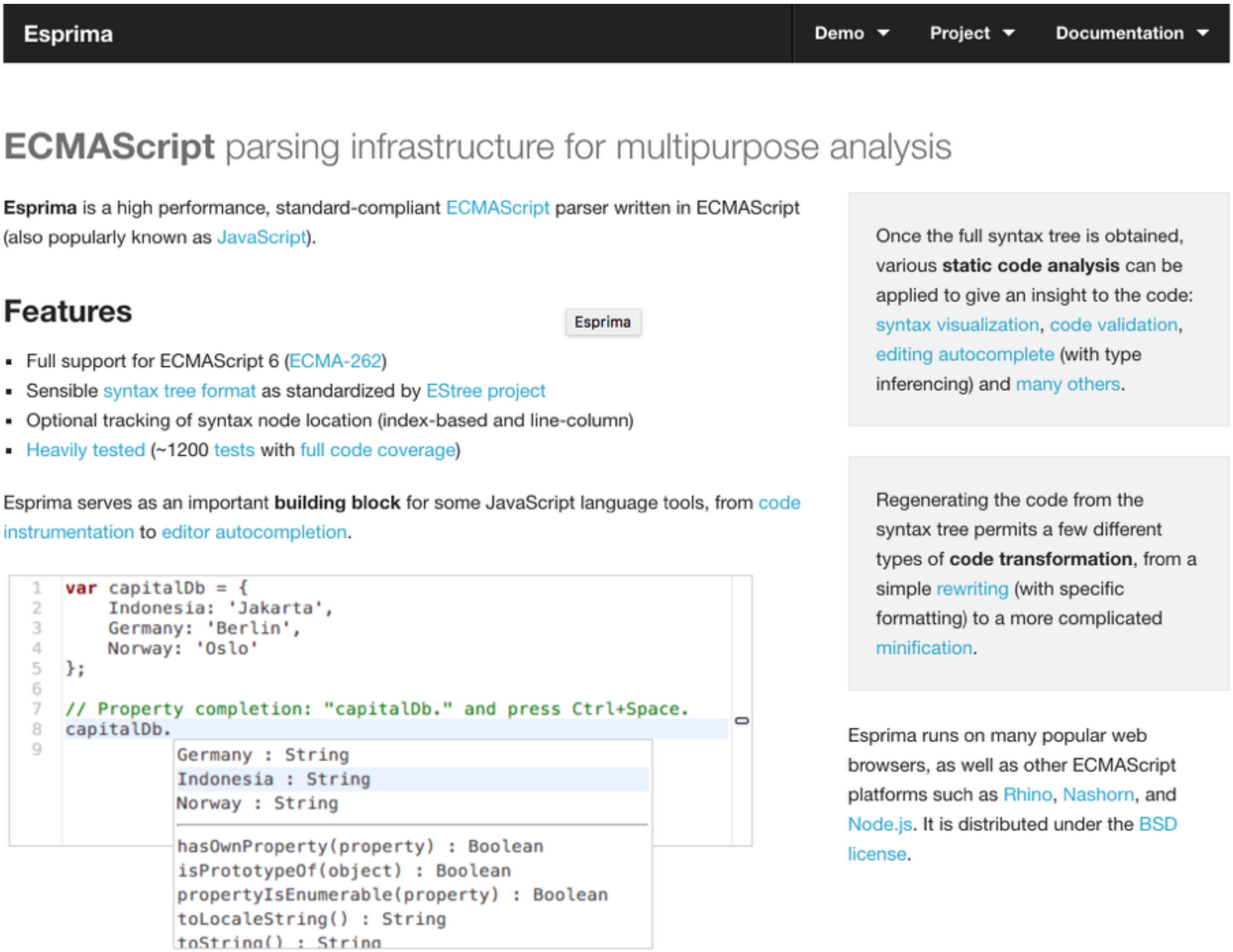
```
var jsfmt = require('jsfmt');
var fs = require('fs');

var js = fs.readFileSync('each.js');

js = jsfmt.rewrite(js, "_.each(a, b) -> a.forEach(b)");
```



A small, fast, JavaScript-based JavaScript parser

TimothyGu committed with marijnh CHANGELOG: Fix date — Latest commit 6618b5b a day ago

bin	Speed up generate-identifier-regex	4 months ago
dist	Use regexps instead of magic generated functions	8 months ago
src	[loose parser] Make sure ExportAllDeclaration always has a source node	19 days ago
test	Disallow shorthand properties with keyword names	a month ago
.editorconfig	Editorconfig: enforce Unix line endings and extra new line in the end…	2 years ago
.gitattributes	Force LF endings in code.	2 years ago
.gitignore	Add bin/acorn to .gitignore	8 months ago
.npmignore	Make sure all ignored files are ignored in npmigore	a year ago
.tern-project	[.tern-project] Load node and es_modules plugins	10 months ago
.travis.yml	[travis.yml] Add sudo: false	9 months ago
AUTHORS	Mark release 3.1.0	2 months ago
CHANGELOG.md	CHANGELOG: Fix date	17 hours ago
LICENSE	Update license year range to 2016	5 months ago
README.md	Add few more plugins	3 days ago
package.json	Mark release 3.1.0	2 months ago



ECMAScript parsing infrastructure for multipurpose analysis

Esprima is a high performance, standard-compliant ECMAScript parser written in ECMAScript (also popularly known as JavaScript).

Features

- Full support for ECMAScript 6 (ECMA-262)
- Sensible syntax tree format as standardized by EStree project
- Optional tracking of syntax node location (index-based and line-column)
- Heavily tested (~1200 tests with full code coverage)

Esprima serves as an important **building block** for some JavaScript language tools, from code instrumentation to editor autocompletion.

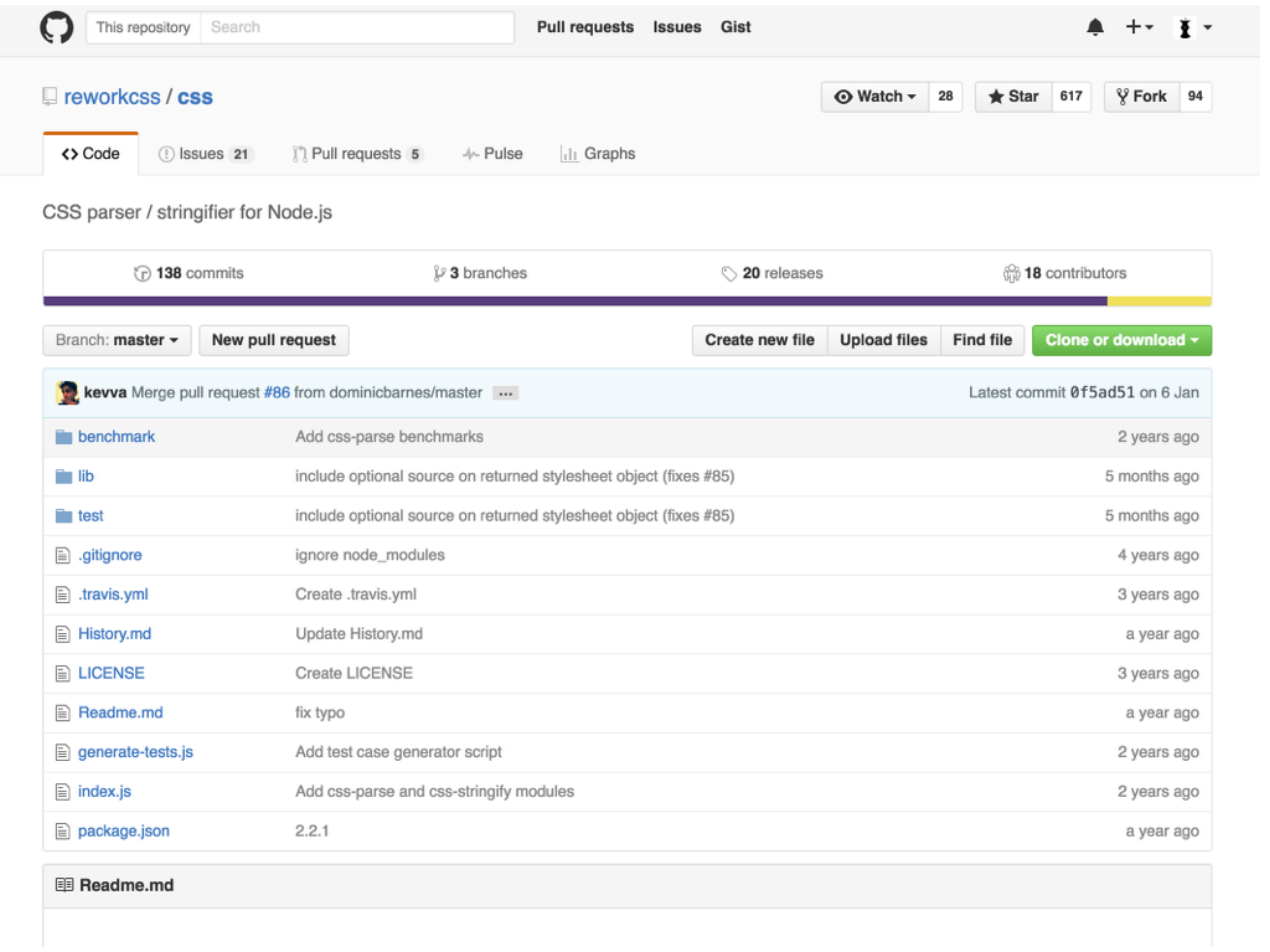
```
var capitalDb = {
    Indonesia: 'Jakarta',
    Germany: 'Berlin',
    Norway: 'Oslo'
};

// Property completion: "capitalDb." and press Ctrl+Space.
capitalDb.
```

```
Germany : String
Indonesia : String
Norway : String
hasOwnProperty(property) : Boolean
isPrototypeOf(object) : Boolean
propertyIsEnumerable(property) : Boolean
toLocaleString() : String
toString() : String
```

Once the full syntax tree is obtained, various **static code analysis** can be applied to give an insight to the code: syntax visualization, code validation, editing autocomplete (with type inferencing) and many others.

Regenerating the code from the syntax tree permits a few different types of **code transformation**, from a simple rewriting (with specific formatting) to a more complicated minification.

Esprima runs on many popular web browsers, as well as other ECMAScript platforms such as Rhino, Nashorn, and Node.js. It is distributed under the BSD license.



reworkcss / css

CSS parser / stringifier for Node.js

138 commits | 3 branches | 20 releases | 18 contributors

kevva Merge pull request #86 from dominicbarnes/master … Latest commit 0f5ad51 on 6 Jan

benchmark	Add css-parse benchmarks	2 years ago
lib	include optional source on returned stylesheet object (fixes #85)	5 months ago
test	include optional source on returned stylesheet object (fixes #85)	5 months ago
.gitignore	ignore node_modules	4 years ago
.travis.yml	Create .travis.yml	3 years ago
History.md	Update History.md	a year ago
LICENSE	Create LICENSE	3 years ago
Readme.md	fix typo	a year ago
generate-tests.js	Add test case generator script	2 years ago
index.js	Add css-parse and css-stringify modules	2 years ago
package.json	2.2.1	a year ago

Readme.md

THE END

Q&A?

Алексей Охрименко

http://bit.ly/1U1Fpa8

Twitter: @Ai_boy