Table 1-1

comment	REGEX	Racket Scheme	Racket #lang PEG	Ohm-JS	ESRAP
			`#lang peg (require peg)`	*name* { … }	
always succeeds		(epsilon)			
matches the character c	/c/	(char c)	‘a’	“a”	#\a
matches any character	/./	(any-char)	.	any	character
match any char between c1 and c2	/[a-z]/	(range #\a #\z)	[a-z]	“a" .. “z”	(character-ranges #\a #\z)
matches string str	/abc/	(string str)	a’ ‘b’ ‘c’	“abc”	“abc”
sequence		(and <rule> ...)	R S T	R S T	(and <rule> …)
prioritized choice	/(ab)|c/	(or <rule> ...)	R / S / T	R / S / T	(or <rule> …)
zero or more char	/a*/	(* #\a)	‘a'*	“a”*	(* #\a)
one or more char	/a+/	(+ #\a)	‘a’+	“a”+	(+ #\a)
zero or one char	/a?/	(? #\a)	‘a’?	“a”?	(? #\a)
		(R)	R	R	(R)
zero or more rule		(* …)	R*	R*	(* …)
one or more rule		(+ …)	R+	R+	(+ …)
zero or one rule		(? ...)	R?	R?	(? ...)
		(capture name <rule>)	x:R	*automatic*	*automatic*
not		(! <rule> ...)	(! R)	~R	(! R)
lookahead		(& <rule>)		&R	(> R)
lookbehind					(< R)
discard the semantic result on matching		(drop <rule> ...)	*(~ R)*	*ignore parameter*	(ignore x)

comment	REGEX	Racket Scheme	Racket #lang PEG	Ohm-JS	ESRAP
define rule		(define-peg/bake R …)		*R = …*	(defrule <name> …)
define rule with semantic action		(define-peg R … Q)	*R <— … —> … ;*	addSemantics (…)	(defrule <name> … …)
tag rule		(define-peg/tag R …)		*manually insert tag in JS*	*manually insert tag in lisp*

run parser		(peg R "…")		function ohm_parse (grammar, text) {
				  var ohm = require ('ohm-js');
				  var parser = ohm.grammar (grammar);
				  var cst = parser.match (text);
				  if (cst.succeeded ()) {
				  return { parser: parser, cst: cst };
				  } else {
				  console.log (parser.trace (text).toString ());
				  throw "glue: Ohm matching failed";
				  }
				}
					(parse '<grammar> <string>)

		manual	*manual*	https://ohmlang.github.io/editor/	*manual*