# R-assign (rightward-assignment) operator

## Description

From [https://bugs.ruby-lang.org/issues/15799#change-78465](https://bugs.ruby-lang.org/issues/15799#change-78465), proposal of the rightward-assignment operator by =>.

```shell
$ ./ruby -v -e '(1..).lazy.map {|x| x*2} => x' -e 'p x.first(10)'
ruby 2.7.0dev (2019-06-12T06:32:32Z feature/rassgn-assoc c928f06b79) [x86_64-darwin18]
last_commit=Rightward-assign by ASSOC
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

```

## Related issues:

- Related to Ruby master - Feature #15799: pipeline operator - Closed

## History

### #1 - 06/14/2019 03:08 AM - nobu (Nobuyoshi Nakada)
- Related to Feature #15799: pipeline operator added

### #2 - 06/14/2019 10:49 AM - Hanmac (Hans Mackowiak)

> where does the rightward assign works and where it is blocked? y => x might be treated as Hash Parameter

```ruby
like m y => x is this m(y) => x or still m({y => x})
```

### #3 - 06/14/2019 12:27 PM - nobu (Nobuyoshi Nakada)

This has lower precedence, so the latter.

### #4 - 06/16/2019 12:55 AM - ioquatix (Samuel Williams)

There are two areas where I think this is a great addition:

```ruby
x = if foo
  bar
else
  baz
end
if foo
  bar
else
  baz
end => x
```

I prefer the latter, because it avoids messing with the indentation/readability of the if expression.

Additionally, sometimes I find using irb I have made very large expression. In terminal, going to start of line isn't always obvious/easy. So, I wish to save expression, usually I just press enter and then write x = _ to save last result. But when I go back in history to execute statement again, I must make same "hack". So, I wish I can just write:

```ruby
very long query to get list of users => users
```

That way I don't need to think so hard or go back to start of statement. It might also be nice in middle of expressions, e.g.

```ruby
Users.where(active: true) => active_users.where(type: "admin") => admin_users
```

I don't know if such usage is possible or anticipated, I just wanted to show some ideas - for long expressions sometimes I want to check the middle of the expression.
If it's not clear, previous statement is evaluated like:

```ruby
(Users.where(active: true) => active_users).where(type: "admin") => admin_users
```

If it's not clear, previous statement is evaluated like:

```ruby
(Users.where(active: true) => active_users).where(type: "admin") => admin_users
```

It can't be higher precedence than ., or it will conflict with other syntaxes too much. Rather it should be interpreted like as:

```ruby
admin_users = (active_users.where(type: "admin") = Users.where(active: true))
```

Though it is a syntax error at the parenthesis after where currently.

- Assignee set to matz (Yukihiro Matsumoto)

You may be able to use |> here.

```ruby
Users.where(active: true) => active_users | where(type: "admin") => admin_users
```

I think => is okay, but in case we want to use a keyword (ordinary word) for this feature, I think as would be good. as in SQL is similar to rightward assignment.

```ruby
{l..}.lazy.map {|x| x**2} as x
p x.first(10)
```