Wavy dot **~.** ***Tildot***

Syntax for promise pipelining

Mark S. Miller, Michael Fig — Agoric
Chip Morningstar — Evernote

tc39 October 2019


Y U NO PROXY?

p = b ~. foo —> promise for eventual get

p = b ~. foo(c) —> promise for eventual call

void b ~. foo(c); —> oneway eventual call

p = b ~. foo —> promise for eventual get

p = b ~. foo(c) —> promise for eventual call

void b ~. foo(c); —> oneway eventual call

R(b) —> use proxy instead of syntax

R(b).foo —> promise or function (proxy)?

p = b ~. foo —> promise for eventual get

p = b ~. foo(c) —> promise for eventual call

void b ~. foo(c); —> oneway eventual call

R(b) —> use proxy instead of syntax

R(b).foo —> promise or function (proxy)?

R(R(b).foo)(c) —> extra bookkeeping

p = b ~. foo —> promise for eventual get

p = b ~. foo(c) —> promise for eventual call

void b ~. foo(c); —> oneway eventual call

R(b) —> use proxy instead of syntax

R(b).foo —> promise or function (proxy)?

R(R(b).foo)(c) —> extra bookkeeping

E(b).foo(c)

p = b ~. foo —> promise for eventual get

p = b ~. foo(c) —> promise for eventual call

void b ~. foo(c); —> oneway eventual call

R(b) —> use proxy instead of syntax

R(b).foo —> promise or function (proxy)?

R(R(b).foo)(c) —> extra bookkeeping

void E(b).foo(c); —> extra bookkeeping

Still experimenting with multiple proxies

Some are usable.
None are pleasant.
Still looking.

Syntax	Internal Method
`p ~. name`	`p.[[GetSend]]('name')`
`p ~. name = value`	`p.[[SetSend]]('name', value)`
`delete p ~. name`	`p.[[DeleteSend]]('name')`
`p ~. (...args)`	`p.[[ApplySend]](args)`
`p ~. name(...args)`	`p.[[ApplyMethodSend]]('name', args)`
`p ~. [prop]`	`p.[[GetSend]](prop)`
`p ~. [prop] = value`	`p.[[SetSend]](prop, value)`
`delete p ~. [prop]`	`p.[[DeleteSend]](prop)`
`p ~. [prop](...args)`	`p.[[ApplyMethodSend]](prop, args)`

Syntax	Internal Method
`p ~. name(...args)`	`p.[[ApplyMethodSend]]('name', args)`

void p ~. name(…args); **p.[[ApplyMethodSendOnly]](‘name’, args)**

```
Expression : ...
     Expression ~. [ Expression ] Arguments    // eventual post
     Expression ~. Arguments                   // eventual post
     Expression ~. [ Expression ]              // eventual get
     Expression ~. [ Expression ] = Expression // eventual put
     delete Expression ~. [ Expression ]       // eventual delete
```

```
WavyDot ::
    ~. [lookahead ∉ DecimalDigit]

MemberExpression : ...
    MemberExpression WavyDot [ Expression ]
    MemberExpression WavyDot IdentifierName
CallExpression : ...
    CallExpression WavyDot [ Expression ] Arguments
    CallExpression WavyDot IdentifierName Arguments
    MemberExpression WavyDot Arguments
    CallExpression WavyDot Arguments
    CallExpression WavyDot [ Expression ]
    CallExpression WavyDot IdentifierName
UnaryExpression : ...
    delete CallExpression WavyDot [ Expression ]
    delete CallExpression WavyDot IdentifierName
LeftHandSideExpression :
    Identifier
    CallExpression [ Expression ]
    CallExpression . IdentifierName
    CallExpression WavyDot [ Expression ]
    CallExpression WavyDot IdentifierName
```

Questions?