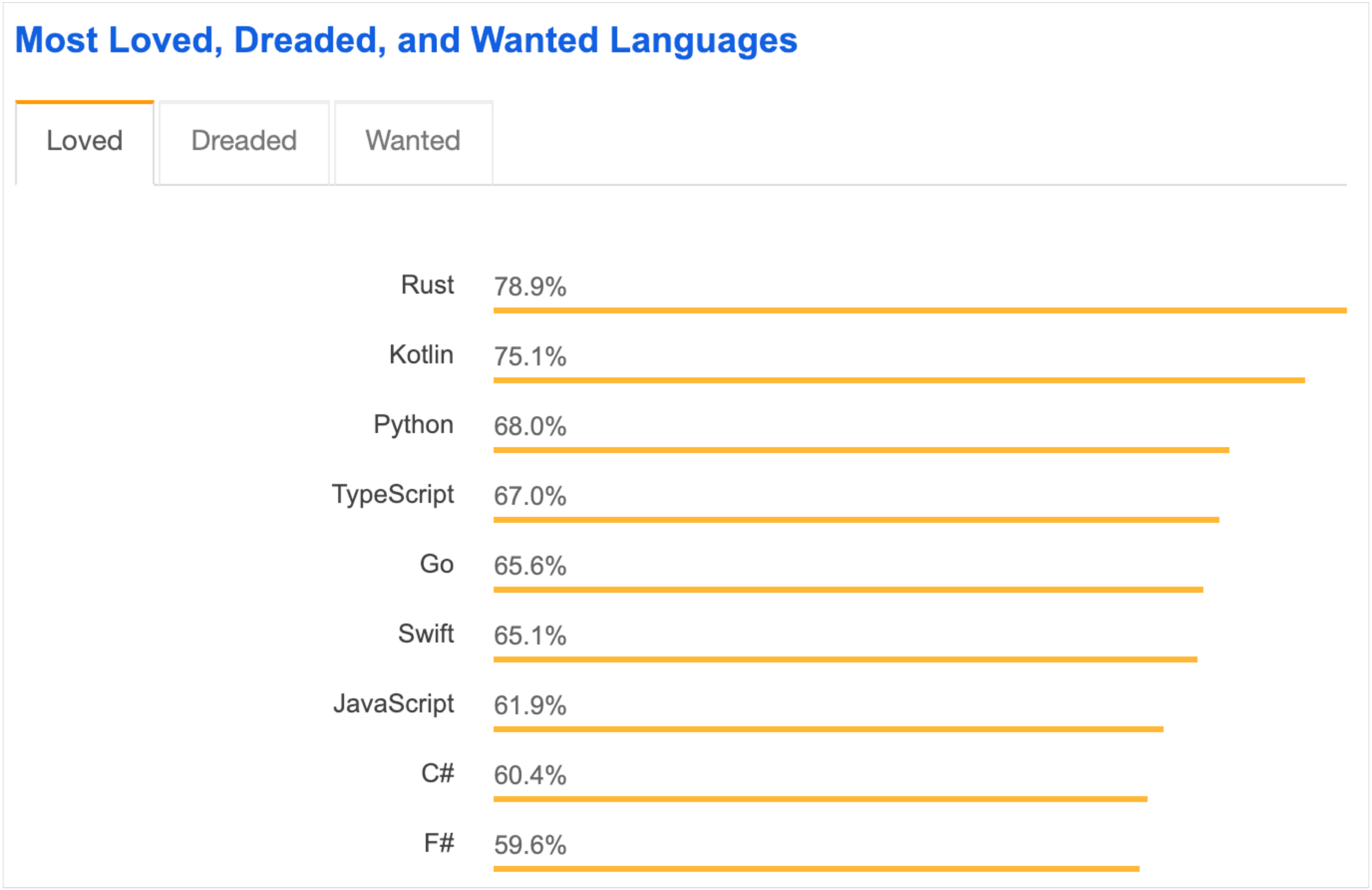
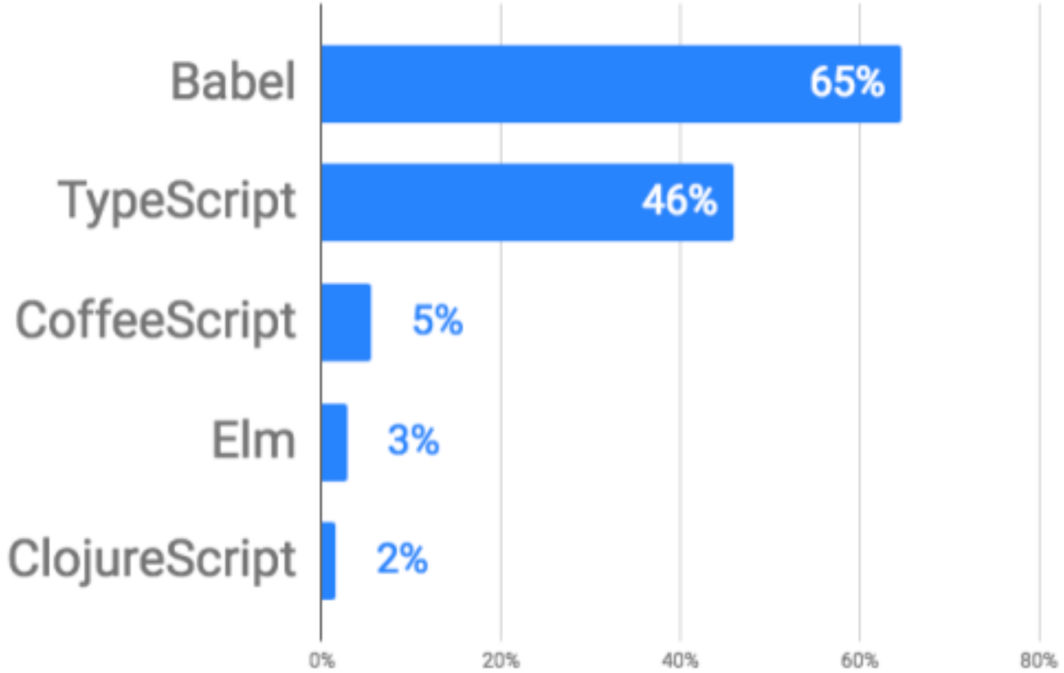
Advanced TypeScript Tooling at Scale

@felixfbecker



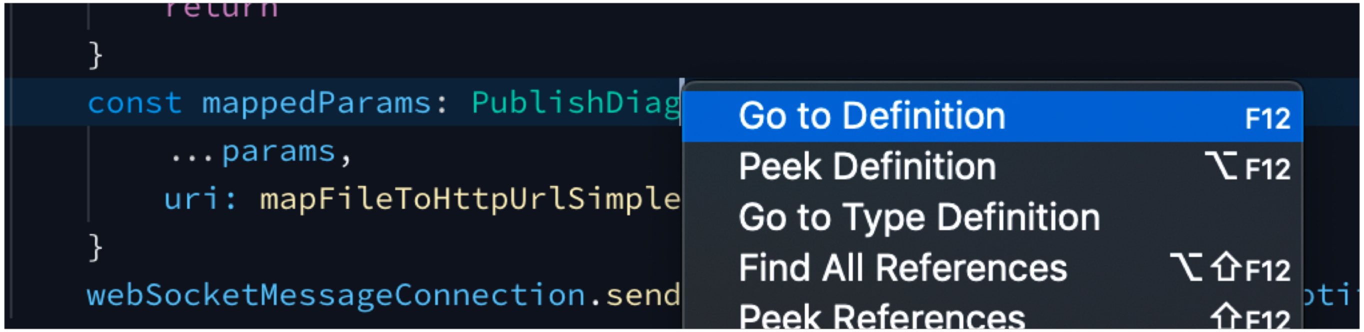
StackOverflow developer survey 2018



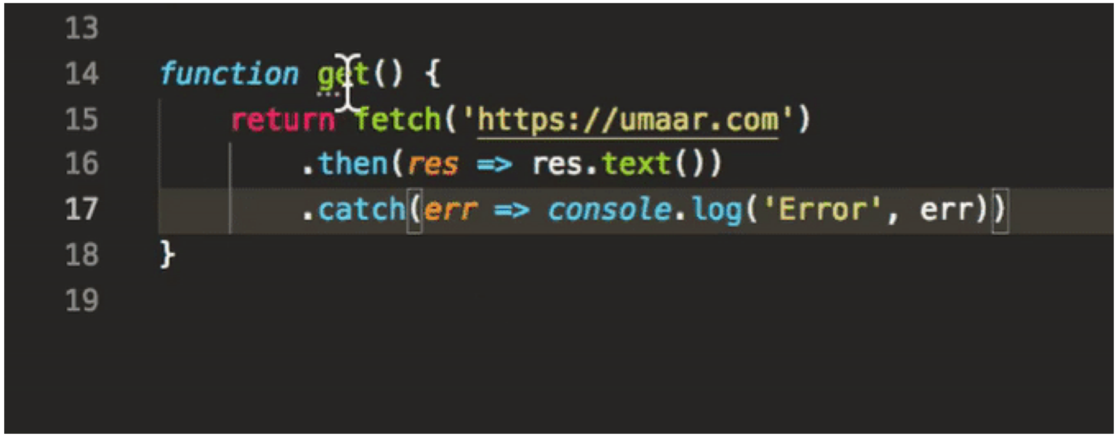
% of npm users using a transpiler,
npm developer survey 2018

What makes TypeScript great?

Tooling

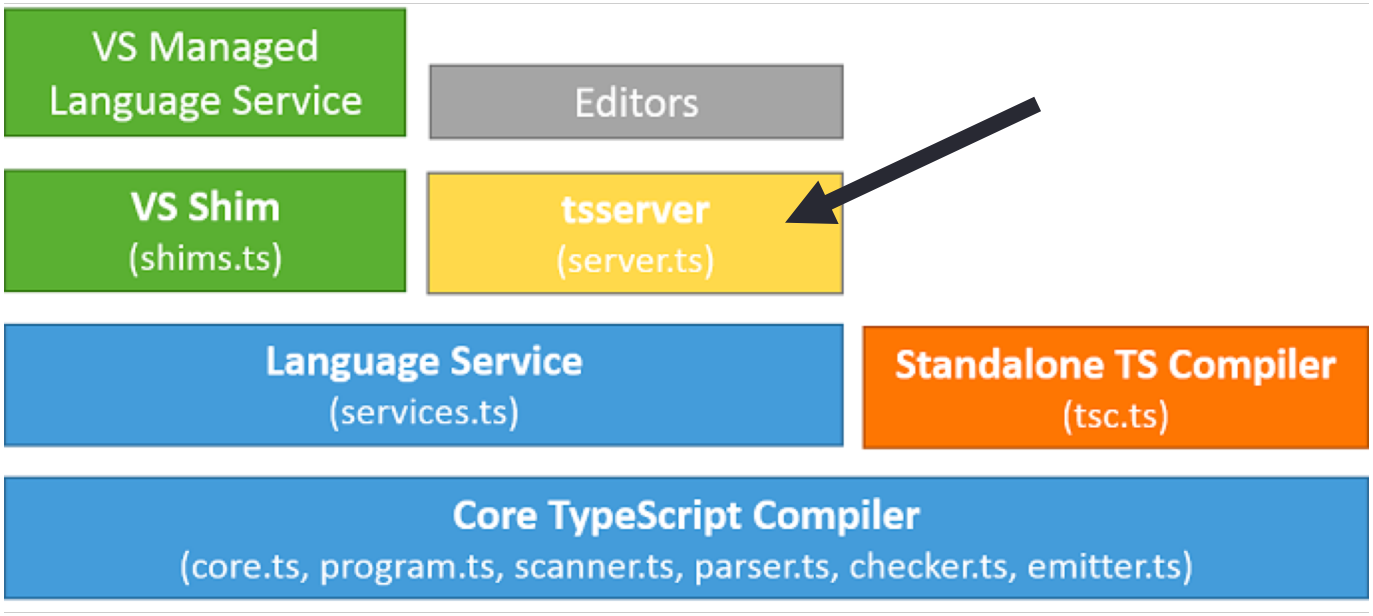


```
    return
}
const mappedParams: PublishDiag
    ...params,
    uri: mapFileToHttpUrlSimple
}
webSocketMessageConnection.send
```



```

function get() {
    return fetch('https://umaar.com')
        .then(res => res.text())
        .catch(err => console.log('Error', err))
}

```

TypeScript architecture



VS Managed Language Service

Editors

VS Shim
(shims.ts)

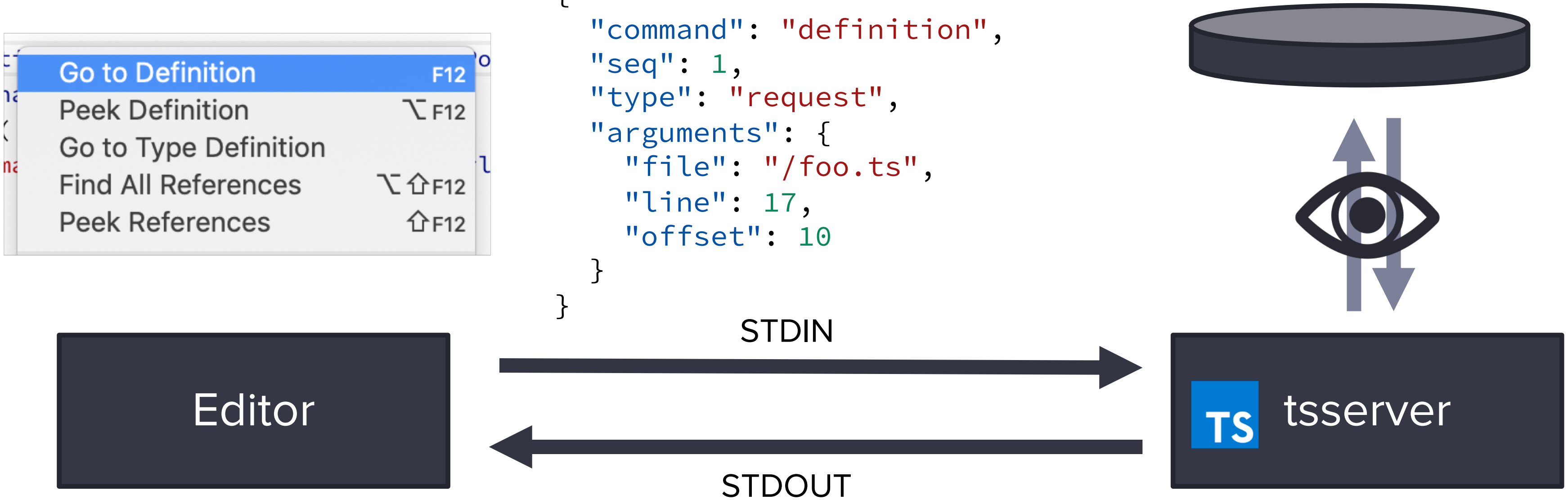
tsserver
(server.ts)

Language Service
(services.ts)

Standalone TS Compiler
(tsc.ts)

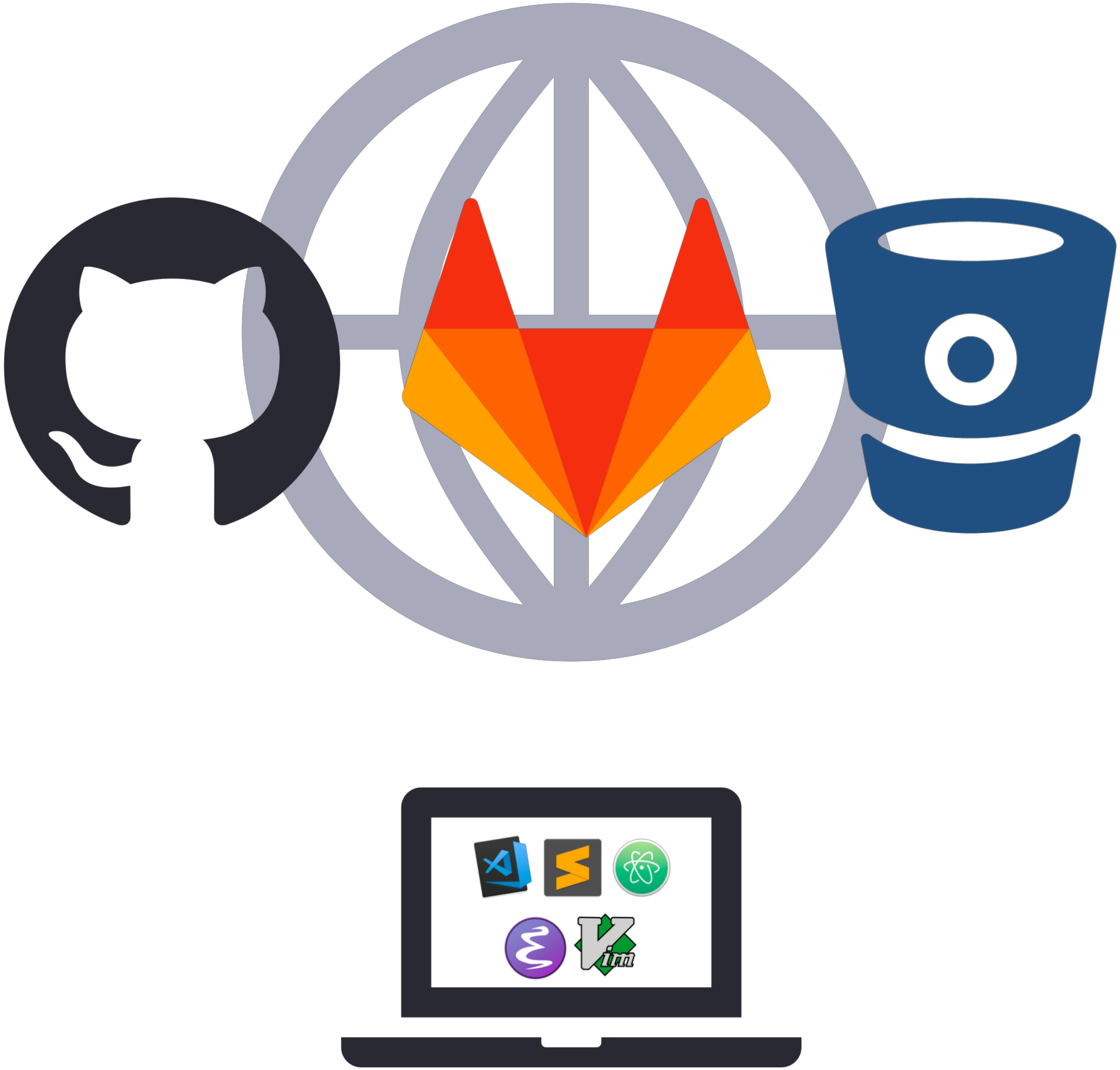
Core TypeScript Compiler
(core.ts, program.ts, scanner.ts, parser.ts, checker.ts, emitter.ts)

TSServer protocol

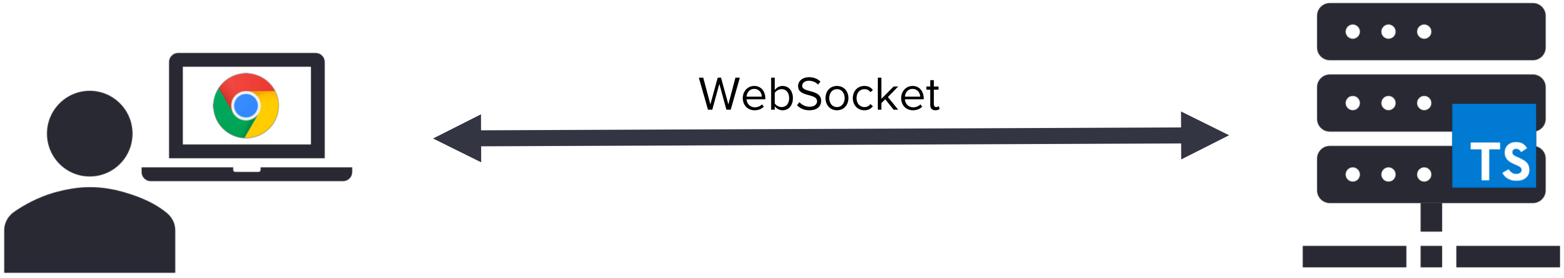


```
{
  "command": "definition",
  "seq": 1,
  "type": "request",
  "arguments": {
    "file": "/foo.ts",
    "line": 17,
    "offset": 10
  }
}
```

```
{
  "seq": 1,
  "type": "response",
  "command": "definition",
  "request_seq": 6,
  "success": true,
  "body": [
    {
      "file": "/bar.ts",
      "start": { "line": 17, "offset": 10 },
      "end": { "line": 17, "offset": 16 }
    }
  ]
}
```

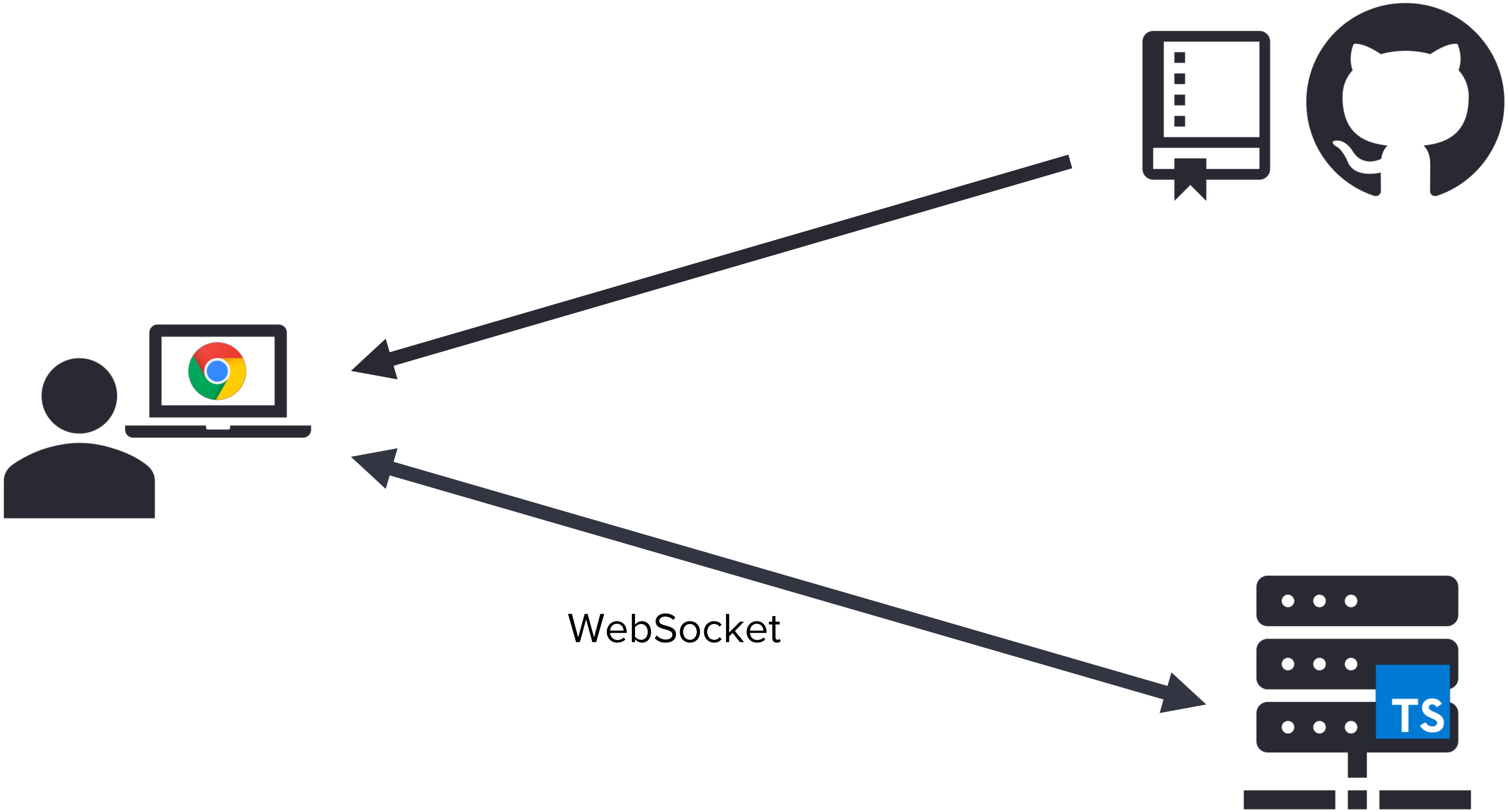

Vim

We can do that!



WebSocket

wss://typescript.sourcegraph.com

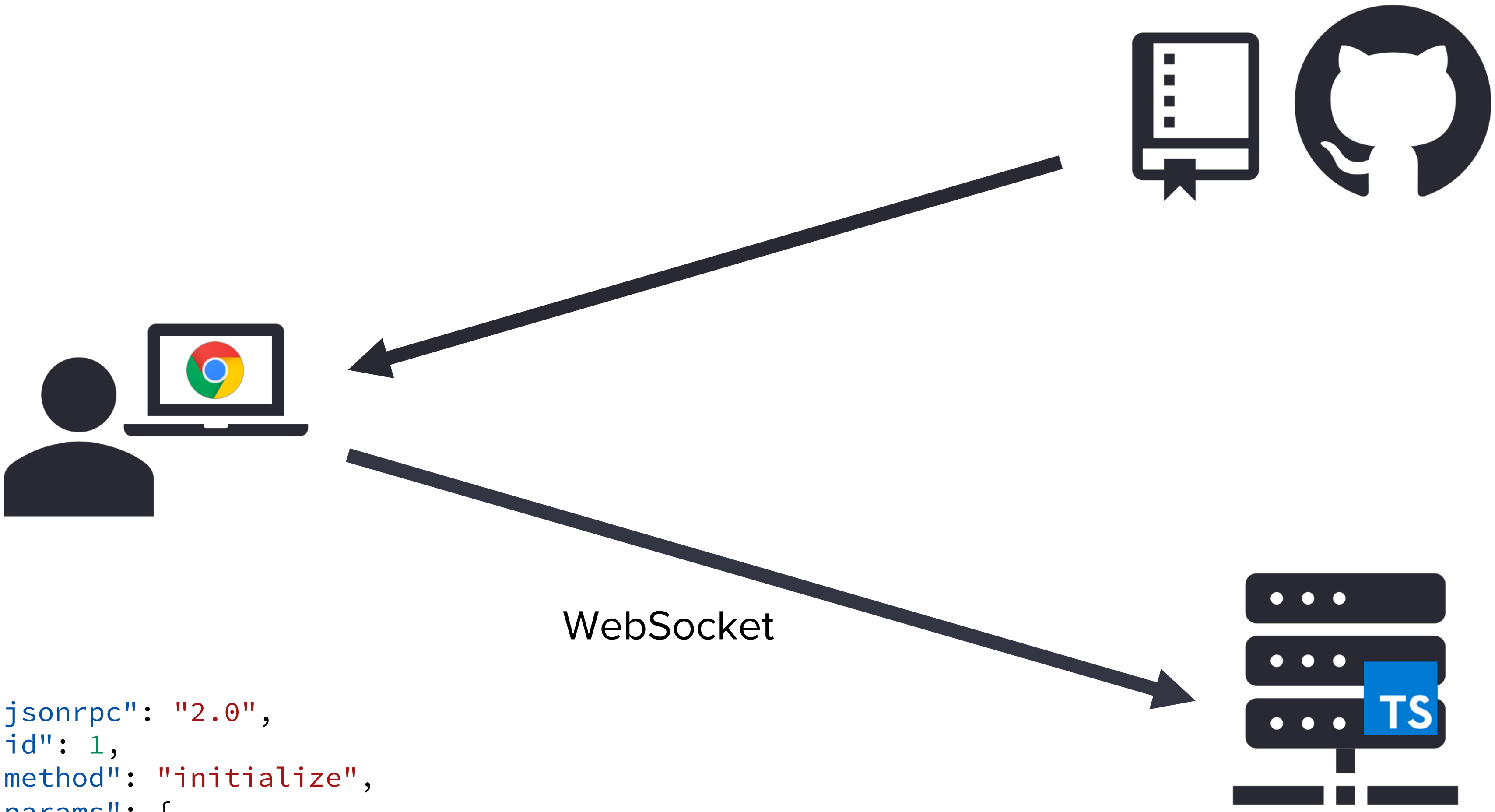

WebSocket
TS

LSP vs TSServer protocol

```
{
  "command": "definition",
  "seq": 1,
  "type": "request",
  "arguments": {
    "file": "/foo.ts",
    "line": 17,
    "offset": 10
  }
}
```

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "textDocument/definition",
  "params": {
    "textDocument": { "uri": "file:///foo.ts" },
    "position": {
      "line": 17,
      "character": 10
    }
  }
}
```

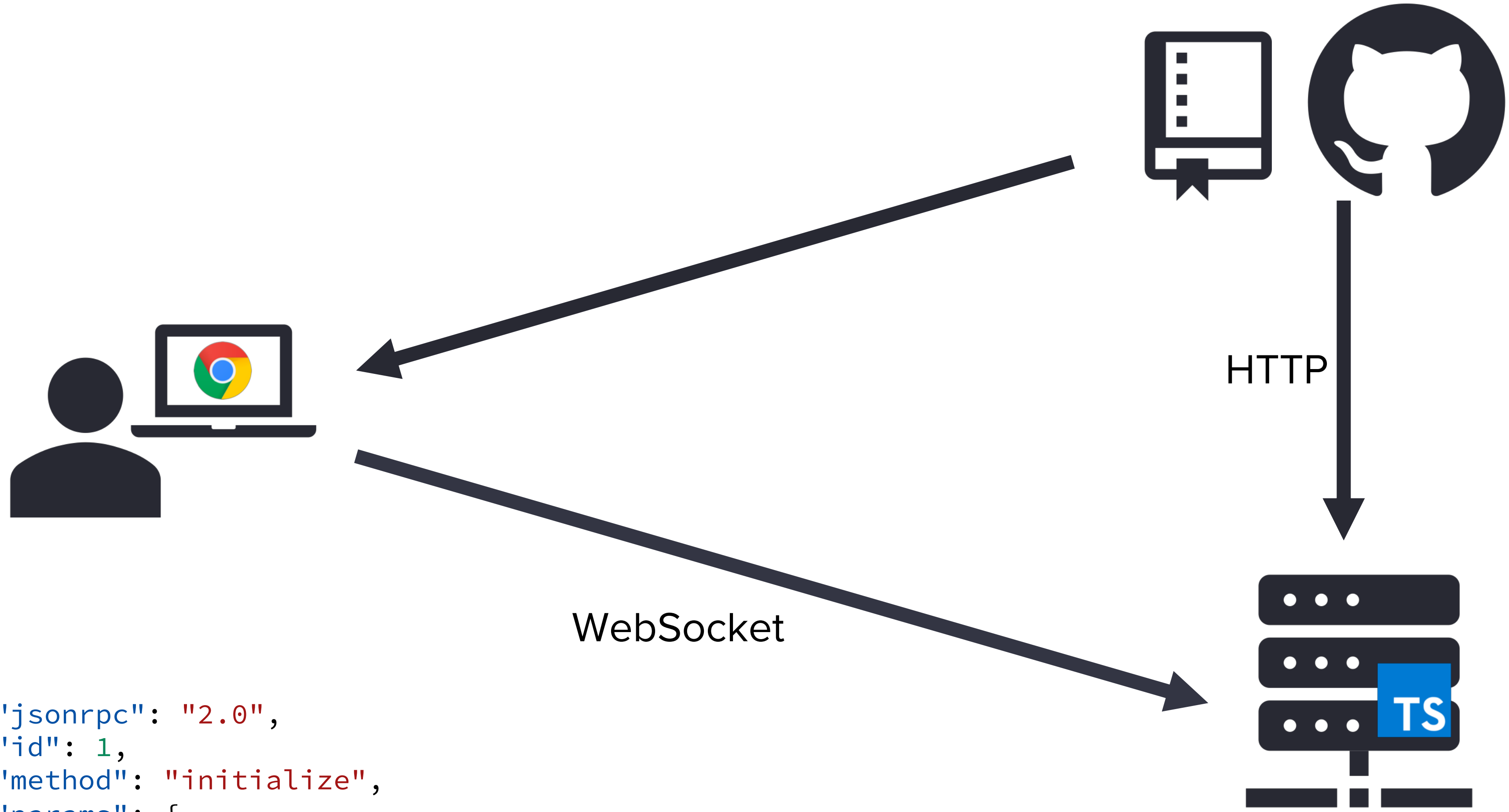
- Almost the same!
- Except LSP
  - Is language-agnostic
  - Uses the JSON-RPC standard
  - Has simpler request cancellation
  - **Uses URIs instead of file paths**



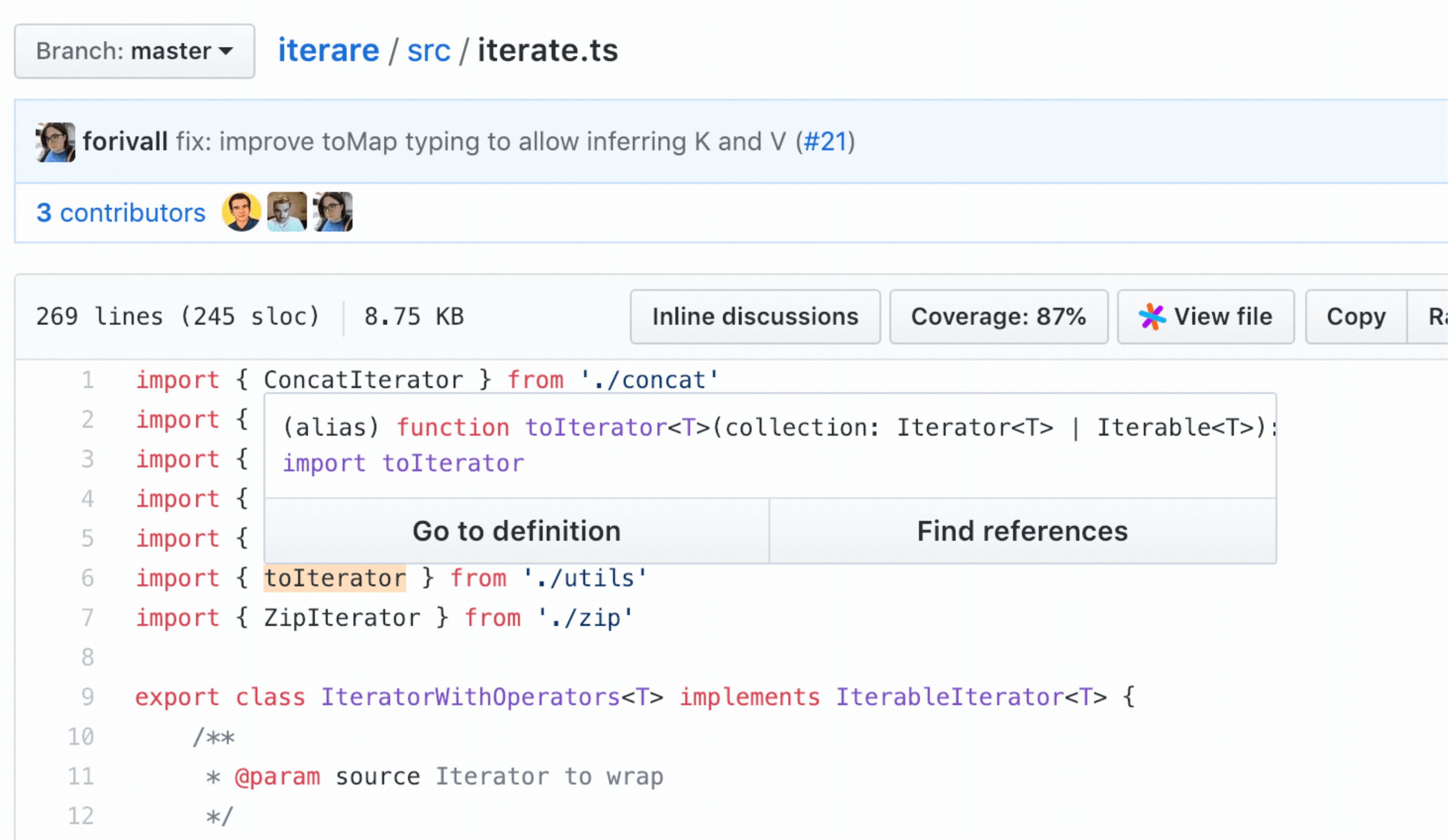
```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {
    "rootUri": "https://sourcegraph.com/github.com/nestjs/nest/-/raw/"
  }
}
```

HTTP root URLs

```
GET https://sourcegraph.com/github.com/nestjs/nest/-/raw/packages/core/index.ts
HEAD https://sourcegraph.com/github.com/nestjs/nest/-/raw/packages/core/notexist.ts
GET https://sourcegraph.com/github.com/nestjs/nest/-/raw/
Accept: application/x-tar
```



```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {
    "rootUri": "https://sourcegraph.com/github.com/nestjs/nest/-/raw/"
  }
}
```

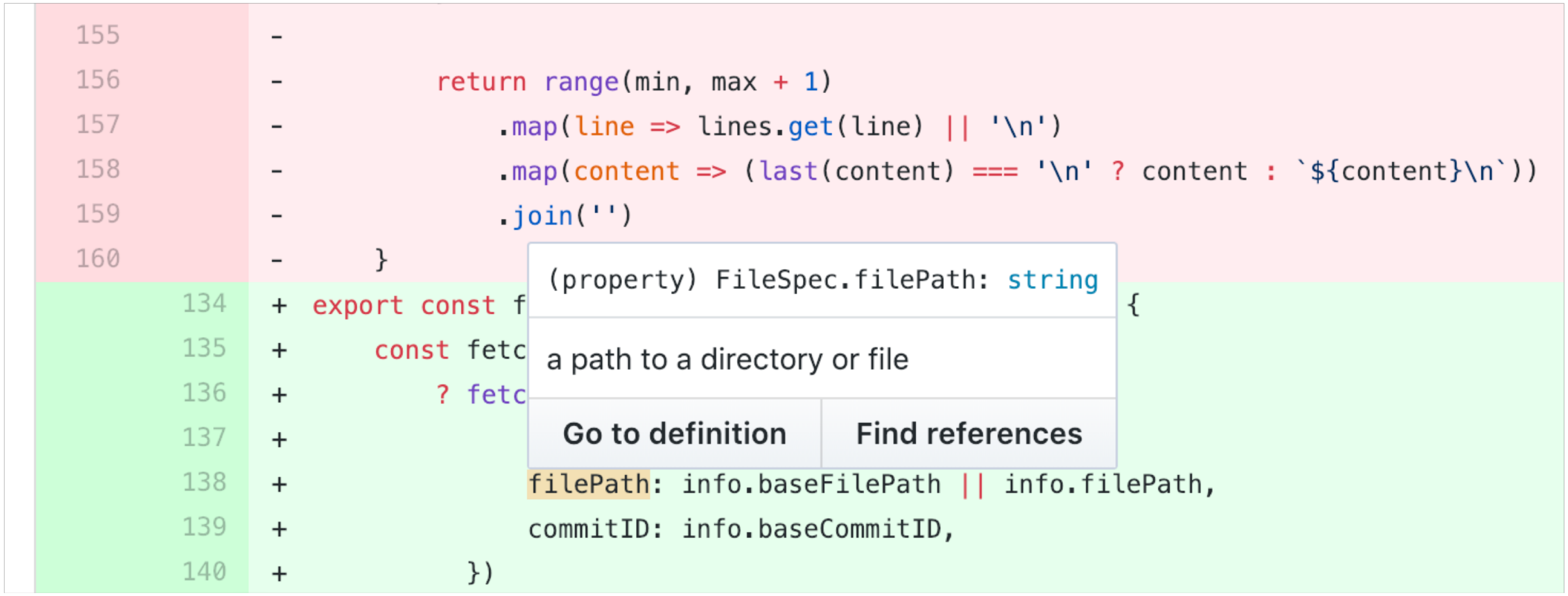


forivall fix: improve toMap typing to allow inferring K and V (#21)

269 lines (245 sloc) | 8.75 KB

```
import { ConcatIterator } from './concat'
import {
import {
import {
import {
import { toIterator } from './utils'
import { ZipIterator } from './zip'

export class IteratorWithOperators<T> implements IterableIterator<T> {
    /**
     * @param source Iterator to wrap
     */
```

```
(alias) function toIterator<T>(collection: Iterator<T> | Iterable<T>):
import toIterator
```

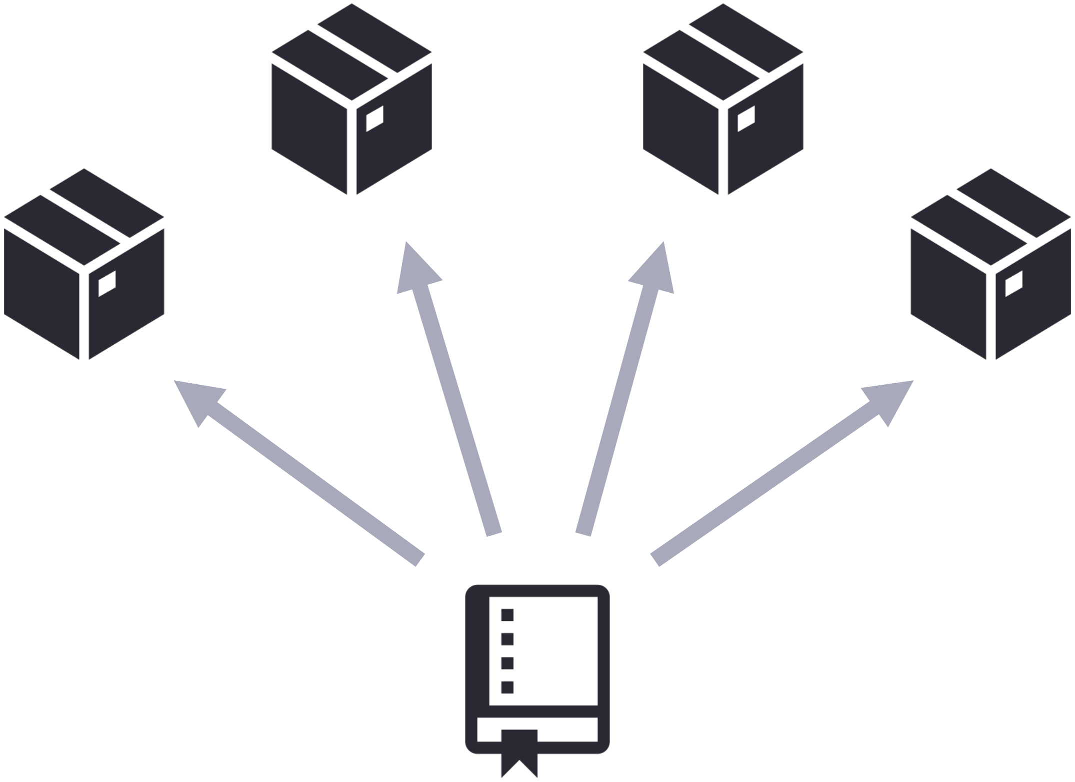


```
-
-        return range(min, max + 1)
-            .map(line => lines.get(line) || '\n')
-            .map(content => (last(content) === '\n' ? content : `${content}\n`))
-            .join('')
-    }
+ export const f                                        {
+     const fetc
+         ? fetc
+
+               filePath: info.baseFilePath || info.filePath,
+               commitID: info.baseCommitID,
+           })
```

(property) FileSpec.filePath: string

a path to a directory or file

Dependencies



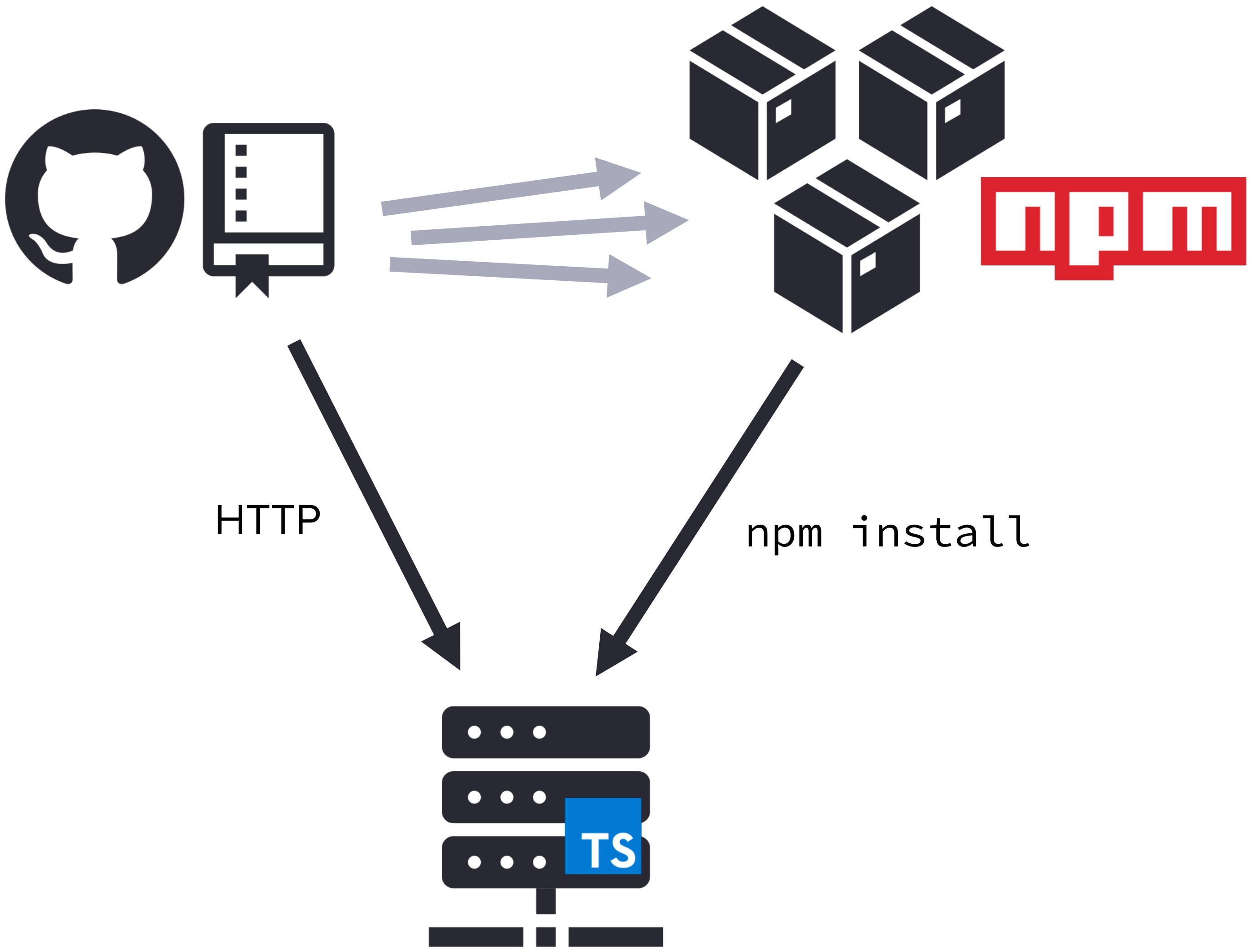
Type declaration files

```
export function insertionSort(array: number[]): number[] {
  let current: number;
  let j: number;
  for (let i = 1; i < array.length; i += 1) {
    current = array[i];
    j = i - 1;
    while (j >= 0 && array[j] - current > 0) {
      array[j + 1] = array[j];
      j -= 1;
    }
    array[j + 1] = current;
  }
  return array;
}
```

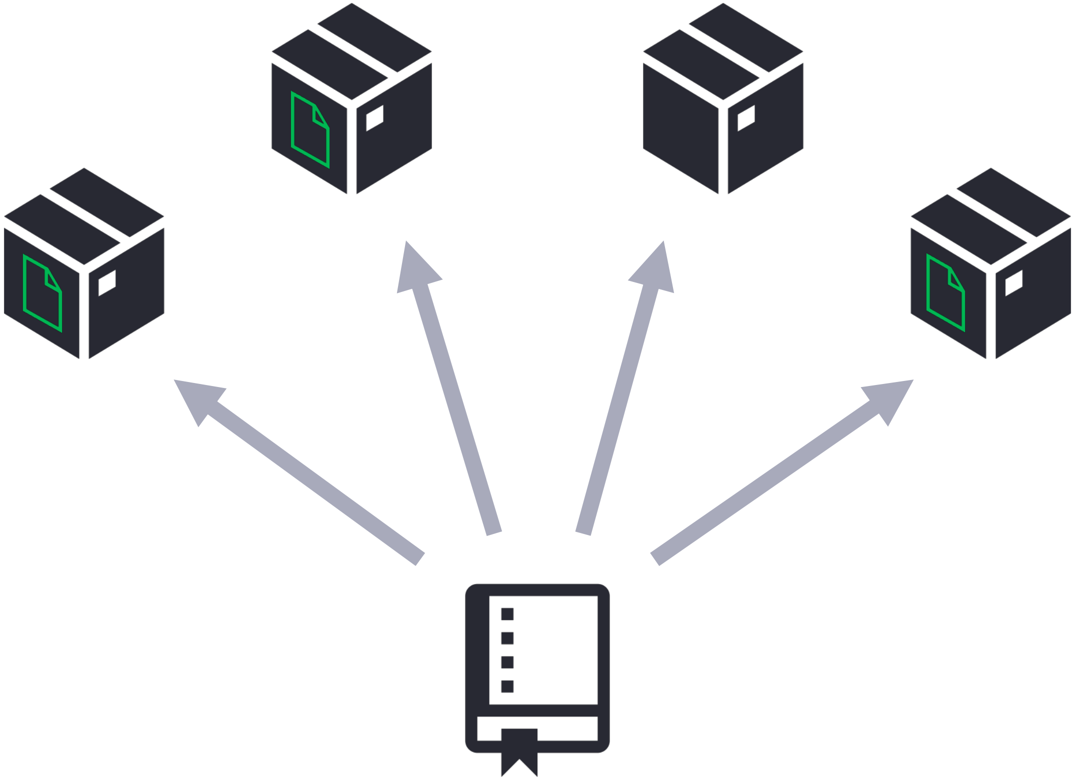
insertionSort.ts

```
export function insertionSort(array: number[]): number[];
```

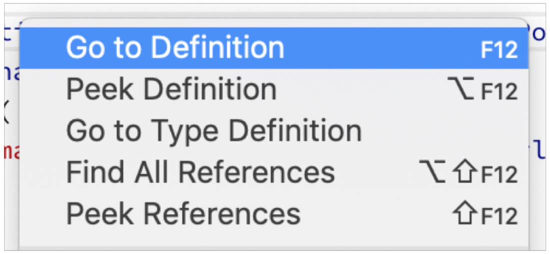
insertionSort.d.ts


npm
HTTP
npm install
TS

Dependencies

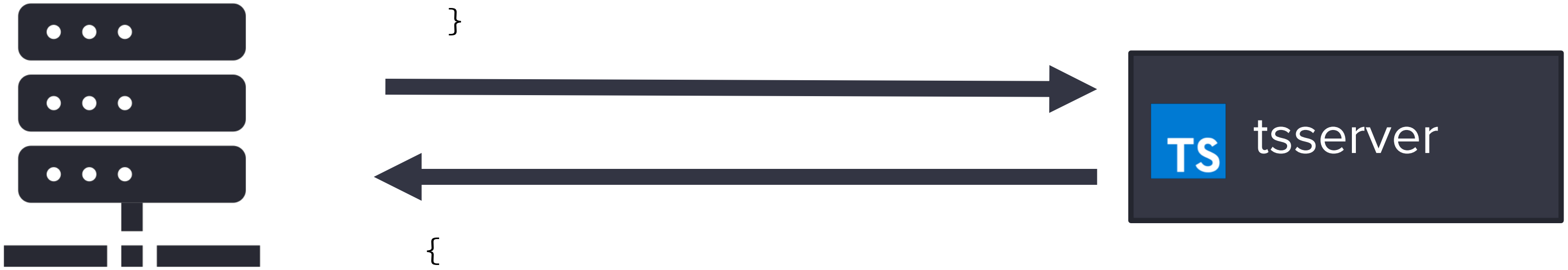


Cross-repository code intelligence



Go to Definition	F12
Peek Definition	⌥F12
Go to Type Definition	
Find All References	⌥⇧F12
Peek References	⇧F12

```
{
  "command": "definition",
  "seq": 1,
  "type": "request",
  "arguments": {
    "file": "/foo.ts",
    "line": 17,
    "offset": 10
  }
}
```

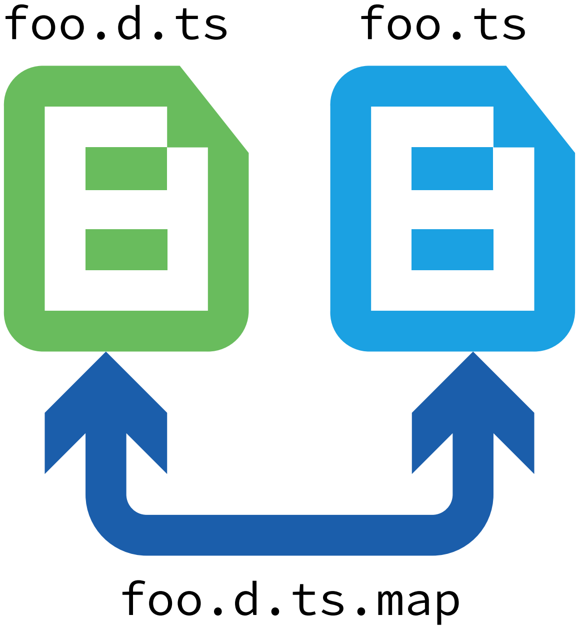


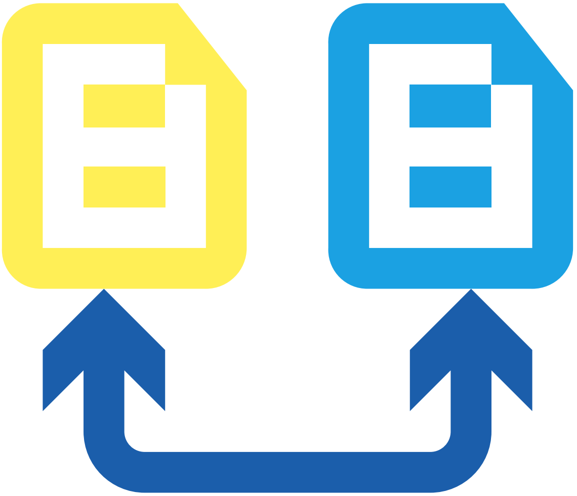
```
{
  "seq": 1,
  "type": "response",
  "command": "definition",
  "request_seq": 6,
  "success": true,
  "body": [
    {
      "file": "/node_modules/bar/index.d.ts",
      "start": { "line": 17, "offset": 10 },
      "end": { "line": 17, "offset": 16 }
    }
  ]
}
```

Where is the source of the package?

```
/node_modules/foo/package.json

{
  "name": "foo",
  "repository": {
    "type": "git",
    "url": "https://github.com/foo/foo",
    "directory": "packages/foo"
  },
  "gitHead": "2d80b06460d26dbbb88ce271c60cfef94ddb5824"
}
```

Declaration Maps





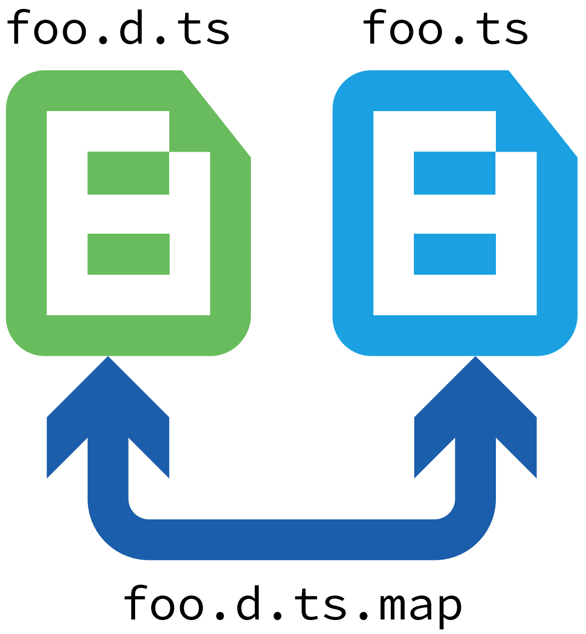
foo.js

foo.ts

foo.js.map

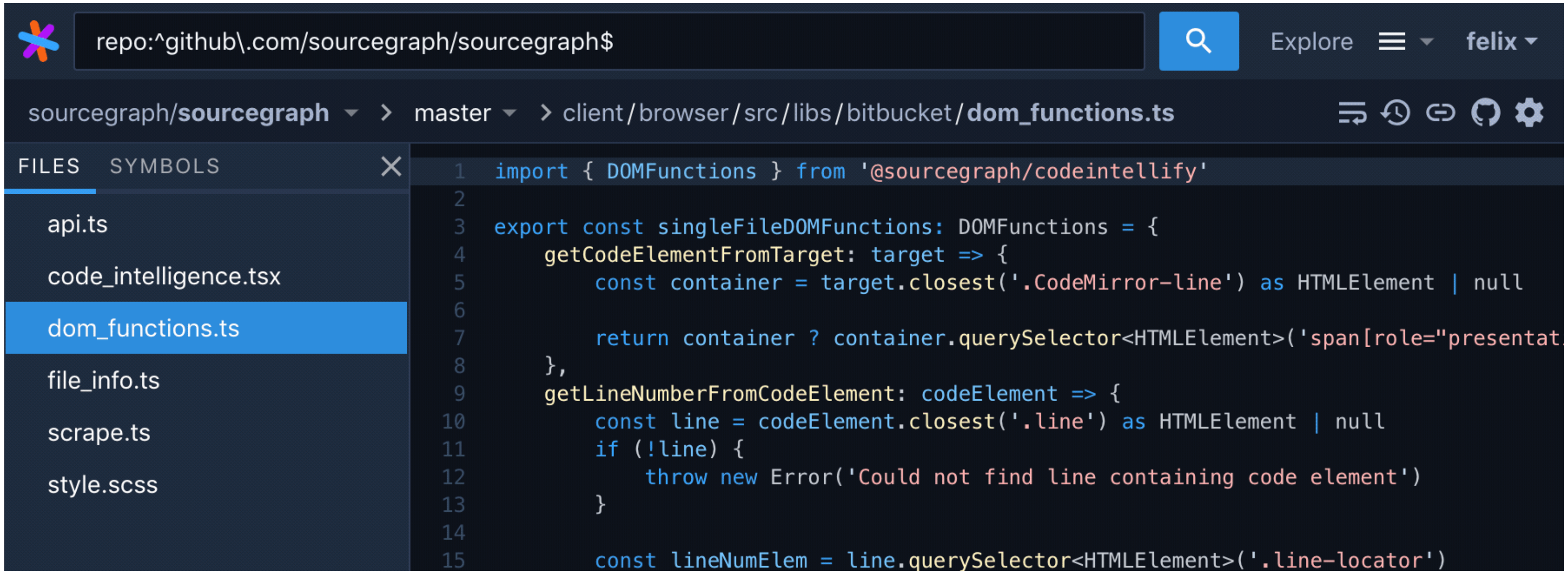
```
{
  "file": "foo.js",
  "sources": ["../src/foo.ts"],
  "mappings": "AAAA,OAAO,6BAA6B,CAAA;AAMpC,OAAO,IAAI,CAAA"
}
```

Declaration Maps



```
export function insertionSort(array:
number[]): number[];
```

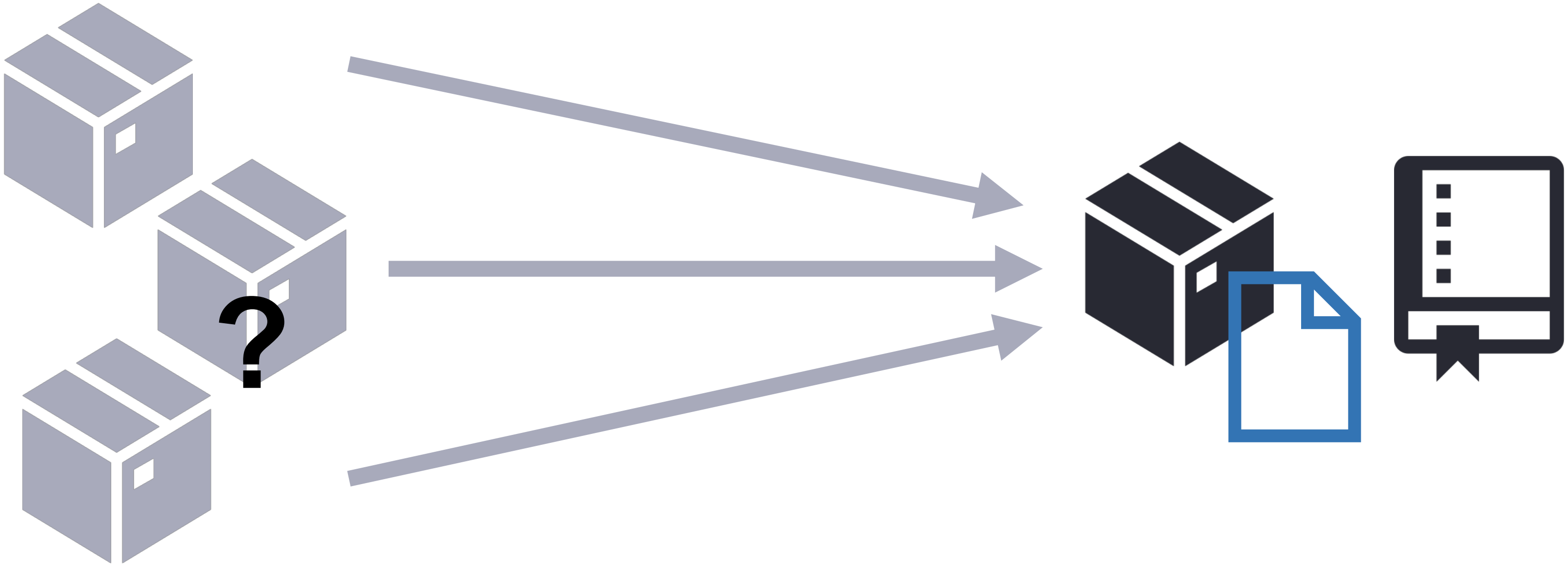
```
export function insertionSort(array:
number[]): number[] {
  let current: number;
  let j: number;
  for (let i = 1; i < array.length; i += 1) {
    current = array[i];
    j = i - 1;
    while (j >= 0 && array[j] - current > 0)
{
      array[j + 1] = array[j];
```

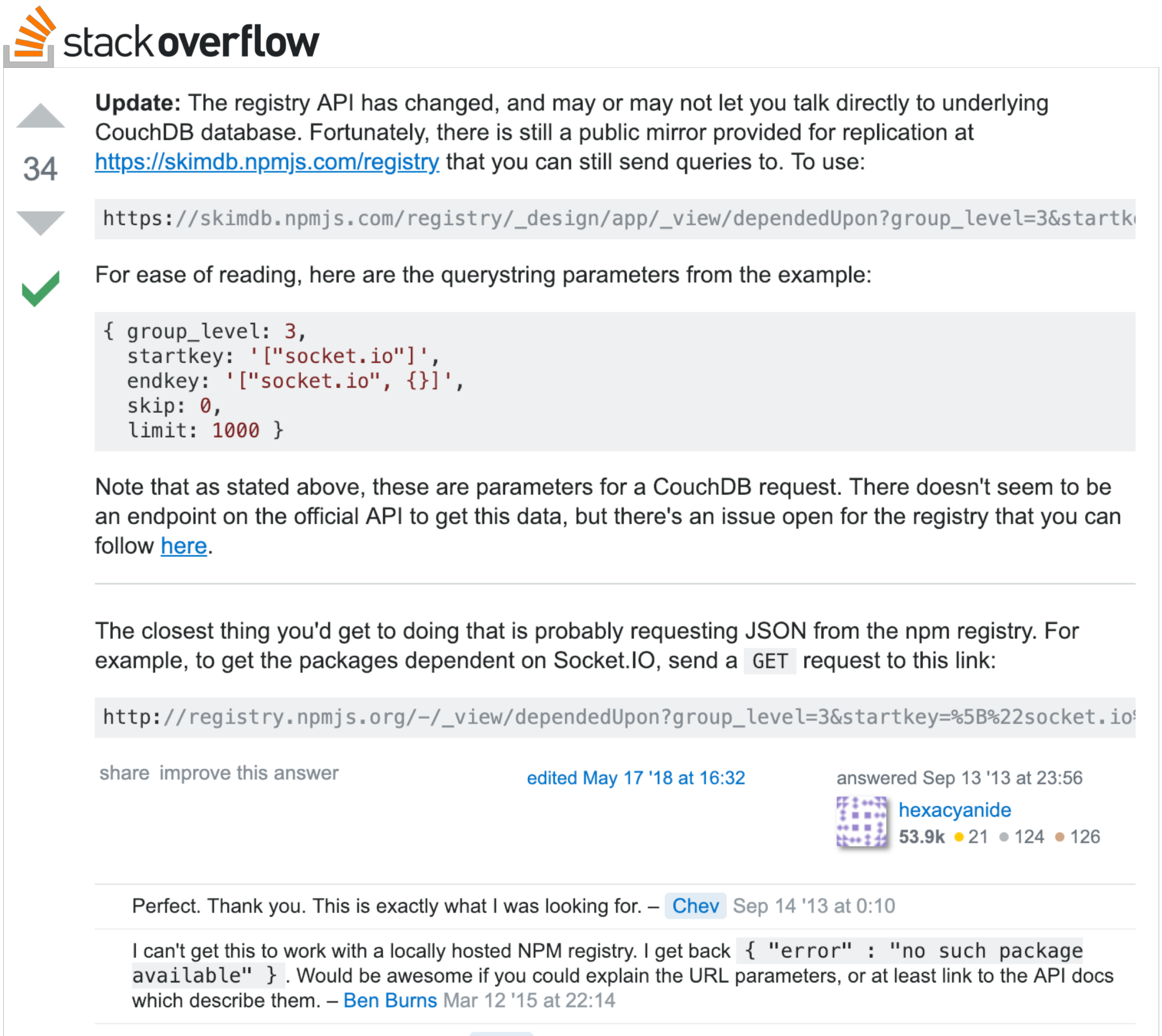


```typescript
import { DOMFunctions } from '@sourcegraph/codeintellify'

export const singleFileDOMFunctions: DOMFunctions = {
    getCodeElementFromTarget: target => {
        const container = target.closest('.CodeMirror-line') as HTMLElement | null

        return container ? container.querySelector<HTMLElement>('span[role="presentat
    },
    getLineNumberFromCodeElement: codeElement => {
        const line = codeElement.closest('.line') as HTMLElement | null
        if (!line) {
            throw new Error('Could not find line containing code element')
        }

        const lineNumElem = line.querySelector<HTMLElement>('.line-locator')
```

What about the inverse?

Cross-repository find-references





Update: The registry API has changed, and may or may not let you talk directly to underlying CouchDB database. Fortunately, there is still a public mirror provided for replication at https://skimdb.npmjs.com/registry that you can still send queries to. To use:

```
https://skimdb.npmjs.com/registry/_design/app/_view/dependedUpon?group_level=3&startk
```

For ease of reading, here are the querystring parameters from the example:

```
{ group_level: 3,
  startkey: '["socket.io"]',
  endkey: '["socket.io", {}]',
  skip: 0,
  limit: 1000 }
```

Note that as stated above, these are parameters for a CouchDB request. There doesn't seem to be an endpoint on the official API to get this data, but there's an issue open for the registry that you can follow here.

The closest thing you'd get to doing that is probably requesting JSON from the npm registry. For example, to get the packages dependent on Socket.IO, send a `GET` request to this link:

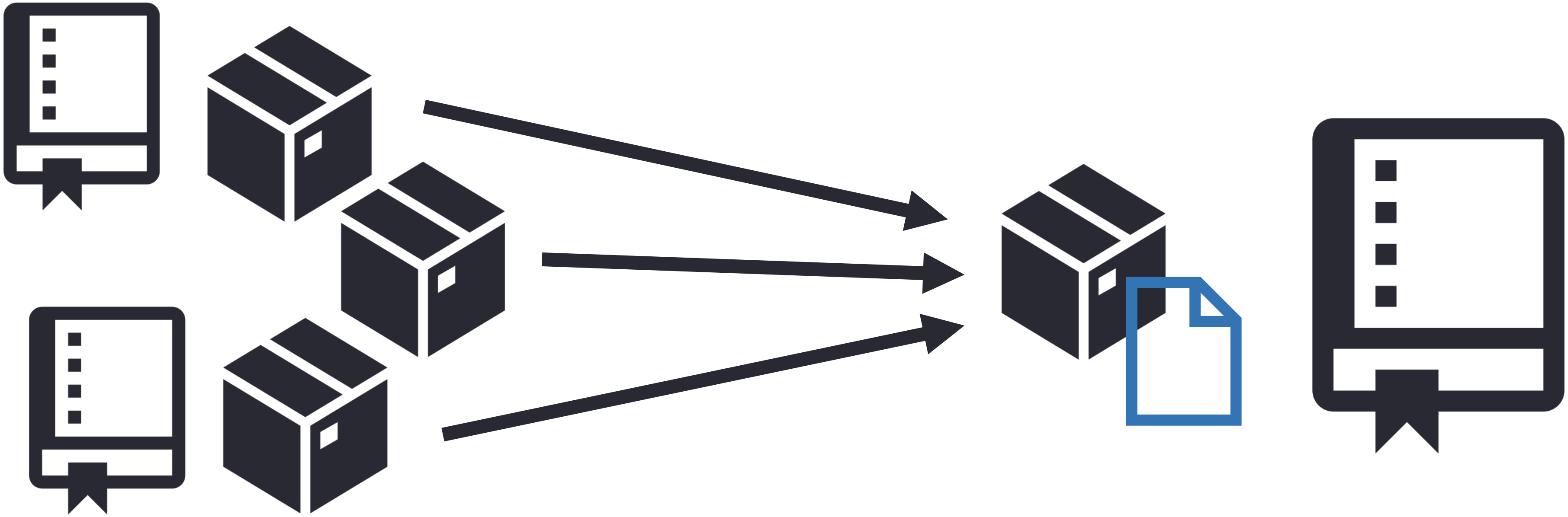
```
http://registry.npmjs.org/-/_view/dependedUpon?group_level=3&startkey=%5B%22socket.io
```

edited May 17 '18 at 16:32

answered Sep 13 '13 at 23:56
hexacyanide
53.9k ● 21 ● 124 ● 126

Perfect. Thank you. This is exactly what I was looking for. – Chev Sep 14 '13 at 0:10

I can't get this to work with a locally hosted NPM registry. I get back `{ "error" : "no such package available" }`. Would be awesome if you could explain the URL parameters, or at least link to the API docs which describe them. – Ben Burns Mar 12 '15 at 22:14



```
{
  "name": "foo",
  "repository": {
    "type": "git",
    "url": "https://github.com/foo/foo"
  }
}
```

D

```
"rootUri": "https://sourcegraph.com/github.com/some/dependent/-/raw/"
```

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "textDocument/references",
  "params": {
    "textDocument": {
      "uri": "https://sourcegraph.com/github.com/foo/foo/-/raw/src/index.ts"
    },
    "position": {
      "line": 17,
      "character": 10
    }
  }
}
```

1. Find out package name

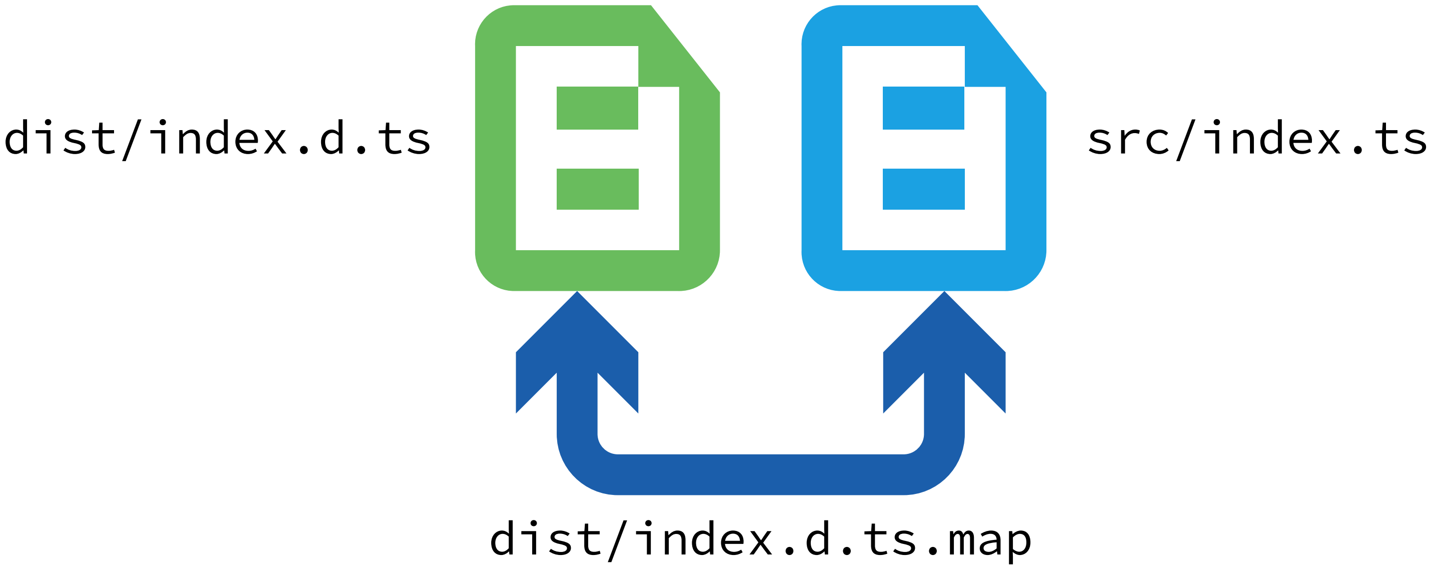
```
GET https://sourcegraph.com/github.com/foo/foo/-/raw/src/package.json
🔴 404 Not Found

GET https://sourcegraph.com/github.com/foo/foo/-/raw/package.json
🟢 200 OK
{ "name": "foo" }
```

2. Find declaration map that points to source file we try to find references for

```
find **/node_modules/foo/**/*.d.ts.map
```

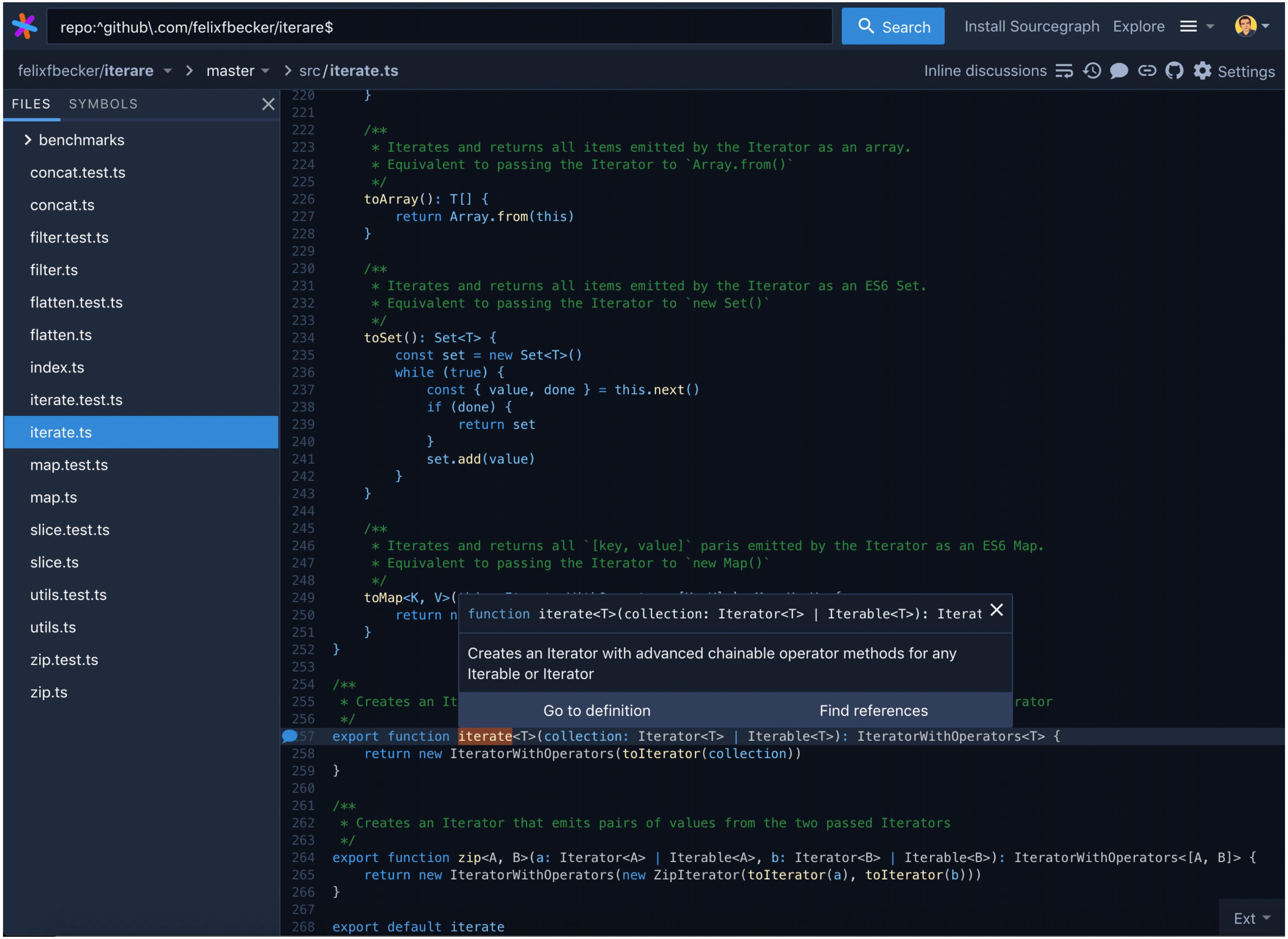
```
{
  "file": “index.d.ts",
  "sources": ["../src/index.ts"],
  "mappings": "AAAA,OAAO,6BAA6B,CAAA;AAMpC,OAAO,IAAI,CAAA"
}
```

3. Use declaration map to map position in source file to position in declaration file



```
export function insertionSort(array:
number[]): number[];
```

```
export function insertionSort(array:
number[]): number[] {
  let current: number;
  let j: number;
  for (let i = 1; i < array.length; i += 1) {
    current = array[i];
    j = i - 1;
    while (j >= 0 && array[j] - current > 0)
{
      array[j + 1] = array[j];
```



```
    }

    /**
     * Iterates and returns all items emitted by the Iterator as an array.
     * Equivalent to passing the Iterator to `Array.from()`
     */
    toArray(): T[] {
        return Array.from(this)
    }

    /**
     * Iterates and returns all items emitted by the Iterator as an ES6 Set.
     * Equivalent to passing the Iterator to `new Set()`
     */
    toSet(): Set<T> {
        const set = new Set<T>()
        while (true) {
            const { value, done } = this.next()
            if (done) {
                return set
            }
            set.add(value)
        }
    }

    /**
     * Iterates and returns all `[key, value]` paris emitted by the Iterator as an ES6 Map.
     * Equivalent to passing the Iterator to `new Map()`
     */
    toMap<K, V>(
        return n
    }
}

/**
 * Creates an It                                                              rator
 */
export function iterate<T>(collection: Iterator<T> | Iterable<T>): IteratorWithOperators<T> {
    return new IteratorWithOperators(toIterator(collection))
}

/**
 * Creates an Iterator that emits pairs of values from the two passed Iterators
 */
export function zip<A, B>(a: Iterator<A> | Iterable<A>, b: Iterator<B> | Iterable<B>): IteratorWithOperators<[A, B]> {
    return new IteratorWithOperators(new ZipIterator(toIterator(a), toIterator(b)))
}

export default iterate
```

```
function iterate<T>(collection: Iterator<T> | Iterable<T>): Iterat
```

Creates an Iterator with advanced chainable operator methods for any Iterable or Iterator

Last words

- Everything shown is live in production!
  - https://sourcegraph.com
  - On GitHub: Sourcegraph browser extension
- Everything shown is open source! https://github.com/sourcegraph/sourcegraph-typescript

Thanks for listening

AMA