Advanced TypeScript Tooling at Scale

@felixfbecker
### Most Loved, Dreaded, and Wanted Languages

<table>
<thead>
<tr>
<th>Loved</th>
<th>Dreaded</th>
<th>Wanted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rust</td>
<td>78.9%</td>
<td></td>
</tr>
<tr>
<td>Kotlin</td>
<td>75.1%</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>68.0%</td>
<td></td>
</tr>
<tr>
<td>TypeScript</td>
<td>67.0%</td>
<td></td>
</tr>
<tr>
<td>Go</td>
<td>65.6%</td>
<td></td>
</tr>
<tr>
<td>Swift</td>
<td>65.1%</td>
<td></td>
</tr>
<tr>
<td>JavaScript</td>
<td>61.9%</td>
<td></td>
</tr>
<tr>
<td>C#</td>
<td>60.4%</td>
<td></td>
</tr>
<tr>
<td>F#</td>
<td>59.6%</td>
<td></td>
</tr>
</tbody>
</table>

StackOverflow developer survey 2018
% of npm users using a transpiler,
npm developer survey 2018
What makes TypeScript great?
Tooling
TypeScript architecture

- **VS Managed Language Service** (shims.ts)
- **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 }
    }
  ]
}
```
We can do that!

wss://typescript.sourcegraph.com
WebSocket
LSP vs TSServer protocol

- Almost the same!
- Except LSP
  - Is language-agnostic
  - Uses the JSON-RPC standard
  - Has simpler request cancellation
  - Uses URIs instead of file paths
```json
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "initialize",
    "params": {
        "rootUri": "https://sourcegraph.com/github.com/nestjs/nest/-/raw/
    }
}
```
HTTP root URLs

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
```json
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "initialize",
    "params": {
        "rootUri": "https://sourcegraph.com/github.com/nestjs/nest/-/raw/
    }
}
```
```typescript
import { ConcatIterator } from './concat'

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

import {
  import toIterator
}

import { toIterator } from './utils'

import { ZipIterator } from './zip'

export class IteratorWithOperators<T> implements IterableIterator<T> {
  /**
   * @param source Iterator to wrap
   */
```
return range(min, max + 1)
  .map(line => lines.get(line) || '
')
  .map(content => (last(content) === '
' ? content : `\${content}\n`))
  .join('')

export const fetchFileSpec = async (path: string): Promise<FileSpec> => {
  const fetch = require('node-fetch')
  const info = await fetch(path)
  return {
    filePath: info.baseFilePath || info.filePath,
    commitID: info.baseCommitID,
  }
}
Dependencies
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;
}
HTTP

npm install
Dependencies
Cross-repository code intelligence

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

```json
{
    "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.d.ts  foo.ts

foo.d.ts.map
{  
    "file": "foo.js",
    "sources": ["../src/foo.ts"],
    "mappings": "AAAA,OAA0,6BAA6B,CAAA;AAMpC,OAA0,IAAI,CAAA"
}
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 = j - 1;
        }
        array[j + 1] = current;
    }
    return array;
}
```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="presentation"]') : null
    },
    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&startkey=*

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

```json
{ 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%22%5D

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

{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "textDocument/references",
    "params": {
        "textDocument": {
        },
        "position": {
            "line": 17,
            "character": 10
        }
    }
}
1. Find out package name

● 404 Not Found

● 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,0AA0,6BAA6B,CAAA;AAMpC,0AA0,IAAI,CAAA"
}
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;
}

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

dist/index.d.ts

src/index.ts

dist/index.d.ts.map
```typescript
/**
 * Iterates and returns all items emitted by the Iterator as an array.
 * Equivalent to passing the Iterator to `Array.from()`
 * @param T - The type of the elements emitted by the Iterator.
 * @returns An array containing all the elements emitted by the Iterator.
 */
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()`
 * @param T - The type of the elements emitted by the Iterator.
 * @returns A Set containing all the elements emitted by the Iterator.
 */
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]` pairs emitted by the Iterator as an ES6 Map.
 * Equivalent to passing the Iterator to `new Map()`
 * @param K - The type of the keys emitted by the Iterator.
 * @param V - The type of the values emitted by the Iterator.
 * @returns A Map containing all the key-value pairs emitted by the Iterator.
 */
toMap<K, V>(key?: K, value?: V): Map<K, V> {
    return new Map;
}

/*
 * Creates an Iterator with advanced chainable operator methods for any
 * Iterable or Iterator
 */
export function iterate<T>(collection: Iterator<T> | Iterable<T>, ...rest): Iterable<T>

/*
 * Creates an Iterator that emits pairs of values from the two passed Iterators
 */
```

Last words

- Everything shown is live in production!
  - [https://sourcegraph.com](https://sourcegraph.com)
  - On GitHub: [Sourcegraph browser extension](https://github.com/sourcegraph/sourcegraph)

- Everything shown is open source!
  [https://github.com/sourcegraph/sourcegraph-typescript](https://github.com/sourcegraph/sourcegraph-typescript)
Thanks for listening

AMA