Code Generation with TypeScript

Hello,

I am Benny.

TypeScript enthusiast and
creator of vote4code.com

You can follow me on Twitter
@bennycode

Berlin.JS!

I am Florian.

TypeScript enthusiast and
creator of sortjson.com

You can follow me on GitHub
@fffflorian

wire™

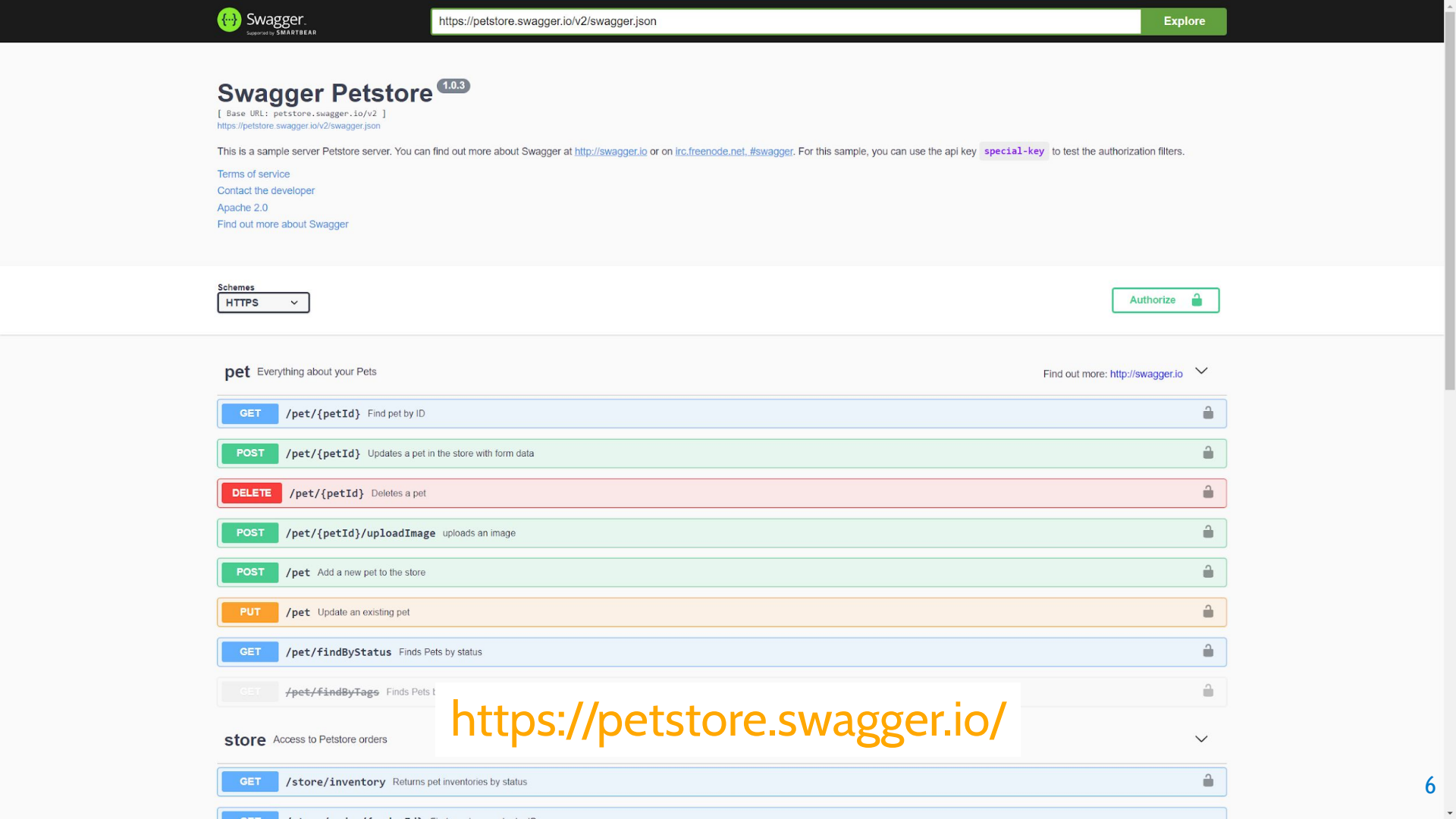
The most secure collaboration platform

END-TO-END ENCRYPTED

OPEN SOURCE

INDEPENDENT SECURITY AUDITS AVAILABLE

Frontend Backend


Swagger
Supported by SMARTBEAR
https://petstore.swagger.io/v2/swagger.json
Explore
Swagger Petstore
1.0.3
[Base URL: petstore.swagger.io/v2]
https://petstore.swagger.io/v2/swagger.json
This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key special-key to test the authorization filters.
Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger
Schemes
HTTPS
Authorize
pet Everything about your Pets
Find out more: http://swagger.io
GET /pet/{petId} Find pet by ID
POST /pet/{petId} Updates a pet in the store with form data
DELETE /pet/{petId} Deletes a pet
POST /pet/{petId}/uploadImage uploads an image
POST /pet Add a new pet to the store
PUT /pet Update an existing pet
GET /pet/findByStatus Finds Pets by status
GET /pet/findByTags Finds Pets t
https://petstore.swagger.io/
store Access to Petstore orders
GET /store/inventory Returns pet inventories by status

```
  /**
   * @param petId Pet id to delete
   */
  deletePet = async (petId: number): Promise<void> => {
    const config: AxiosRequestConfig = {
      method: 'delete',
      url: `/pet/${petId}`,
    };
    await this.apiClient.request(config);
  };

  /**
   * @param body Pet object that needs to be added to the store
   */
  addPet = async (body: Pet): Promise<void> => {
    const config: AxiosRequestConfig = {
      data: {
        ...body,
      },
      method: 'post',
      url: '/pet',
    };
    await this.apiClient.request(config);
  };

  /**
   * @param body Pet object that needs to be added to the store
   */
  updatePet = async (body: Pet): Promise<void> => {
    const config: AxiosRequestConfig = {
      data: {
        ...body,
      },
      method: 'put',
      url: '/pet',
    };
    await this.apiClient.request(config);
  };
}
```

Generated with Swaxios

Code Generation

> A mechanism to produce *[…]* computer programs *[…]* in some automatic manner.
>
> *wikipedia.org*

Iteration I


Trainee
Requires onboarding
Requires reviews
Requires park walks
Doesn’t check syntax

Iteration II

```
/* tslint:disable */

/**
 * This file was automatically generated by "Swaxios".
 * It should not be modified by hand.
 */

import {AxiosInstance, AxiosRequestConfig} from 'axios';
{{#if imports.list.length}}
import {
  {{#each imports.list}}
    {{{this}}},
  {{/each}}
} from '{{{imports.path}}}'
{{/if}}

export class {{{name}}} {
  private readonly apiClient: AxiosInstance;

  constructor(apiClient: AxiosInstance) {
    this.apiClient = apiClient;
  }
  {{#each methods}}

    {{#if this.descriptions}}
      /**
      {{#each this.descriptions}}
       * @param {{{this.name}}} {{{this.text}}}
      {{/each}}
       */
    {{/if}}
    {{{this.parameterMethod}}} = async (
      {{#each this.pathParameters}}
        {{{this.name}}}: {{{this.type}}},
      {{/each}}
      {{#each this.bodyParameters}}
        {{{this.name}}}{{#isnt this.required true}}?{{/isnt}}: {{{this.type}}},
      {{/each}}
      {{#if this.requiresBearerAuthorization}}
        accessTokenCallback: () => Promise<string>,
      {{/if}}
      {{#if this.queryParameters.length}}
        params?: {
```

Handlebars

Requires dependency

Requires helpers

Requires IDE plugin

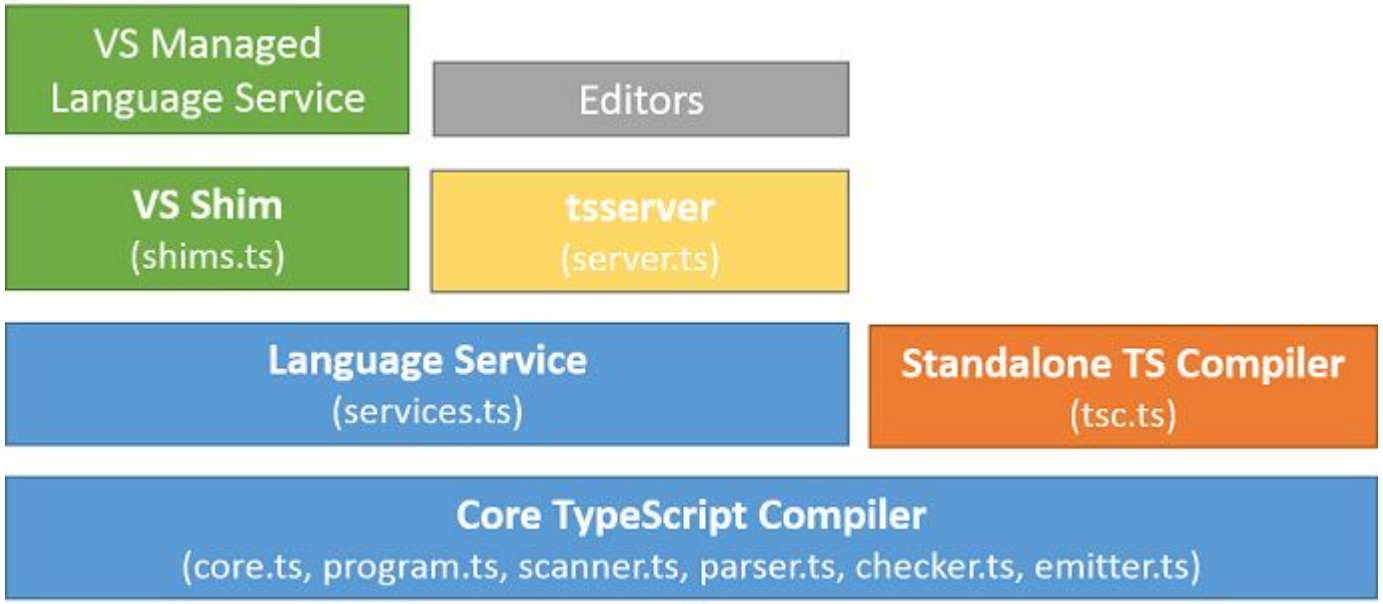
Doesn’t check syntax

Iteration III

Compiler API

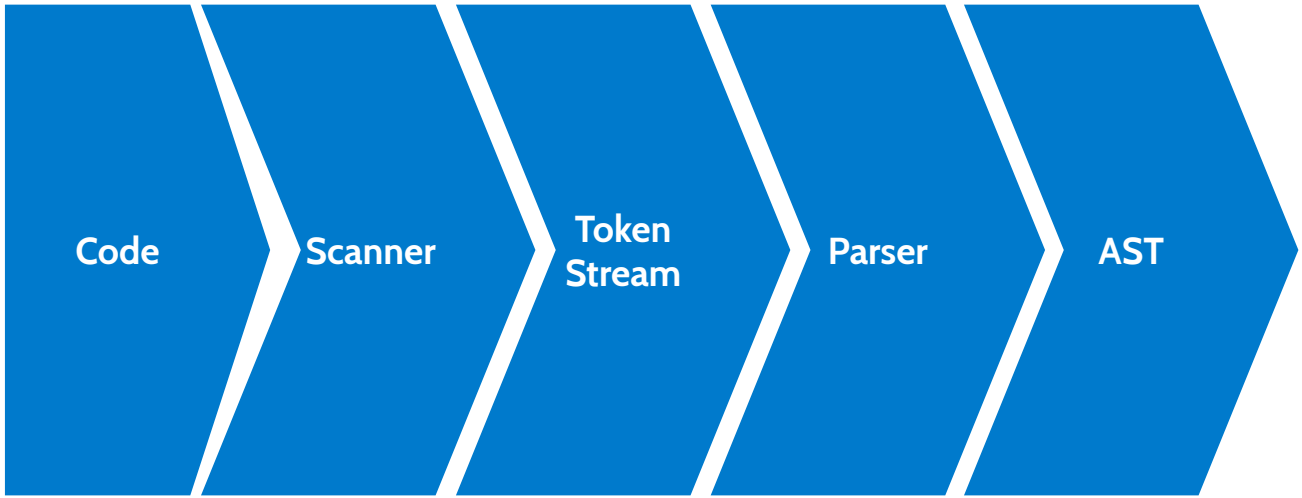
- Ships with TypeScript
- Uses AST format
- Checks syntax
- Generates valid output

Architectural Overview

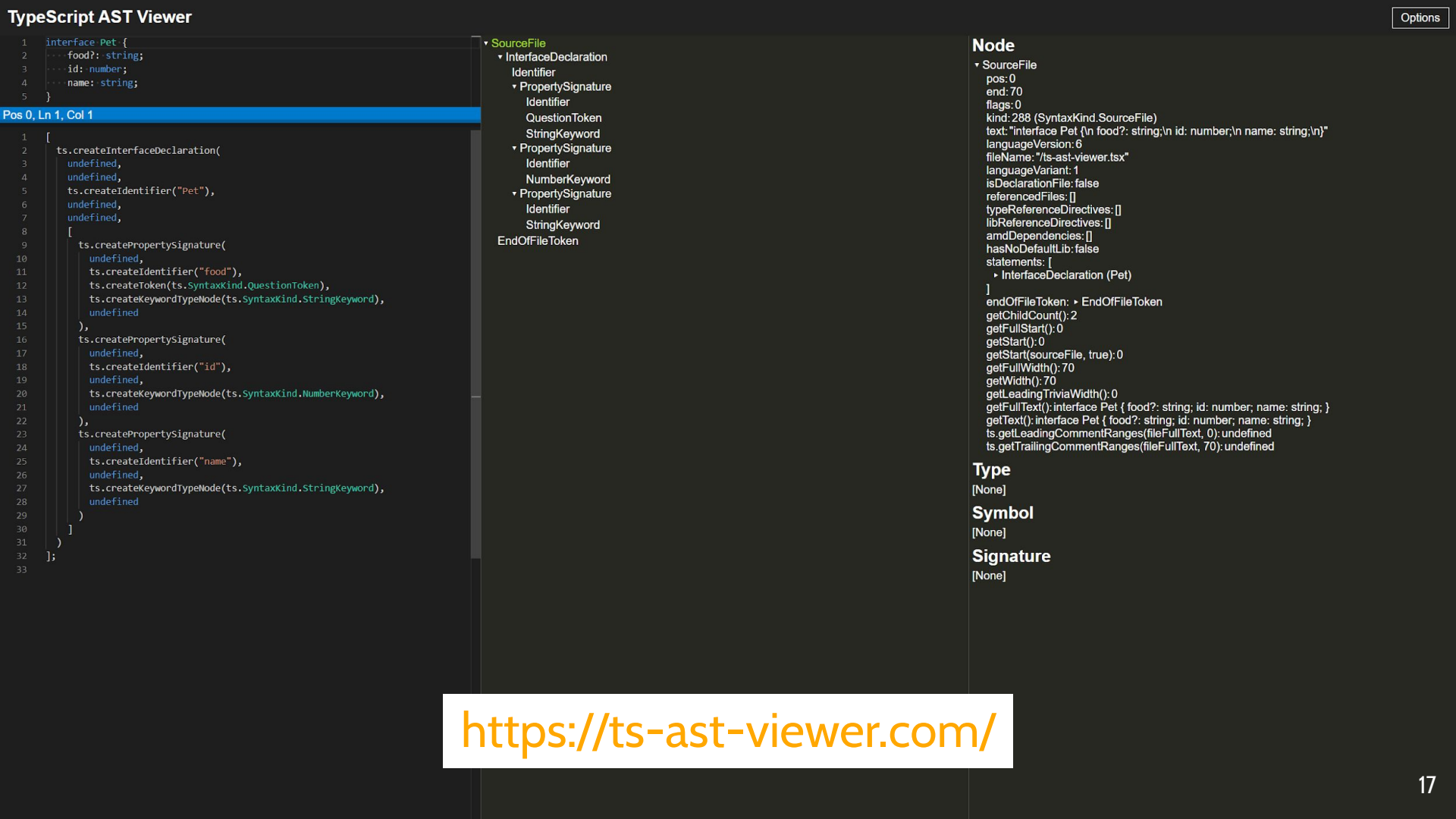


https://github.com/microsoft/TypeScript/wiki/Architectural-Overview

Compiler Phases



https://basarat.gitbook.io/typescript/overview/scanner


TypeScript AST Viewer
Options
interface Pet {
food?: string;
id: number;
name: string;
}
Pos 0, Ln 1, Col 1
[
ts.createInterfaceDeclaration(
undefined,
undefined,
ts.createIdentifier("Pet"),
undefined,
undefined,
[
ts.createPropertySignature(
undefined,
ts.createIdentifier("food"),
ts.createToken(ts.SyntaxKind.QuestionToken),
ts.createKeywordTypeNode(ts.SyntaxKind.StringKeyword),
undefined
),
ts.createPropertySignature(
undefined,
ts.createIdentifier("id"),
undefined,
ts.createKeywordTypeNode(ts.SyntaxKind.NumberKeyword),
undefined
),
ts.createPropertySignature(
undefined,
ts.createIdentifier("name"),
undefined,
ts.createKeywordTypeNode(ts.SyntaxKind.StringKeyword),
undefined
)
]
)
];
SourceFile
InterfaceDeclaration
Identifier
PropertySignature
Identifier
QuestionToken
StringKeyword
PropertySignature
Identifier
NumberKeyword
PropertySignature
Identifier
StringKeyword
EndOfFileToken
Node
SourceFile
pos: 0
end: 70
flags: 0
kind: 288 (SyntaxKind.SourceFile)
text: "interface Pet {\n food?: string;\n id: number;\n name: string;\n}"
languageVersion: 6
fileName: "/ts-ast-viewer.tsx"
languageVariant: 1
isDeclarationFile: false
referencedFiles: []
typeReferenceDirectives: []
libReferenceDirectives: []
amdDependencies: []
hasNoDefaultLib: false
statements: [
▸ InterfaceDeclaration (Pet)
]
endOfFileToken: ▸ EndOfFileToken
getChildCount(): 2
getFullStart(): 0
getStart(): 0
getStart(sourceFile, true): 0
getFullWidth(): 70
getWidth(): 70
getLeadingTriviaWidth(): 0
getFullText(): interface Pet { food?: string; id: number; name: string; }
getText(): interface Pet { food?: string; id: number; name: string; }
ts.getLeadingCommentRanges(fileFullText, 0): undefined
ts.getTrailingCommentRanges(fileFullText, 70): undefined
Type
[None]
Symbol
[None]
Signature
[None]
https://ts-ast-viewer.com/

Code Generation

Nodes (AST)

A program is basically a sequence of statements.

All expressions and statements can be represented as nodes and subtrees.

Printer

Prints a node and its subtree as-is, without any emit transformations.

The result is a stringified version of the source code.

Source File

A destination file that provides context for the node.

Can be written with a wrapper method like `ts.sys.writeFile`.

```ts
import ts from 'typescript';

const ast = ts.createExpressionStatement(ts.createCall(
  ts.createPropertyAccess(
    ts.createIdentifier( text: 'console'),
    ts.createIdentifier( text: 'log'),
  ),
  typeArguments: undefined,
  argumentsArray: [ts.createStringLiteral( text: 'Hello, Berlin.JS!')],
));

const printer = ts.createPrinter();

const fileContext = ts.createSourceFile(
  fileName: '',
  sourceText: '',
  ts.ScriptTarget.Latest,
);

const string = printer.printNode(
  ts.EmitHint.Unspecified,
  ast,
  fileContext,
);

ts.sys.writeFile( path: `./dist/GeneratedCode.ts`, string);
```

Thanks!

Any questions?

Create an issue in
our demo repository.