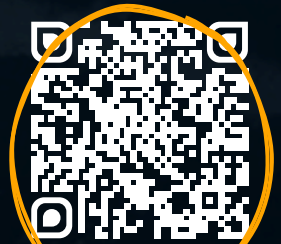
Abstract Syntax Tree (AST) & Typescript API

How I learned to stop worrying and love the AST

Thibault Friedrich – February 26th, 2025



Feedback form

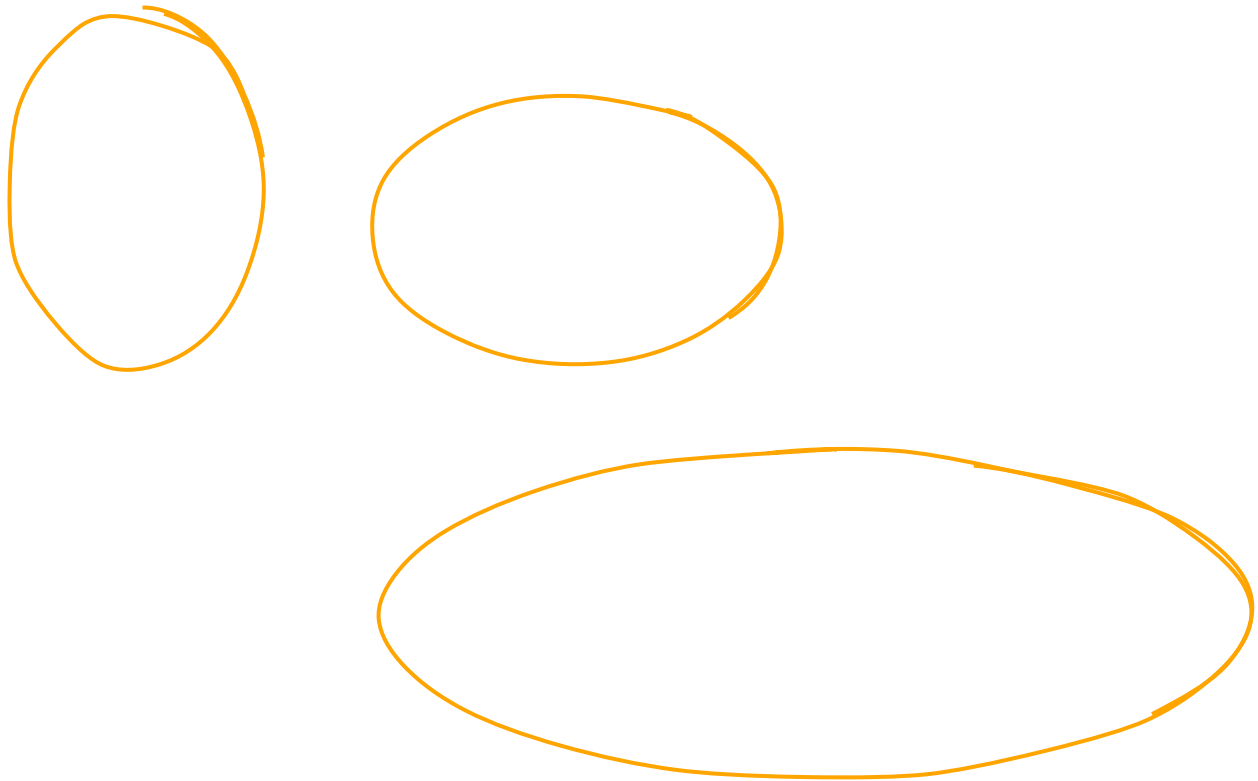
About me: Thibault Friedrich

- Frontend developer for 10 years
- Using *React* for 6+ years and love it
- Strong focus on Ux, Agile and Code craftsmanship
  - ➡️ find my articles on Medium
- Implementing design systems for 3 years
- Maintainer of DesignSystemHub
  - Documentation for design systems
  - Storybook alternative
  - zero config
  - automatic detection of React components

Spoiler

- Instead of focusing on the theory, we will focus on the practice
- Discover AST through a real use case
- Understand the Typescript Compiler API
- Won’t be a complete guide
- Methdology can be reproduced with other AST generators and use cases
- Beginner friendly
- We will dive into the code

Use case: design-system-hub.com



Need to find:

- React components
- JsDoc
- Properties

How to detect a React component in the code?

```tsx
type Props = {
  children: React.ReactNode
  variant: 'primary' | 'black' | 'basic'
  onClick: () => void | Promise<void>
}

/**
 * @deprecated
 */
const ButtonLegacy = ({ children, variant, onClick }: Props) => {
  return <button onClick={onClick} className={style[variant]}>{children}</button>
}
```

Long online search: You don't want to recreate a parser yourself. ➡️ Use an **AST** generator.

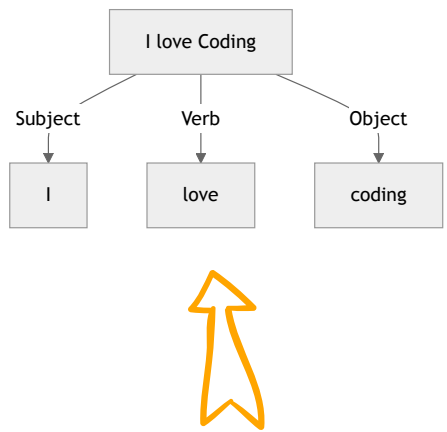
That's the moment you start to be scared and you think this project is too complex for you. 🤯😬

What is an AST (Abstract Syntax Tree)?

When you analyze a language like Javascript or English:

"I love coding"

1. **lexical** analysis to detect the tokens: `I` `love` `coding`
2. **syntactic** analysis to understand the relationship between tokens



This is an AST (Abstract Syntax Tree)

Abstract Syntax Tree (AST) - Explained in Plain English

Good news: Typescript Compiler API

```
import ts from 'typescript'

const program = ts.createProgram(fileNames, tsConfig)
const checker = program.getTypeChecker()

return program
  .getSourceFiles()
  .filter(sourceFile ⇒ !sourceFile.isDeclarationFile)
  .flatMap(sourceFile ⇒ visitSourceFile(sourceFile, checker))
```

Using the Compiler API

Demo time

Conclusion: AST

- don’t be scared
- very powerful
- write unit tests to avoid regressions
- AST Explorer is your friend
- try and retry
- other use cases:
  - eslint custom rules
  - babel plugins
- typescript Compiler API is very easy to master

Questions ?

- Shoot me your questions

Feedback form

Stay in contact

- https://github.com/friedrith/abstract-syntax-tree
- https://thibaultfriedrich.io
- https://design-system-hub.com