

# Project 2

- Example “feature themes”
  - I want to mark questions as resolved,
  - Allow instructors to endorse posts
  - Mark as duplicate
  - Search for topics
  - Tagging based on user type, predefined label, etc.
- Others
  - Auto-disappearing posts 😊
  - Poll posts

# Software Archaeology and Anthropology

17-313 Fall 2023

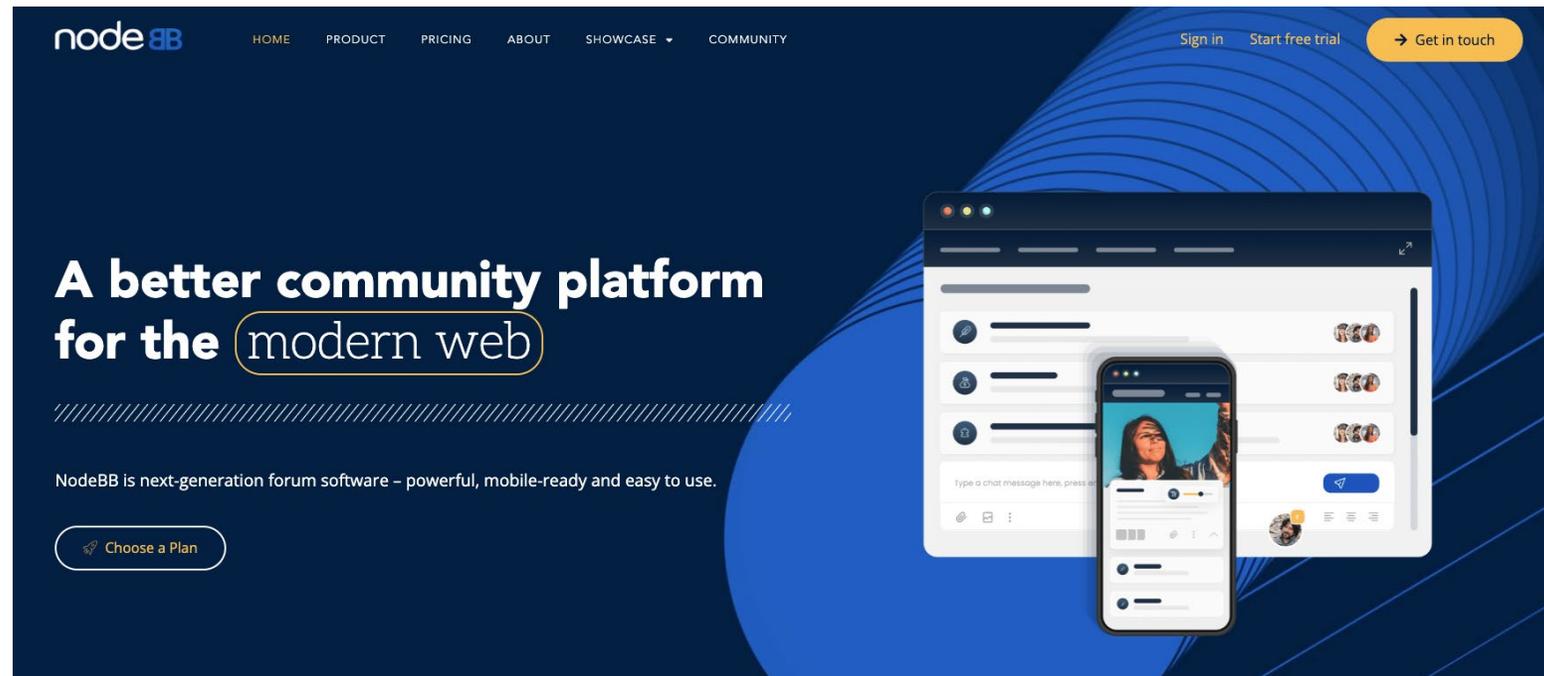
Foundations of Software Engineering

# Learning Goals

- Understand and scope the task of taking on and understanding a new and complex piece of existing software
- Appreciate the importance of configuring an effective IDE
- Contrast different types of code execution environments including local, remote, application, and libraries
- Enumerate both static and dynamic strategies for understanding and modifying a new codebase

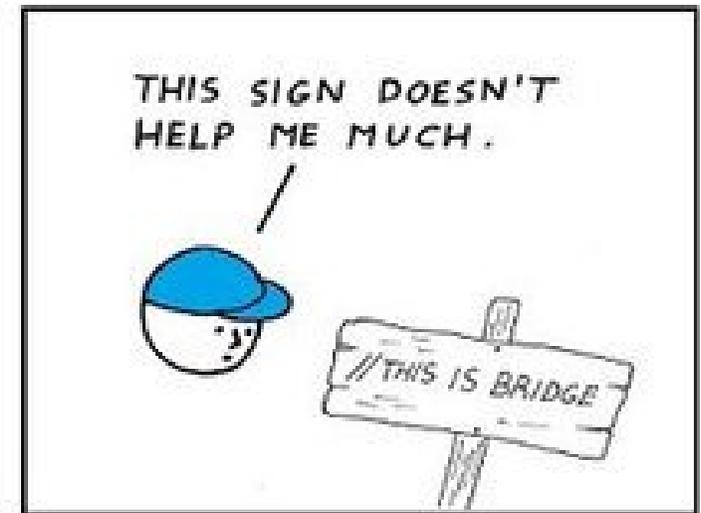
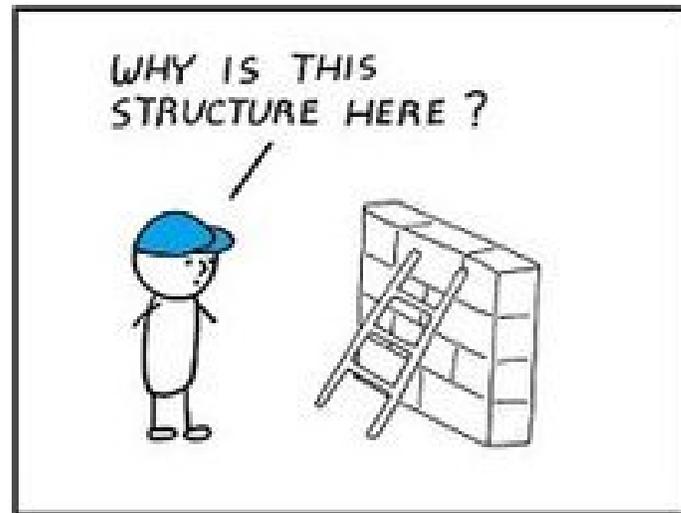
# Context: big old pile of code

- ... do something with it!



**You will never  
understand the  
entire system!**

# Challenge: How do I tackle this codebase?



# Challenge: How do I tackle this codebase?

- Leverage your previous experiences (languages, technologies, patterns)
- Consult documentation, whitepapers
- Talk to experts, code owners
- Follow best practices to build a working model of the system

# Today: How to tackle codebases

- Goal: develop and test a working model or set of working hypotheses about how (some part of) a system works
- Working model: an understanding of the pieces of the system (components), and the way they interact (connections)
- Focus: Observation, probes, and hypothesis testing
  - Helpful tools and techniques!



essentially,  
all models are wrong,  
but some are useful

George E. P. Box

# Live Demonstration: NodeBB

# Steps to Understand a New Codebase

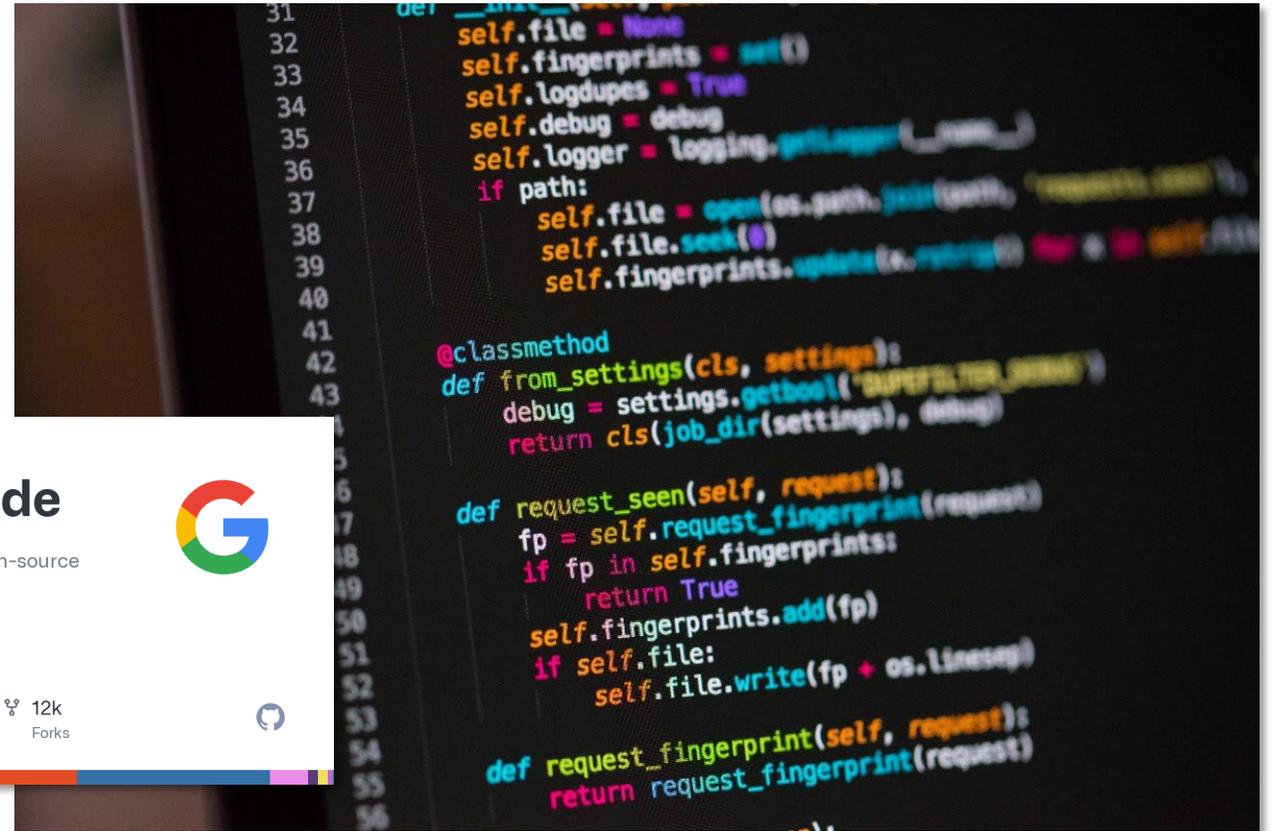
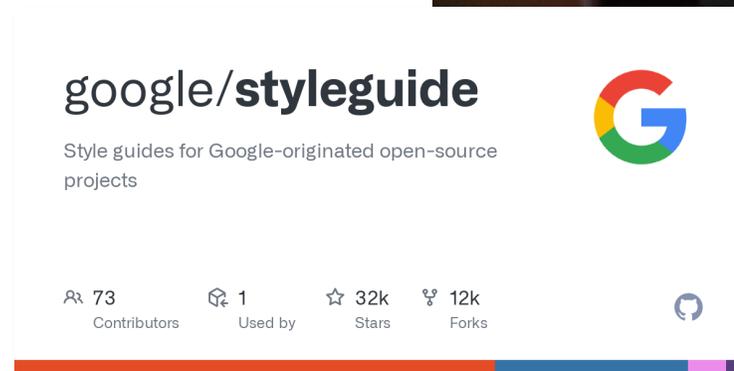
- Look at README.md
- Clone the repo.
- Build the codebase.
- Figure out how to make it run.
- What do you want to mess with?
  - Clone and own
- Traceability - Attach a debugger
  - View Source
  - Find the logs.
  - Search for constants (strings, colors, weird integers (#DEADBEEF))

# Participation Activity

- Take out a piece of paper.
- Write down one pro and one con about trying to understand a new codebase by compiling and building it vs. just reading the code.
- Pair with your neighbor and discuss your answers. Do you agree?
- Share with the class!
- Submit it by the end of class.

# Observation: Software is full of patterns

- File structure
- System architecture
- Code structure
- Names
- ...



# Observation: Software is massively redundant

- There's always something to copy/use as a starting point!



Observation: If code runs, it must have a beginning...



Observation: If code runs, it must exist...

```
0x08048416 <+18>: jg     DWORD PTR [ebp+0x8], 0x1
0x08048419 <+21>: mov   eax, DWORD PTR [ebp+0xc]
0x0804841b <+23>: mov   ecx, DWORD PTR [eax]
0x08048420 <+28>: mov   edx, 0x8048520
0x08048425 <+33>: mov   eax, ds:0x8049648
0x08048429 <+37>: mov   DWORD PTR [esp+0x8], ecx
0x0804842d <+41>: mov   DWORD PTR [esp+0x4], edx
0x08048430 <+44>: mov   DWORD PTR [esp], eax
0x08048435 <+49>: call  0x8048338 <fprintf@plt>
0x0804843a <+54>: mov   eax, 0x1
0x0804843c <+56>: jmp   0x8048459 <main+85>
0x0804843f <+59>: mov   eax, DWORD PTR [ebp+0xc]
0x08048442 <+62>: add   eax, 0x4
0x08048444 <+64>: mov   eax, DWORD PTR [eax]
0x08048448 <+68>: mov   DWORD PTR [esp+0x4], eax
0x0804844c <+72>: lea  eax, [esp+0x10]
0x0804844f <+75>: mov   DWORD PTR [esp], eax
0x08048454 <+78>: call  0x8048338 <fprintf@plt>
```

# The Beginning: Entry Points

- Locally installed programs: run cmd, OS launch, I/O events, etc.
- Local applications in dev: build + run, test, deploy (e.g., docker)
- Web apps server-side: Browser sends HTTP request (GET/POST)
- Web apps client-side: Browser runs JavaScript, event handlers

# Creating a model of unfamiliar code



Source code built  
locally

Static  
Information  
Gathering

Dynamic  
Information  
Gathering

# Static Information Gathering

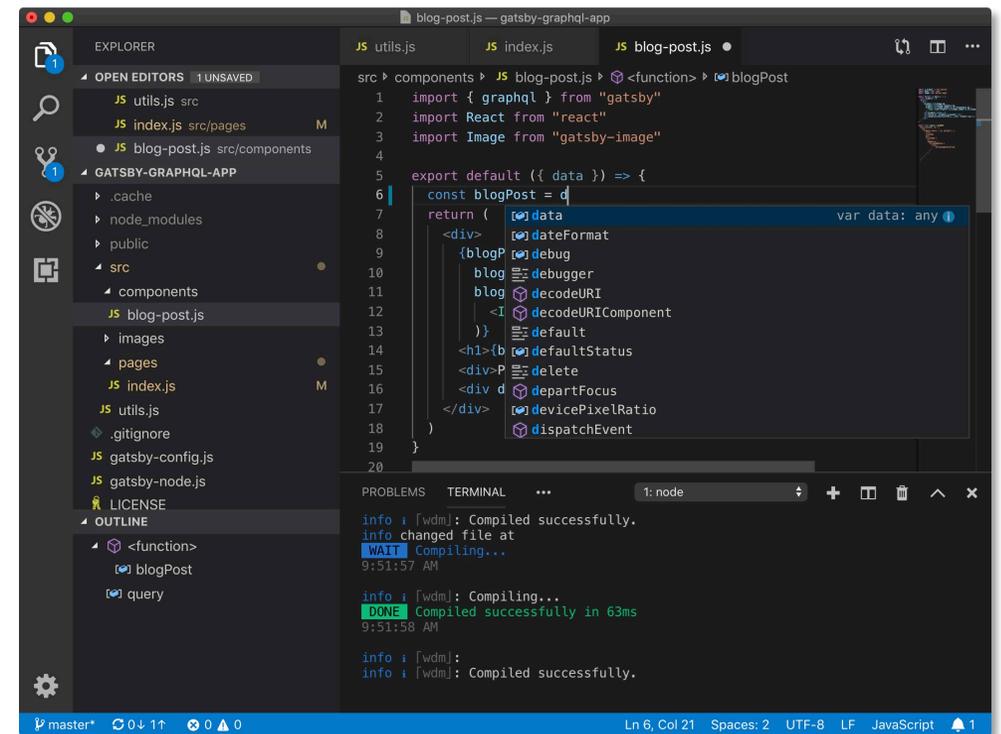
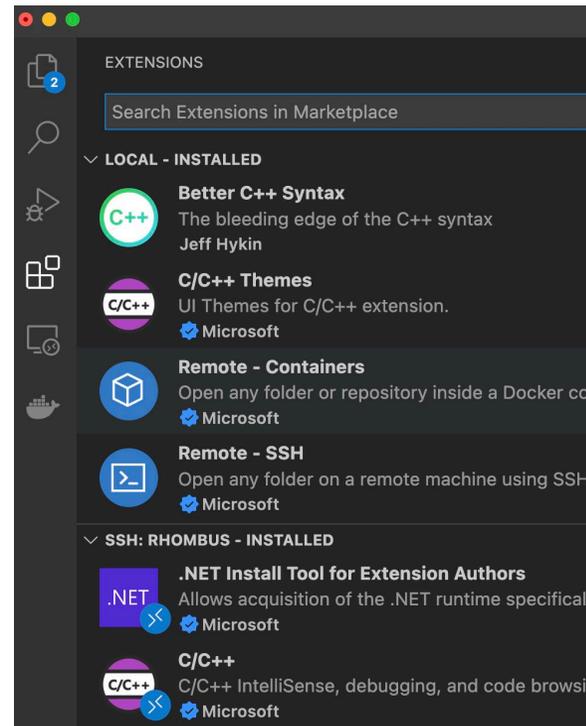
- Basic needs:
  - Code/file search and navigation
  - Code editing (probes)
  - Execution of code, tests
  - Observation of output (observation)
- Many choices here on tools! Depends on circumstance.
  - grep/find/etc. Knowing Unix tools is invaluable
  - A decent IDE
  - Debugger
  - Test frameworks + coverage reports
  - Google (or your favorite web search engine)
  - ChatGPT or LaMA



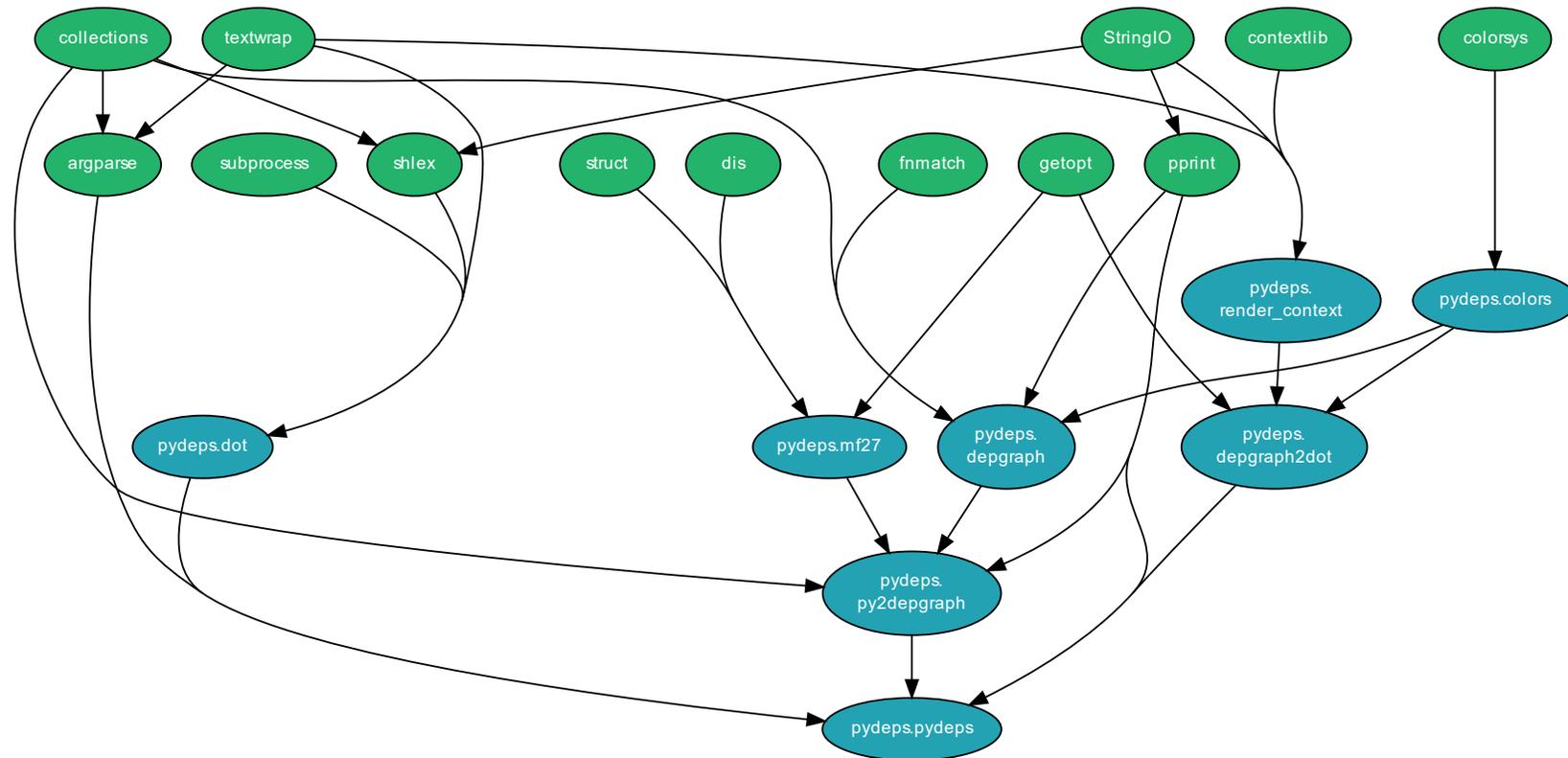
At the command line: **grep** and **find**!  
(Google for tutorials)

# Static Information Gathering: Use an IDE!

## Real software is too complex to keep in your head

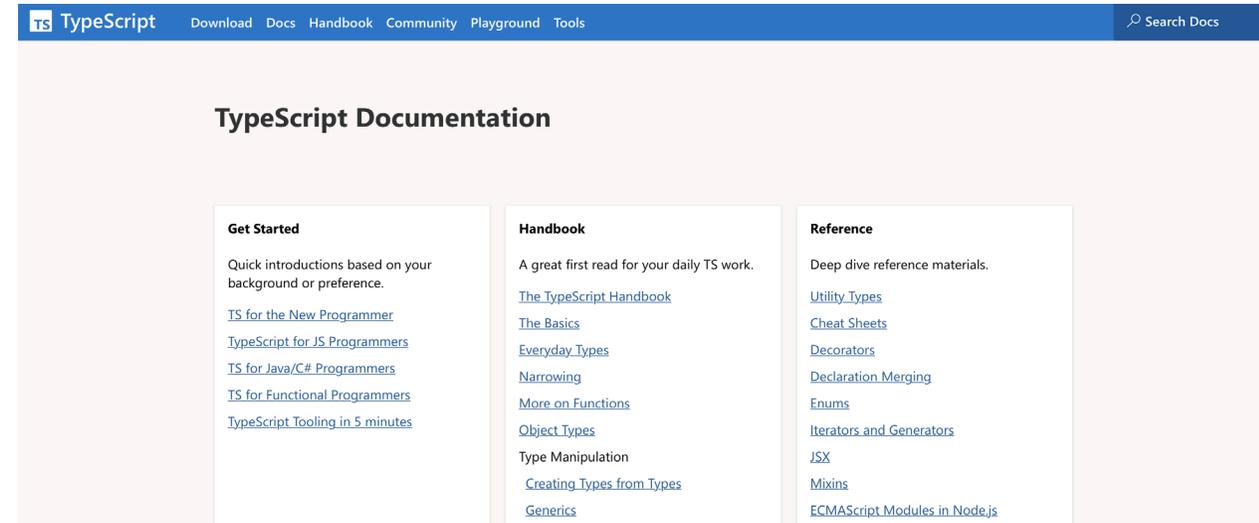


# Dependency maps



# Consider documentation and tutorials judiciously

- Great for discovering entry points!
- Can teach you about general structure, architecture (more on this later in the semester)
- Often out of date.
- As you gain experience, you will recognize more of these, and you will immediately know something about how the program works
- Also: discussion boards; issue trackers



The screenshot shows the TypeScript Documentation website. The header is blue with the TypeScript logo and navigation links: Download, Docs, Handbook, Community, Playground, Tools. A search bar is on the right. The main content area is titled "TypeScript Documentation" and is divided into three columns:

- Get Started**: Quick introductions based on your background or preference. Links include: [TS for the New Programmer](#), [TypeScript for JS Programmers](#), [TS for Java/C# Programmers](#), [TS for Functional Programmers](#), and [TypeScript Tooling in 5 minutes](#).
- Handbook**: A great first read for your daily TS work. Links include: [The TypeScript Handbook](#), [The Basics](#), [Everyday Types](#), [Narrowing](#), [More on Functions](#), [Object Types](#), [Type Manipulation](#), [Creating Types from Types](#), and [Generics](#).
- Reference**: Deep dive reference materials. Links include: [Utility Types](#), [Cheat Sheets](#), [Decorators](#), [Declaration Merging](#), [Enums](#), [Iterators and Generators](#), [JSX](#), [Mixins](#), and [ECMAScript Modules in Node.js](#).

# Discussion Boards and Issue Trackers

The screenshot shows the Stack Overflow search results page for the query "java on mac". The page includes a navigation bar with "stackoverflow", "About", "Products", "For Teams", a search bar containing "java on mac", and "Log in" and "Sign up" buttons. On the left, there is a sidebar with "Home", "PUBLIC", "Questions", "Tags", "Users", "Companies", "COLLECTIVES", "Explore Collectives", and "TEAMS". The main content area displays "Search Results" for "java on mac" with "500 results". Three search results are visible, each with a title, vote count, answer count, view count, and tags. The first result is "How to set or change the default Java (JDK) version on mac-OS?" with 1311 votes and 36 answers. The second is "How to install Java 8 on Mac" with 1271 votes and 34 answers. The third is "Where is Java Installed on Mac OS X?" with 861 votes and 20 answers. A "Hot Network Questions" section is also visible on the right side of the search results.

The screenshot shows the GitHub Issues page for the "sismics" repository. The page includes a navigation bar with "sismics / reader", a search bar, and buttons for "Code", "Issues", "Pull requests", "Actions", "Projects", "Wiki", "Security", and "Insights". The "Issues" section is active, showing "30 Open" and "131 Closed" issues. A list of issues is displayed, including "Rss feed", "Docker and database docker name", "Error on OPML import", "feature request: naive bayes ham / spam classifier", "file is broken msg on mac", "default credentials don't work", "Detect duplicate article in different feed", and "Android : Dark mode". Each issue entry includes a title, a brief description, and the date it was opened.

# Dynamic Information Gathering

## Change helps to inform and refine mental models

- Build it.
- Run it.
- Change it.
- Run it again.
- How did the behavior change?

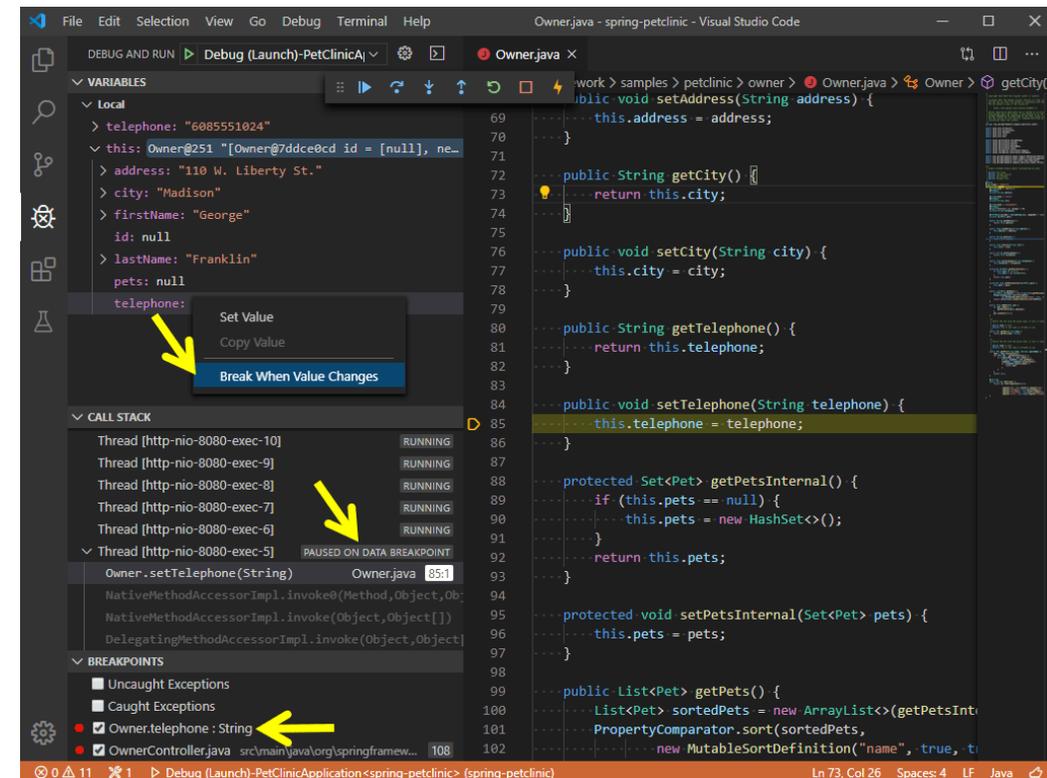


# How to start?

- Confirm that you can build and run the code.
  - Ideally both using the tests provided, and by hand.
- **Confirm that the code you are running is the code you built!**
- Confirm that you can make an externally visible change
- How? Where? Starting points:
  - Run an existing test, change it
  - Write a new test
  - Change the code, write or rerun a test that should notice the change
- Ask someone for help

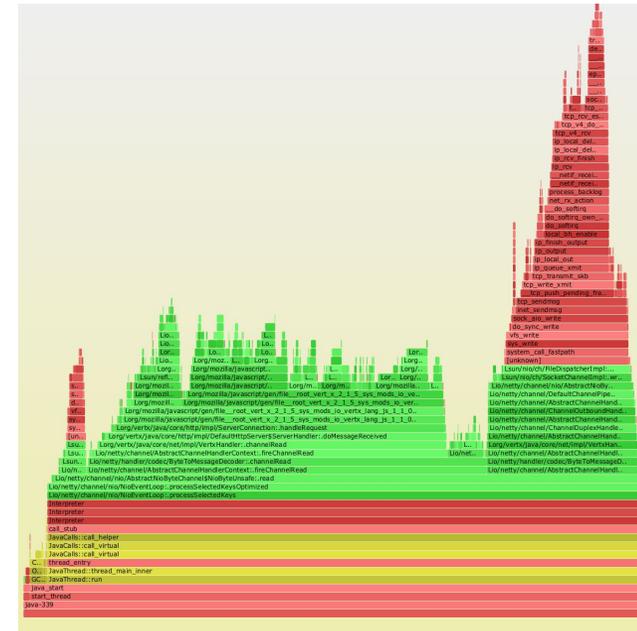
# Probes: Observe, control or “lightly” manipulate execution

- print(“this code is running!”)
- Structured logging
- Debuggers
  - Breakpoint, eval, step through / step over
  - (Some tools even support remote debugging)
- Delete debugging
- Chrome Developer Tools



# Runtime code analysis tools

- Collect runtime traces and visualize them
  - Flame graphs
  - Sequence diagrams
- Use judiciously



# Tip: Find a particular thing and trace the action backward

The screenshot shows the NodeBB interface. The 'Announcements' category is circled in black. A blue arrow points from this category to a post in the 'General Discussion' category. The post is by user '@admin' and says 'Hello'. The interface also shows statistics for each category: Topics and Posts counts, and a 'No new posts' indicator.

Category	Topics	Posts	Recent Post
Announcements	0	0	No new posts.
General Discussion	1	2	@admin Hello (about 16 hours ago)
Comments & Feedback	0	0	No new posts.
Blogs	0	0	No new posts.

Powered by [NodeBB](#) | [Contributors](#)

E.g.,  
Where do categories come from?  
How are they stored?  
How are they rendered?

# Remember...

- Reading and understanding code is one of the most important skills you should learn
- It's common to get stuck or feel overwhelmed. **Don't give up!**
- You are lucky! There are many tools available today



# Learning Goals

- Understand and scope the task of taking on and understanding a new and complex piece of existing software
- Appreciate the importance of configuring an effective IDE
- Contrast different types of code execution environments including local, remote, application, and libraries
- Enumerate both static and dynamic strategies for understanding and modifying a new codebase

# Tip: Document and share your findings!

- Update README and docs
  - Or better: use a Developer Wiki
  - Use [Mermaid](#) for diagrams
- Collaborate with others
- Include negative results, too!

