CEN 5016: Software Engineering

Spring 2026

University of Central Florida

Dr. Kevin Moran

Week 1 - Class 2: Software Archeology & Anthropology

Administrivia

- Let me know if you are not on Ed Discussions
- Assignment 1, Getting started with Git, GitHub, and Typescript is posted
  - Due Tuesday, January 20th at 11:59 pm
  - Use Megathread on Ed Discussions to ask questions
- Course Entrance Survey
  - Please complete by Friday at 11:59 pm

Goals for Today

- 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

Software Archeology & Anthropology

Context: Big Ole Pile of Code

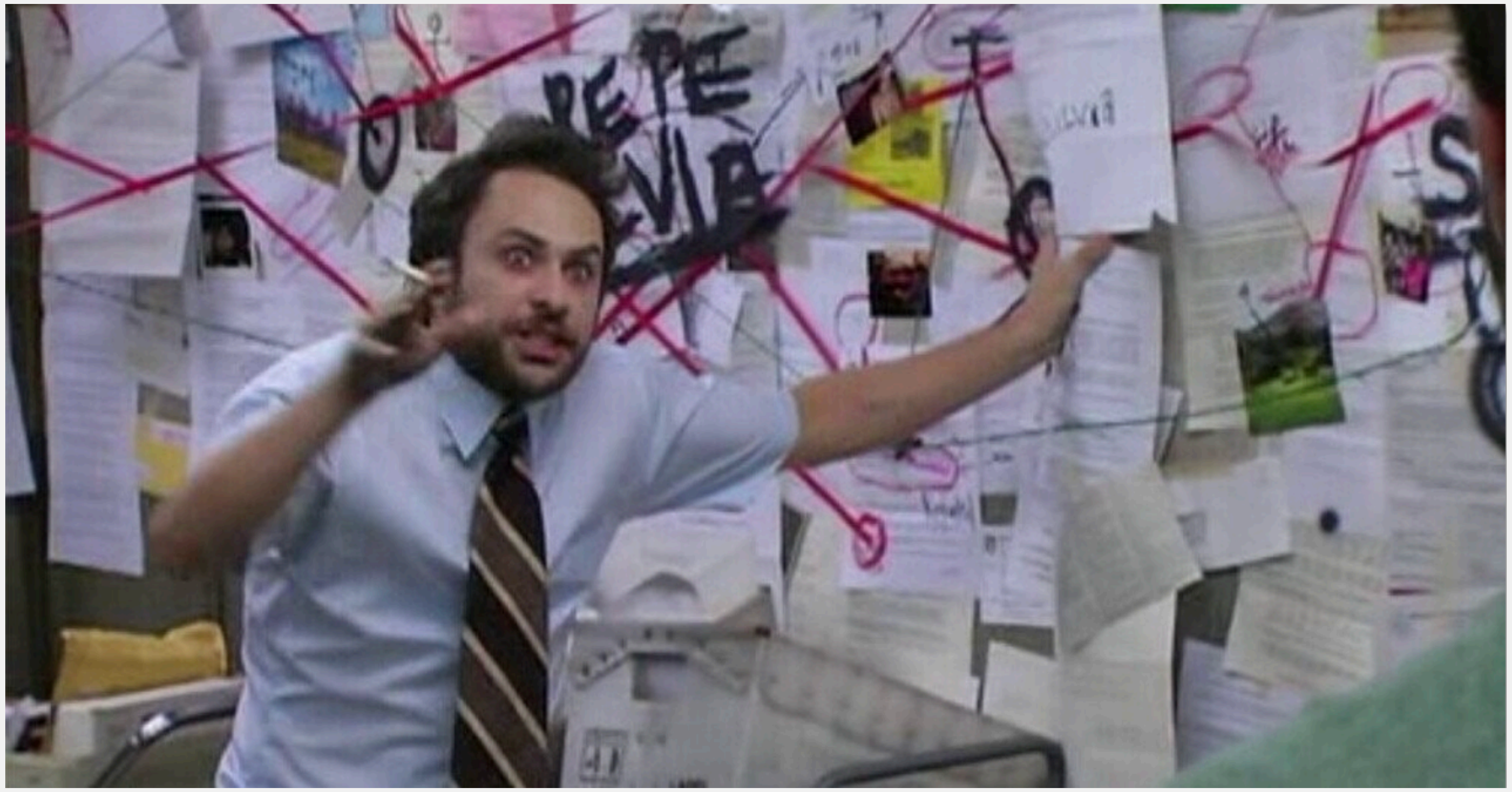
- Chances are that you will need to work with existing code at some point in your career…

The Challenge?

The Challenge?

You will never understand the entire system!

So Then: How do I tackle this Codebase?

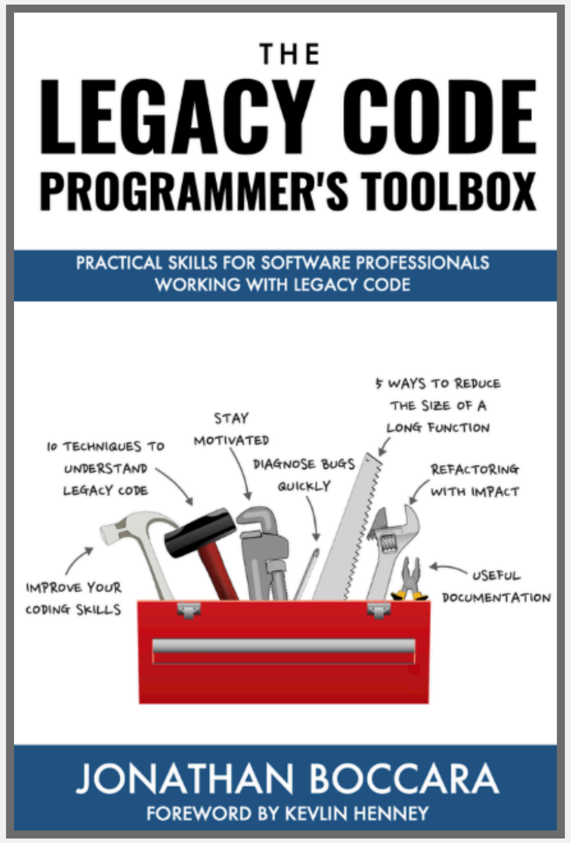


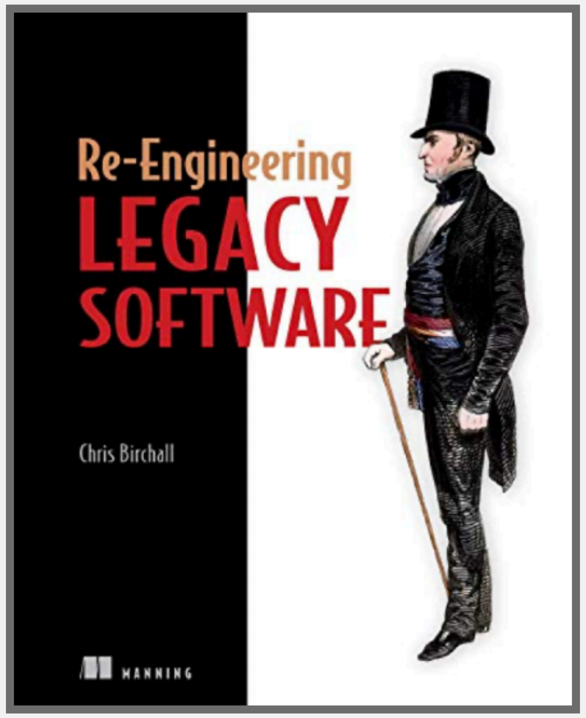
High-Level Strategies

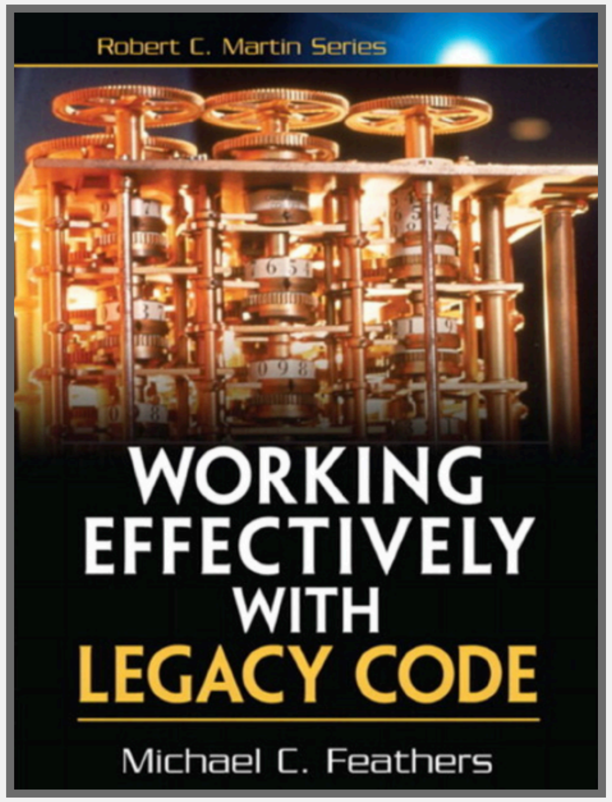
- Leverage your previous experiences (languages, technologies, patterns)
- Consult Documentation, white papers
- Talk to experts, code owners
- Follow best practices to build a working model of a system

Bad News: There are not many Helpful Resources

- *Working Effectively with Legacy Code.* Michael C. Feathers. 2004.
- *Re-Engineering Legacy Software.* Chris Birchall. 2016.
- *The Legacy Code Programmer's Toolbox.* Jonathan Boccara. 2019.

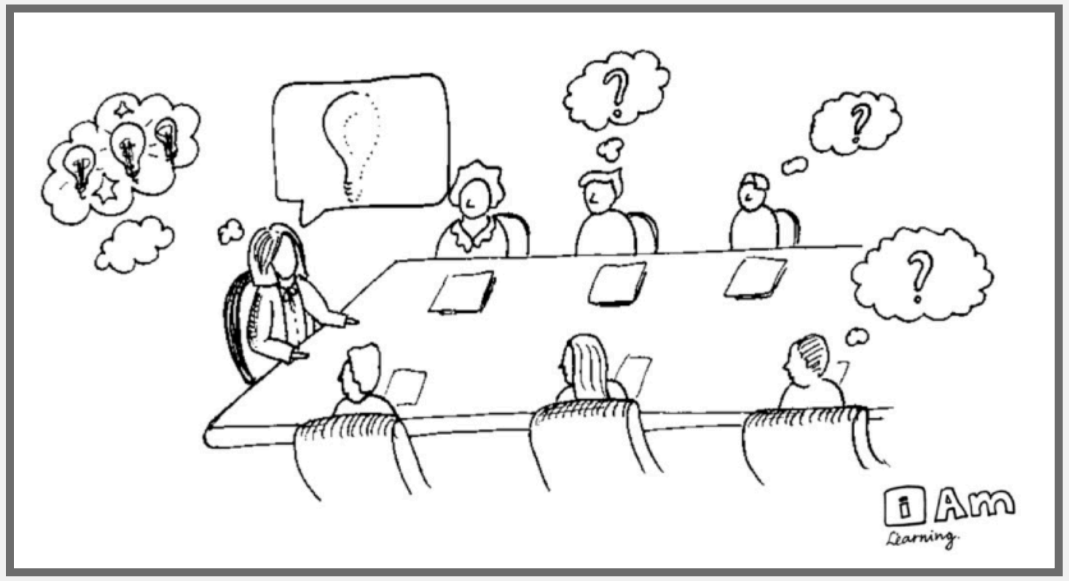

THE
LEGACY CODE
PROGRAMMER'S TOOLBOX
PRACTICAL SKILLS FOR SOFTWARE PROFESSIONALS WORKING WITH LEGACY CODE
10 TECHNIQUES TO UNDERSTAND LEGACY CODE
STAY MOTIVATED
DIAGNOSE BUGS QUICKLY
5 WAYS TO REDUCE THE SIZE OF A LONG FUNCTION
REFACTORING WITH IMPACT
IMPROVE YOUR CODING SKILLS
USEFUL DOCUMENTATION
JONATHAN BOCCARA
FOREWORD BY KEVLIN HENNEY



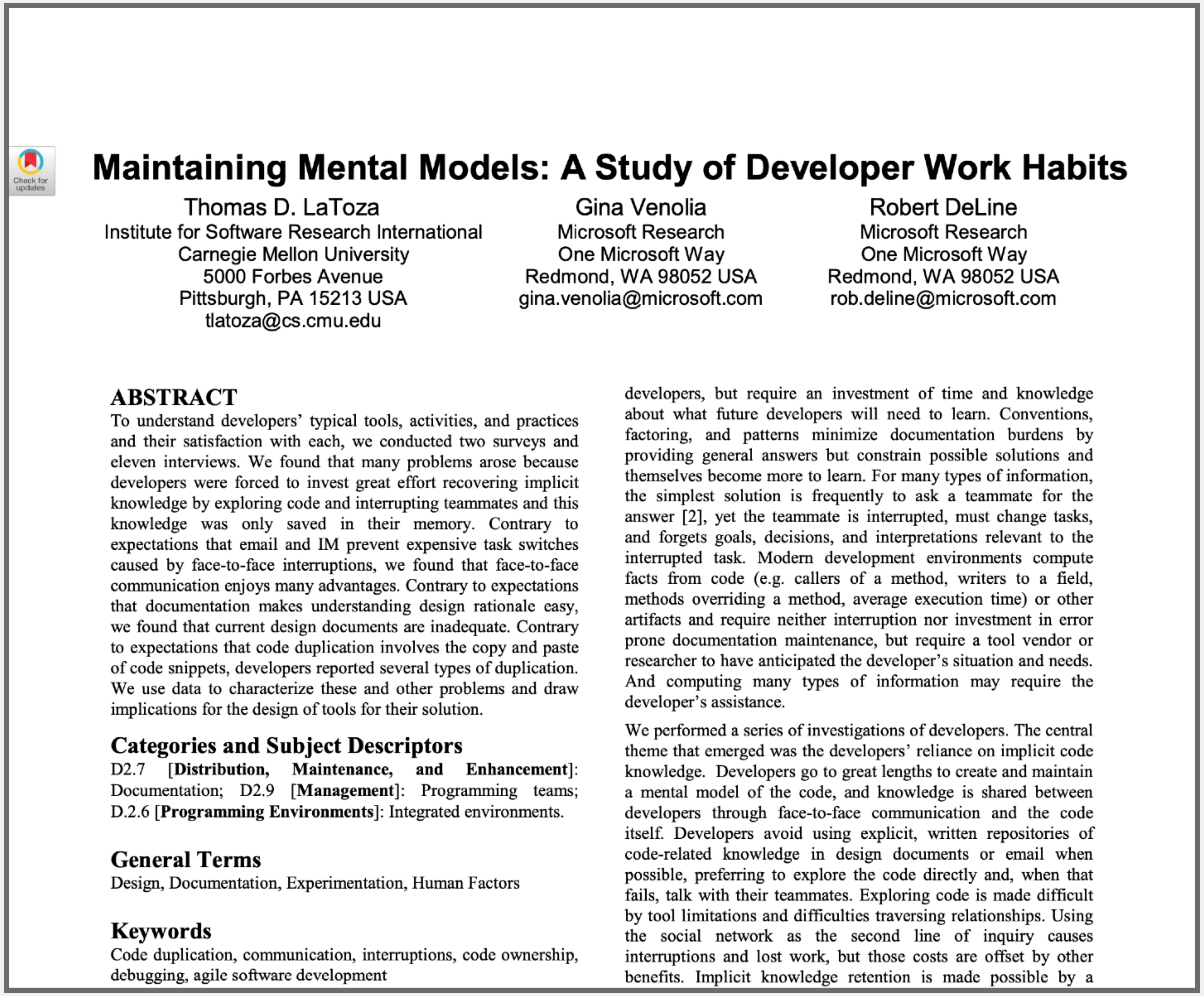


Why? Because of Tacit Knowledge

- *Tacit knowledge* or *implicit knowledge*—as opposed to formalized, codified or explicit knowledge—is knowledge that is difficult to express or extract; therefore it is more difficult to transfer to others by means of writing it down or verbalizing it.



Why? Because of Tacit Knowledge



Maintaining Mental Models: A Study of Developer Work Habits

Thomas D. LaToza
Institute for Software Research International
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
tlatoza@cs.cmu.edu

Gina Venolia
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
gina.venolia@microsoft.com

Robert DeLine
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
rob.deline@microsoft.com

ABSTRACT

To understand developers' typical tools, activities, and practices and their satisfaction with each, we conducted two surveys and eleven interviews. We found that many problems arose because developers were forced to invest great effort recovering implicit knowledge by exploring code and interrupting teammates and this knowledge was only saved in their memory. Contrary to expectations that email and IM prevent expensive task switches caused by face-to-face interruptions, we found that face-to-face communication enjoys many advantages. Contrary to expectations that documentation makes understanding design rationale easy, we found that current design documents are inadequate. Contrary to expectations that code duplication involves the copy and paste of code snippets, developers reported several types of duplication. We use data to characterize these and other problems and draw implications for the design of tools for their solution.

Categories and Subject Descriptors
D2.7 [**Distribution, Maintenance, and Enhancement**]: Documentation; D2.9 [**Management**]: Programming teams; D.2.6 [**Programming Environments**]: Integrated environments.

General Terms
Design, Documentation, Experimentation, Human Factors

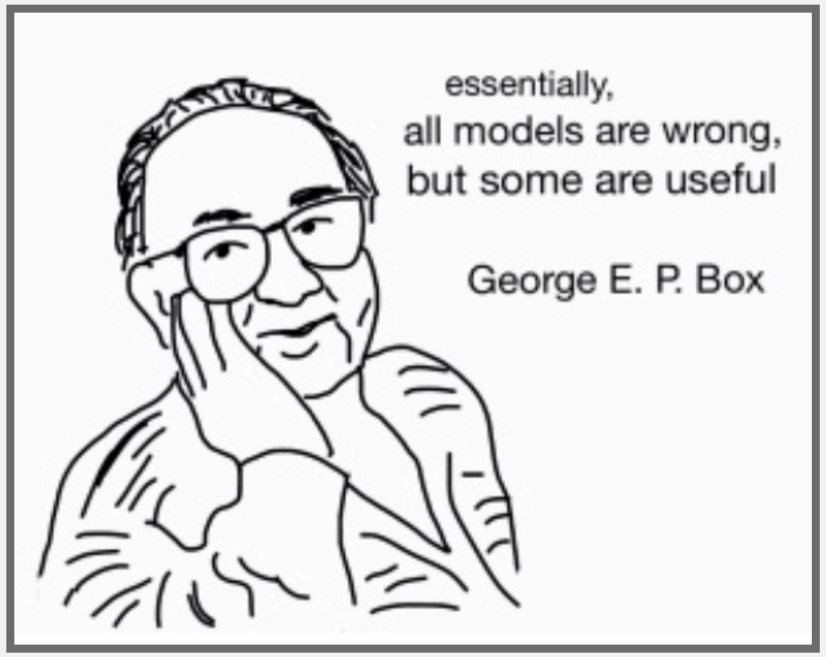
Keywords
Code duplication, communication, interruptions, code ownership, debugging, agile software development

developers, but require an investment of time and knowledge about what future developers will need to learn. Conventions, factoring, and patterns minimize documentation burdens by providing general answers but constrain possible solutions and themselves become more to learn. For many types of information, the simplest solution is frequently to ask a teammate for the answer [2], yet the teammate is interrupted, must change tasks, and forgets goals, decisions, and interpretations relevant to the interrupted task. Modern development environments compute facts from code (e.g. callers of a method, writers to a field, methods overriding a method, average execution time) or other artifacts and require neither interruption nor investment in error prone documentation maintenance, but require a tool vendor or researcher to have anticipated the developer's situation and needs. And computing many types of information may require the developer's assistance.

We performed a series of investigations of developers. The central theme that emerged was the developers' reliance on implicit code knowledge. Developers go to great lengths to create and maintain a mental model of the code, and knowledge is shared between developers through face-to-face communication and the code itself. Developers avoid using explicit, written repositories of code-related knowledge in design documents or email when possible, preferring to explore the code directly and, when that fails, talk with their teammates. Exploring code is made difficult by tool limitations and difficulties traversing relationships. Using the social network as the second line of inquiry causes interruptions and lost work, but those costs are offset by other benefits. Implicit knowledge retention is made possible by a

Today: How to tackle Codebases

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



Demo: Android Application

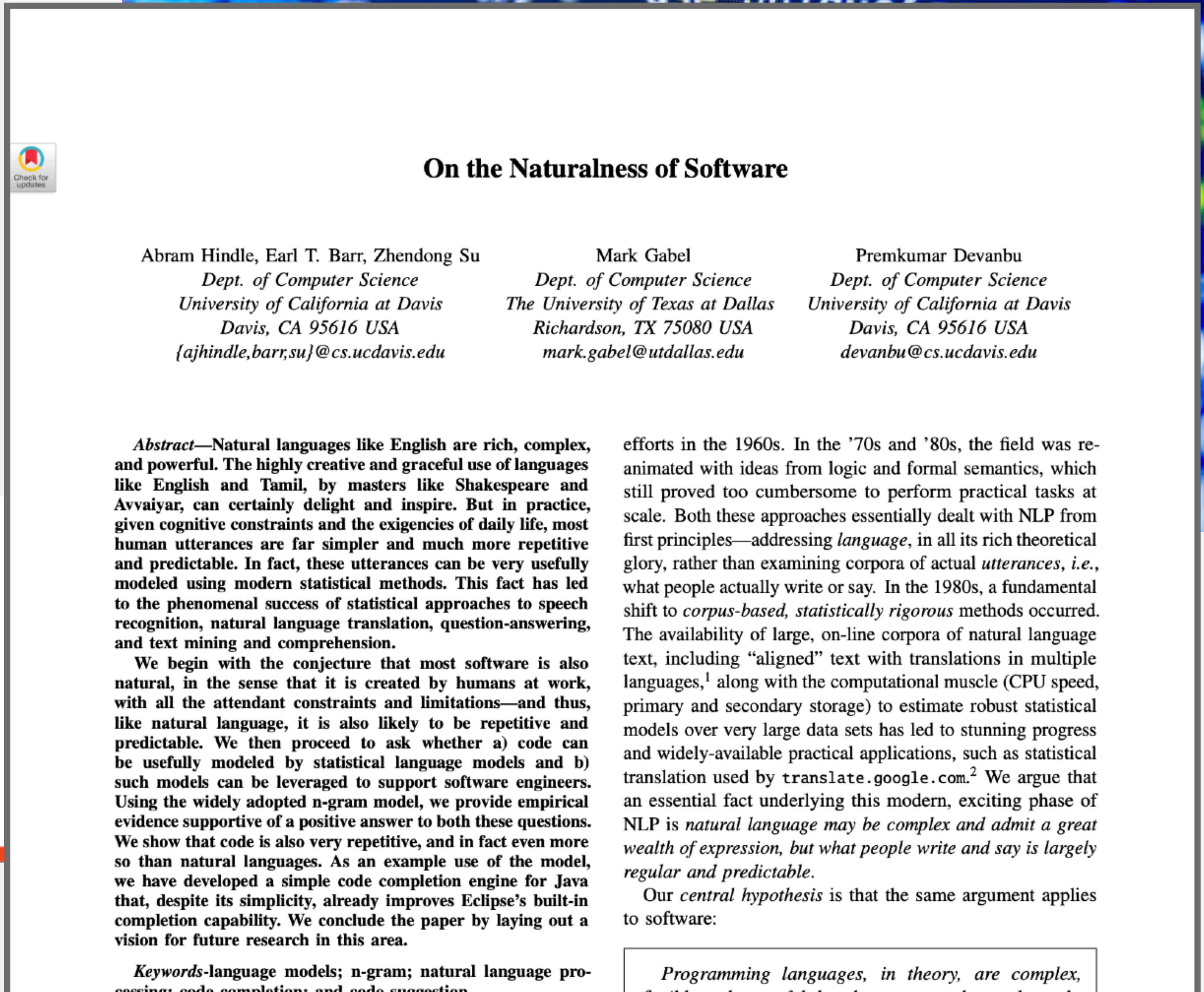
Steps to Understand a New Codebase

- 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))

Observation: Software is Full of Patterns

- File structure
- System architecture
- Code structure
- Names
- …

```
replaceAll(",", " ", a); a = a.repl
return a.split(" "); } $("#unique").cl
var a = array_from_string($("#fin").val
.val(), c = use_unique(
```



On the Naturalness of Software

Abram Hindle, Earl T. Barr, Zhendong Su
Dept. of Computer Science
University of California at Davis
Davis, CA 95616 USA
{ajhindle,barr,su}@cs.ucdavis.edu

Mark Gabel
Dept. of Computer Science
The University of Texas at Dallas
Richardson, TX 75080 USA
mark.gabel@utdallas.edu

Premkumar Devanbu
Dept. of Computer Science
University of California at Davis
Davis, CA 95616 USA
devanbu@cs.ucdavis.edu

***Abstract*—Natural languages like English are rich, complex, and powerful. The highly creative and graceful use of languages like English and Tamil, by masters like Shakespeare and Avvaiyar, can certainly delight and inspire. But in practice, given cognitive constraints and the exigencies of daily life, most human utterances are far simpler and much more repetitive and predictable. In fact, these utterances can be very usefully modeled using modern statistical methods. This fact has led to the phenomenal success of statistical approaches to speech recognition, natural language translation, question-answering, and text mining and comprehension.**

We begin with the conjecture that most software is also natural, in the sense that it is created by humans at work, with all the attendant constraints and limitations—and thus, like natural language, it is also likely to be repetitive and predictable. We then proceed to ask whether a) code can be usefully modeled by statistical language models and b) such models can be leveraged to support software engineers. Using the widely adopted n-gram model, we provide empirical evidence supportive of a positive answer to both these questions. We show that code is also very repetitive, and in fact even more so than natural languages. As an example use of the model, we have developed a simple code completion engine for Java that, despite its simplicity, already improves Eclipse's built-in completion capability. We conclude the paper by laying out a vision for future research in this area.

***Keywords*-language models; n-gram; natural language pro-**

efforts in the 1960s. In the '70s and '80s, the field was re-animated with ideas from logic and formal semantics, which still proved too cumbersome to perform practical tasks at scale. Both these approaches essentially dealt with NLP from first principles—addressing *language*, in all its rich theoretical glory, rather than examining corpora of actual *utterances*, *i.e.*, what people actually write or say. In the 1980s, a fundamental shift to *corpus-based, statistically rigorous* methods occurred. The availability of large, on-line corpora of natural language text, including "aligned" text with translations in multiple languages,[1] along with the computational muscle (CPU speed, primary and secondary storage) to estimate robust statistical models over very large data sets has led to stunning progress and widely-available practical applications, such as statistical translation used by `translate.google.com`.[2] We argue that an essential fact underlying this modern, exciting phase of NLP is *natural language may be complex and admit a great wealth of expression, but what people write and say is largely regular and predictable.*

Our *central hypothesis* is that the same argument applies to software:

Programming languages, in theory, are complex,

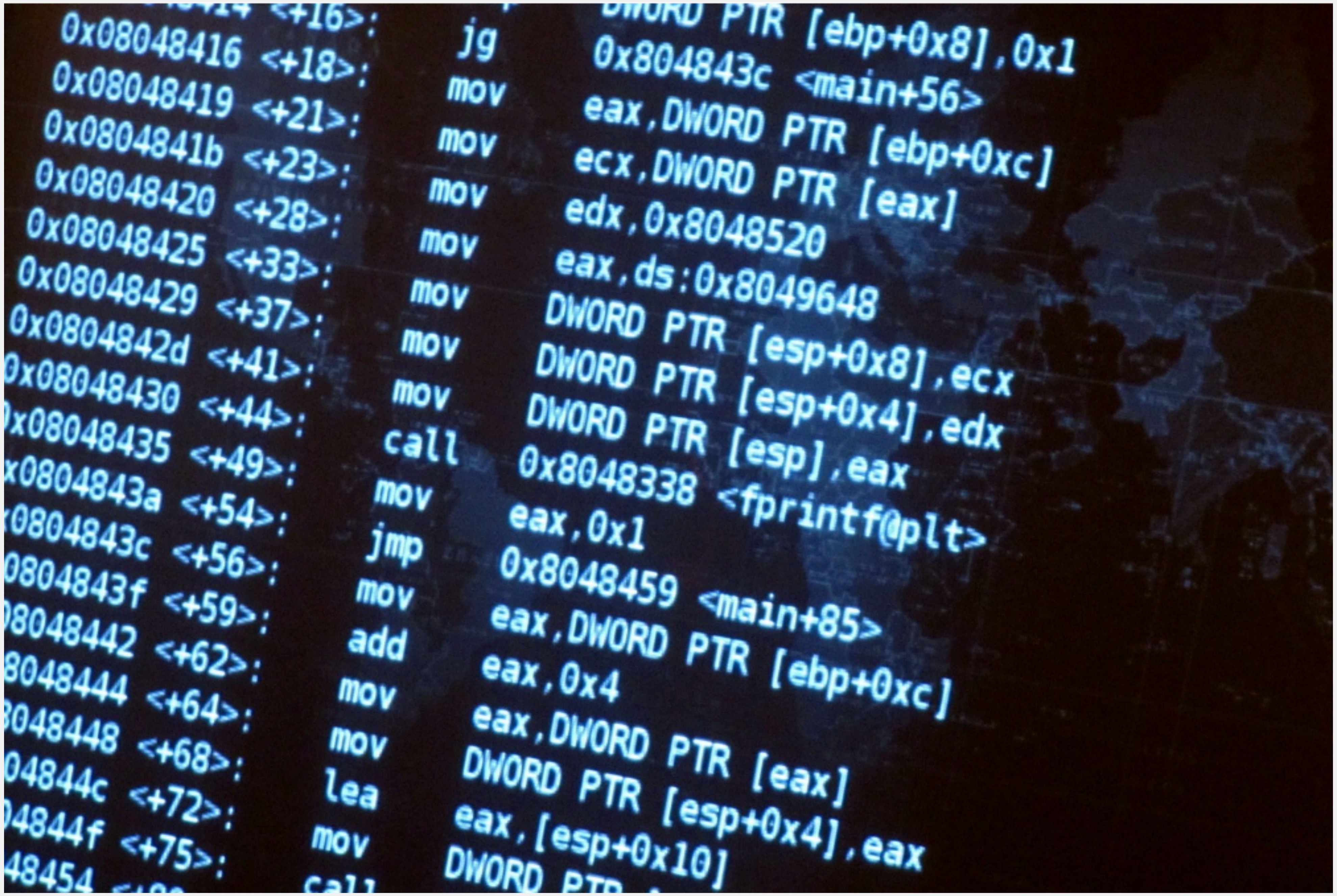
Observation: Software is Massively Redundant

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

Observation: Code Must Run to Do Stuff!

Observation: If Code Runs, it Must have a Beginning…

Observation: If Code Runs, it Must Exist



```
           <+16>:         DWORD PTR [ebp+0x8],0x1
0x08048416 <+18>:   jg    0x804843c <main+56>
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
x08048435 <+49>:   call  0x8048338 <fprintf@plt>
x0804843a <+54>:   mov   eax,0x1
0804843c <+56>:   jmp   0x8048459 <main+85>
0804843f <+59>:   mov   eax,DWORD PTR [ebp+0xc]
8048442 <+62>:   add   eax,0x4
8048444 <+64>:   mov   eax,DWORD PTR [eax]
8048448 <+68>:   mov   DWORD PTR [esp+0x4],eax
804844c <+72>:   lea   eax,[esp+0x10]
4844f <+75>:   mov   DWORD PTR
48454
```

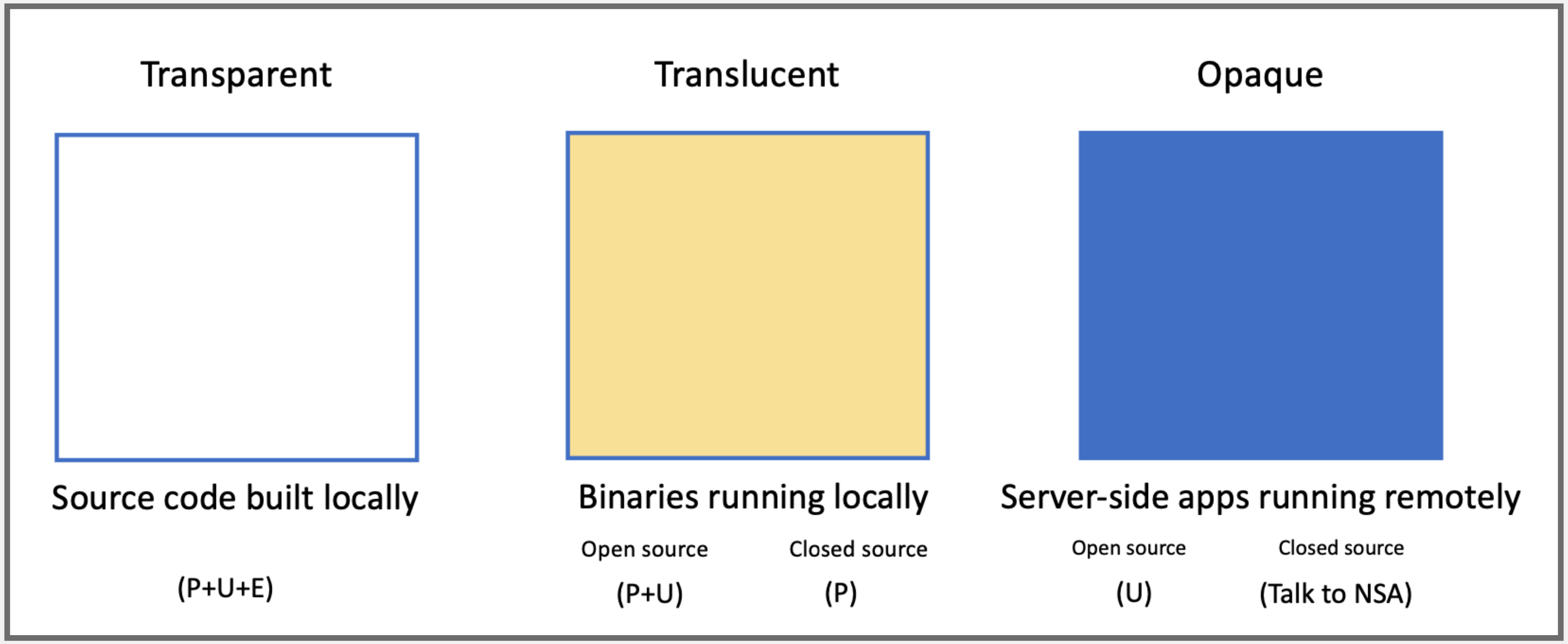
The Beginning: Entry Points

- 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

Code Must Exist: But Where?

- Locally installed programs: run cmd, OS launch, I/O events, etc.
  - Binaries (machine code) on your computer
- Local applications in dev: build + run, test, deploy (e.g., docker)
  - Source code in repository (+ dependencies)
- Web apps server-side: Browser sends HTTP request (e.g., GET, POST)
  - Code runs remotely (you can only observe outputs)
- Web apps client-side: Browser runs JavaScript, event handlers
  - Source code is downloaded and run locally (see: browser dev tools!)

Can Running Code be Probed/Understood/Edited?



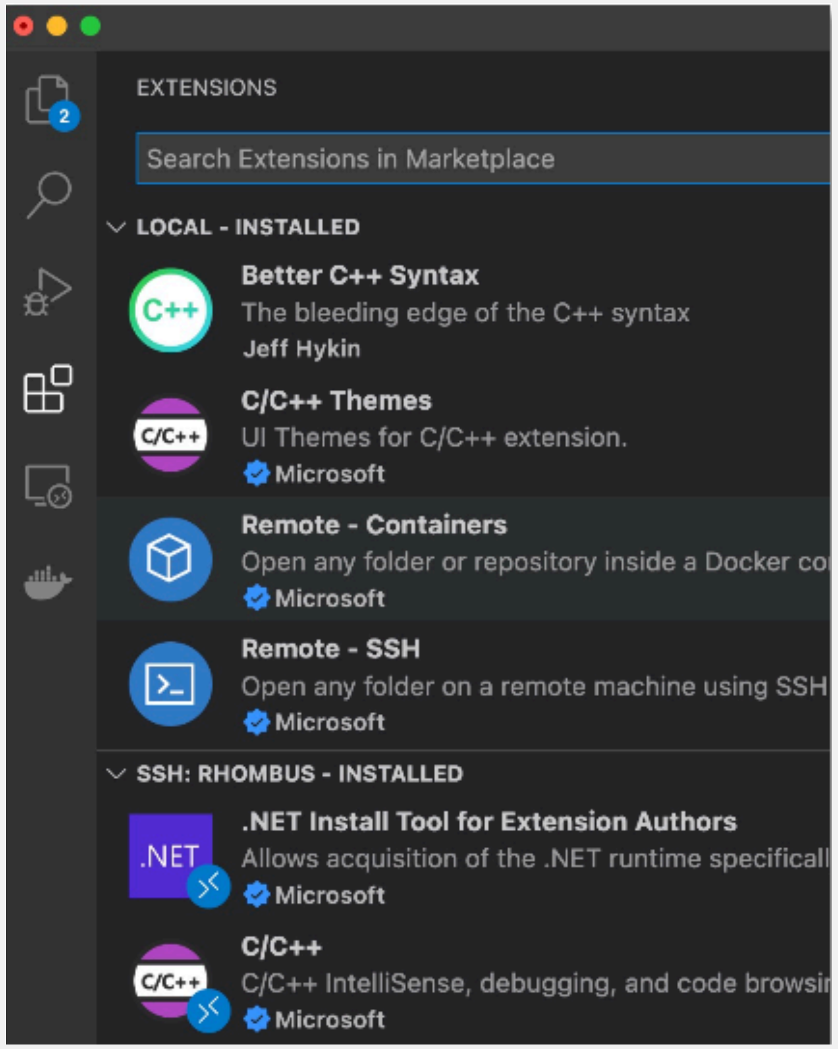
Creating a Model of Unfamiliar Code

Information Gathering

- Basic needs:
  - Code/file search and navigation
  - Code editing (probes)
  - Execution of code, tests
  - Observation of output (observation)
- At the command line: grep and find! (Google for tutorials)
- 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

Static Information Gathering: Use an IDE!

eclipse



EXTENSIONS

Search Extensions in Marketplace

LOCAL - INSTALLED

Better C++ Syntax
The bleeding edge of the C++ syntax
Jeff Hykin

C/C++ Themes
UI Themes for C/C++ extension.
Microsoft

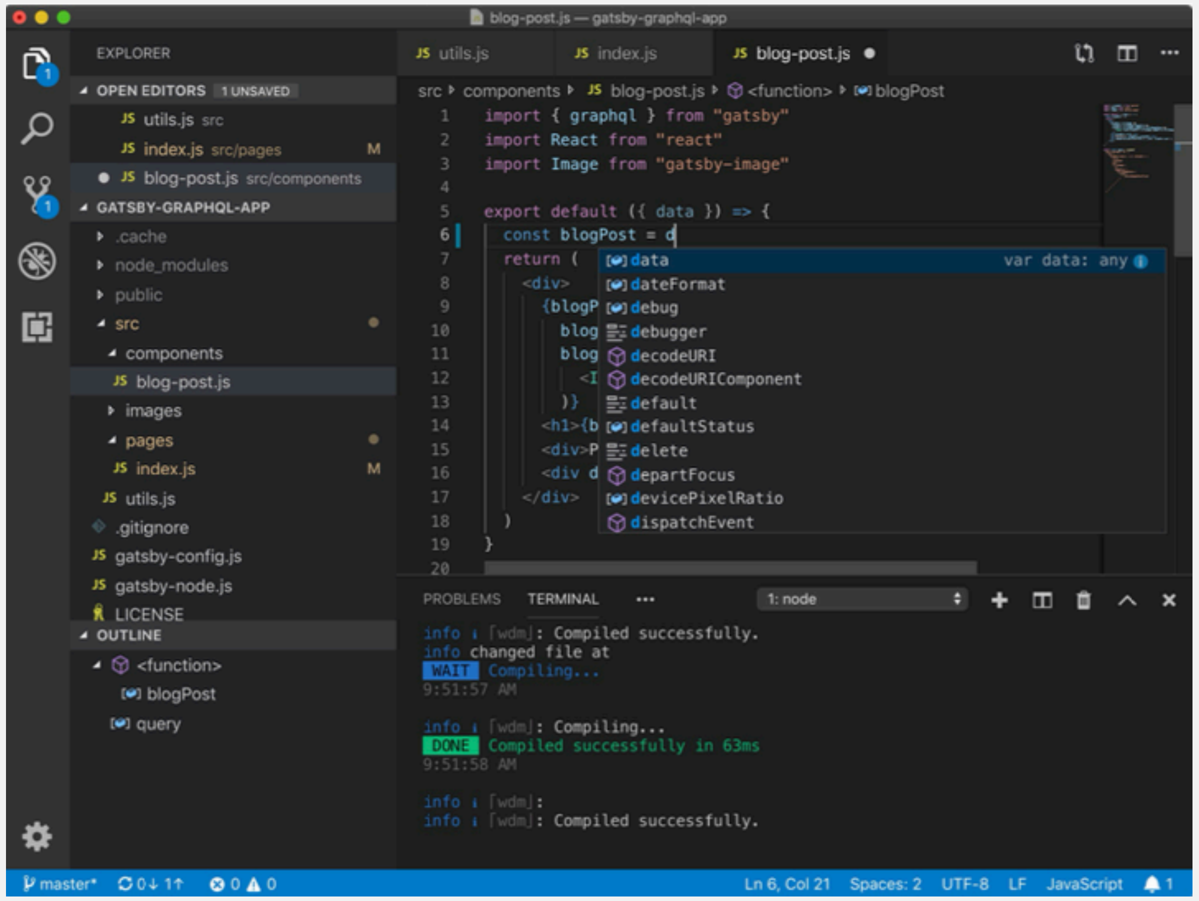
Remote - Containers
Open any folder or repository inside a Docker co
Microsoft

Remote - SSH
Open any folder on a remote machine using SSH
Microsoft

SSH: RHOMBUS - INSTALLED

.NET Install Tool for Extension Authors
Allows acquisition of the .NET runtime specificall
Microsoft

C/C++
C/C++ IntelliSense, debugging, and code browsi
Microsoft



blog-post.js — gatsby-graphql-app

EXPLORER
OPEN EDITORS 1 UNSAVED
- utils.js src
- index.js src/pages M
- ● blog-post.js src/components

GATSBY-GRAPHQL-APP
- .cache
- node_modules
- public
- src
  - components
    - blog-post.js
  - images
  - pages
    - index.js M
  - utils.js
- .gitignore
- gatsby-config.js
- gatsby-node.js
- LICENSE

OUTLINE
- <function>
  - blogPost
  - query

utils.js | index.js | blog-post.js ●

src ▸ components ▸ blog-post.js ▸ <function> ▸ blogPost

```
import { graphql } from "gatsby"
import React from "react"
import Image from "gatsby-image"

export default ({ data }) => {
  const blogPost = d
  return (
    <div>
      {blogP
        blog
        blog
          <I
      )}
      <h1>{b
      <div>P
      <div d
    </div>
  )
}
```

```
data                                   var data: any
dateFormat
debug
debugger
decodeURI
decodeURIComponent
default
defaultStatus
delete
departFocus
devicePixelRatio
dispatchEvent
```

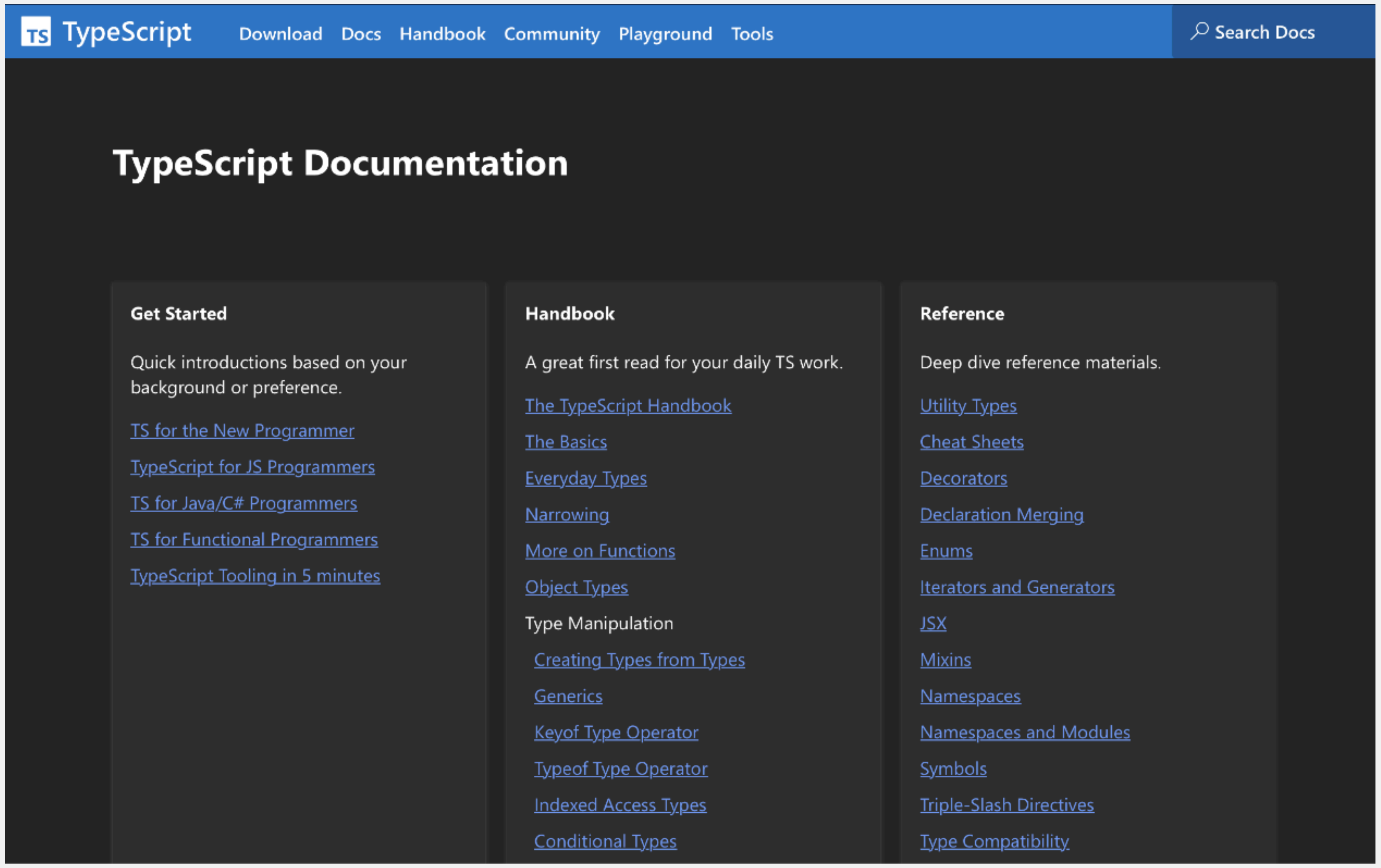
PROBLEMS | TERMINAL | ··· | 1: node

```
info ℹ ｢wdm｣: Compiled successfully.
info changed file at
 WAIT  Compiling...
9:51:57 AM

info ℹ ｢wdm｣: Compiling...
 DONE  Compiled successfully in 63ms
9:51:58 AM

info ℹ ｢wdm｣:
info ℹ ｢wdm｣: Compiled successfully.
```

master* 0↓ 1↑ ⊗ 0 ⚠ 0 Ln 6, Col 21 Spaces: 2 UTF-8 LF JavaScript 🔔 1

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



TypeScript Documentation

Get Started

Quick introductions based on your background or preference.

- TS for the New Programmer
- TypeScript for JS Programmers
- TS for Java/C# Programmers
- TS for Functional Programmers
- TypeScript Tooling in 5 minutes

Handbook

A great first read for your daily TS work.

- The TypeScript Handbook
- The Basics
- Everyday Types
- Narrowing
- More on Functions
- Object Types
- Type Manipulation
  - Creating Types from Types
  - Generics
  - Keyof Type Operator
  - Typeof Type Operator
  - Indexed Access Types
  - Conditional Types

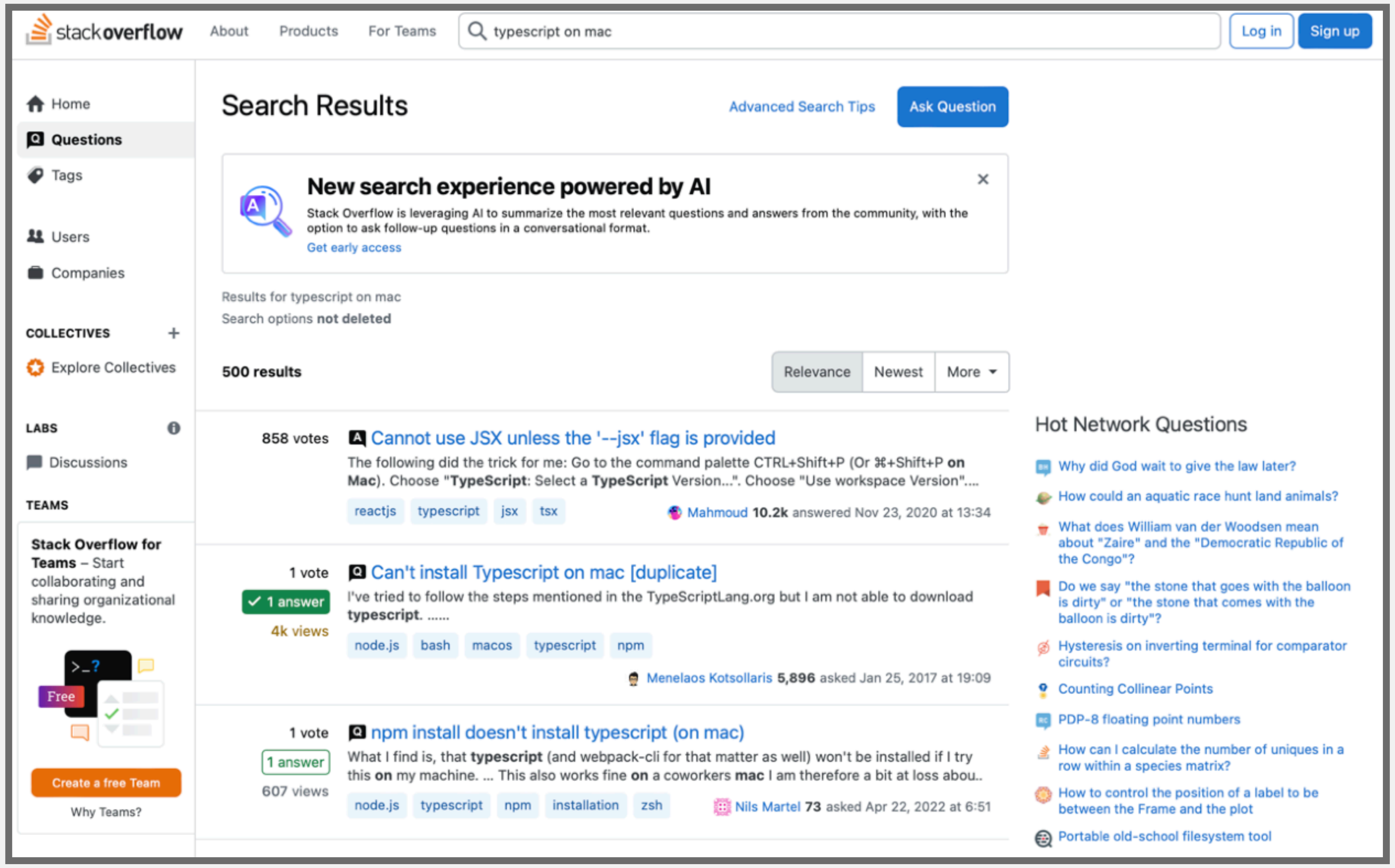
Reference

Deep dive reference materials.

- Utility Types
- Cheat Sheets
- Decorators
- Declaration Merging
- Enums
- Iterators and Generators
- JSX
- Mixins
- Namespaces
- Namespaces and Modules
- Symbols
- Triple-Slash Directives
- Type Compatibility

Discussion Boards and Issue Trackers

- Software is written by people.
- How can we talk to them?
- Fortunately, they probably aren't dead.
- So, you can report problems on GitHub.
- Or, ask them questions on StackOverflow.



Search Results

New search experience powered by AI

Stack Overflow is leveraging AI to summarize the most relevant questions and answers from the community, with the option to ask follow-up questions in a conversational format.

Get early access

Results for typescript on mac

Search options **not deleted**

500 results

858 votes — Cannot use JSX unless the '--jsx' flag is provided

The following did the trick for me: Go to the command palette CTRL+Shift+P (Or ⌘+Shift+P **on Mac**). Choose "**TypeScript**: Select a **TypeScript** Version...". Choose "Use workspace Version"....

reactjs typescript jsx tsx — Mahmoud **10.2k** answered Nov 23, 2020 at 13:34

1 vote, ✓ 1 answer, 4k views — Can't install Typescript on mac [duplicate]

I've tried to follow the steps mentioned in the TypeScriptLang.org but I am not able to download **typescript**. ......

node.js bash macos typescript npm — Menelaos Kotsollaris **5,896** asked Jan 25, 2017 at 19:09

1 vote, 1 answer, 607 views — npm install doesn't install typescript (on mac)

What I find is, that **typescript** (and webpack-cli for that matter as well) won't be installed if I try this **on** my machine. ... This also works fine **on** a coworkers **mac** I am therefore a bit at loss abou..

node.js typescript npm installation zsh — Nils Martel **73** asked Apr 22, 2022 at 6:51

Stack Overflow for Teams – Start collaborating and sharing organizational knowledge.

Hot Network Questions

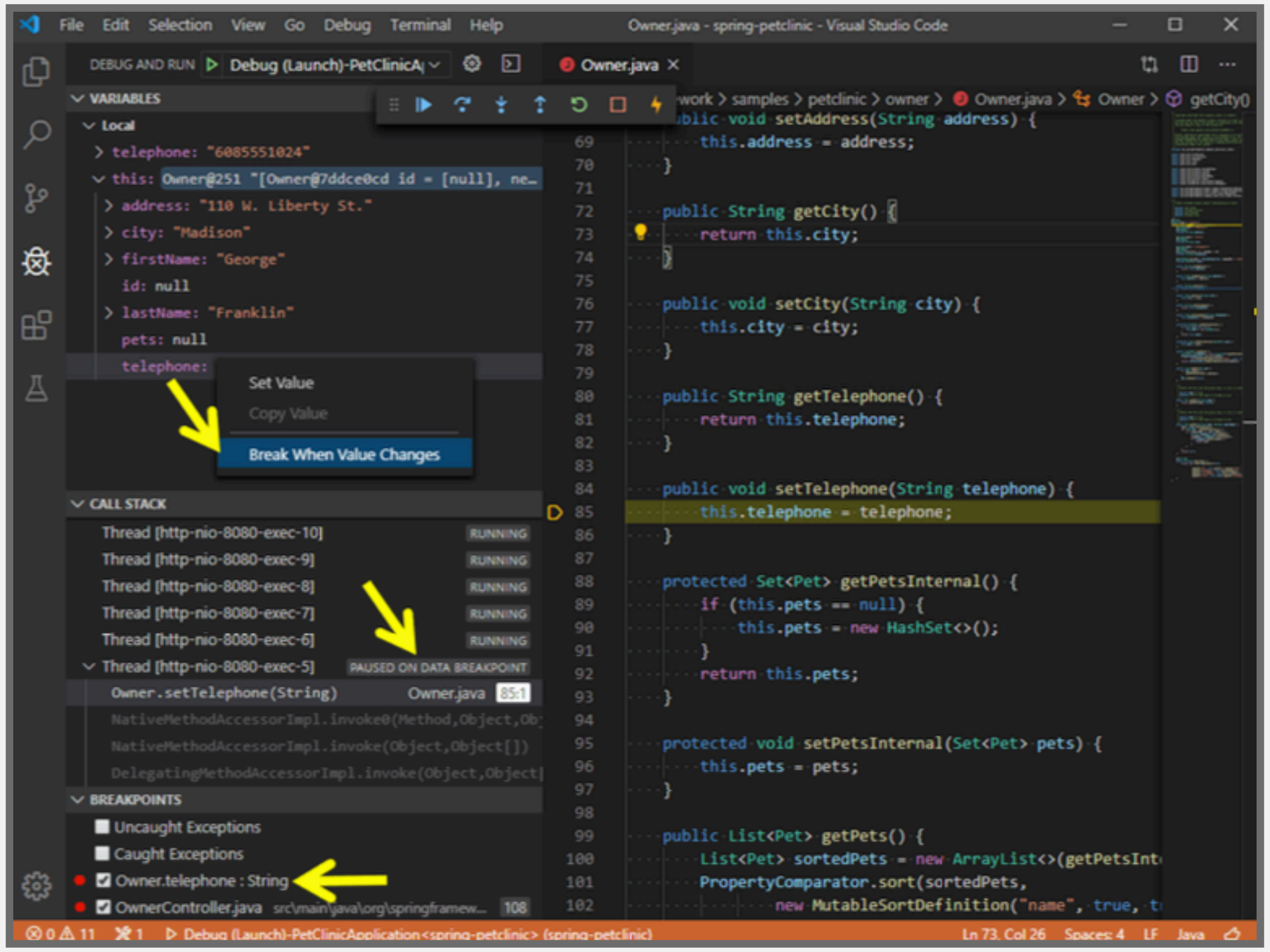
- Why did God wait to give the law later?
- How could an aquatic race hunt land animals?
- What does William van der Woodsen mean about "Zaire" and the "Democratic Republic of the Congo"?
- Do we say "the stone that goes with the balloon is dirty" or "the stone that comes with the balloon is dirty"?
- Hysteresis on inverting terminal for comparator circuits?
- Counting Collinear Points
- PDP-8 floating point numbers
- How can I calculate the number of uniques in a row within a species matrix?
- How to control the position of a label to be between the Frame and the plot
- Portable old-school filesystem tool

Dynamic Information Gathering

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

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


File
Edit
Selection
View
Go
Debug
Terminal
Help
Owner.java - spring-petclinic - Visual Studio Code
DEBUG AND RUN
Debug (Launch)-PetClinicA
Owner.java
work > samples > petclinic > owner > Owner.java > Owner > getCity()
VARIABLES
Local
Set Value
Copy Value
Break When Value Changes
CALL STACK
BREAKPOINTS

```
> telephone: "6085551024"
v this: Owner@251 "[Owner@7ddce0cd id = [null], ne…
  > address: "110 W. Liberty St."
  > city: "Madison"
  > firstName: "George"
    id: null
  > lastName: "Franklin"
    pets: null
    telephone:
```

```
Thread [http-nio-8080-exec-10]    RUNNING
Thread [http-nio-8080-exec-9]     RUNNING
Thread [http-nio-8080-exec-8]     RUNNING
Thread [http-nio-8080-exec-7]     RUNNING
Thread [http-nio-8080-exec-6]     RUNNING
Thread [http-nio-8080-exec-5]     PAUSED ON DATA BREAKPOINT
  Owner.setTelephone(String)      Owner.java 85:1
  NativeMethodAccessorImpl.invoke0(Method,Object,Ob
  NativeMethodAccessorImpl.invoke(Object,Object[])
  DelegatingMethodAccessorImpl.invoke(Object,Object
```

```
☐ Uncaught Exceptions
☐ Caught Exceptions
☑ Owner.telephone : String
☑ OwnerController.java  src\main\java\org\springframew…  108
```

```java
    public void setAddress(String address) {
        this.address = address;
    }

    public String getCity() {
        return this.city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getTelephone() {
        return this.telephone;
    }

    public void setTelephone(String telephone) {
        this.telephone = telephone;
    }

    protected Set<Pet> getPetsInternal() {
        if (this.pets == null) {
            this.pets = new HashSet<>();
        }
        return this.pets;
    }

    protected void setPetsInternal(Set<Pet> pets) {
        this.pets = pets;
    }

    public List<Pet> getPets() {
        List<Pet> sortedPets = new ArrayList<>(getPetsInt
        PropertyComparator.sort(sortedPets,
                new MutableSortDefinition("name", true, t
```

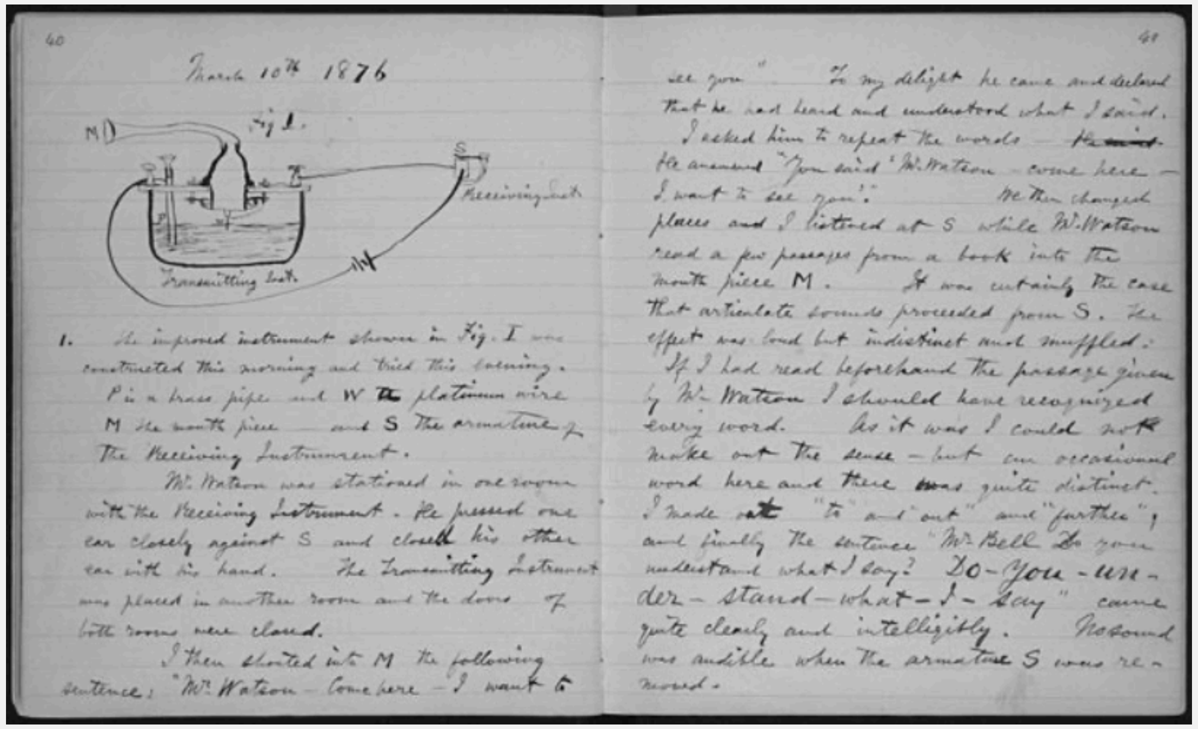
```
⊗ 0 ⚠ 11   1   Debug (Launch)-PetClinicApplication<spring-petclinic> (spring-petclinic)   Ln 73, Col 26   Spaces: 4   LF   Java
```

Step 0: Sanity Check Basic Model + Hypotheses

- *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*

Document and Share Your Findings!

- Update README and docs
  - Or better: use a Developer Wiki
  - Use Mermaid for diagrams
- Screencast on Twitch
- Collaborate with others
- Include negative results, too!



40

March 10th 1876

Fig. 1.
M
S
Receiving Inst.
Transmitting Inst.

1. The improved instrument shown in Fig. I was constructed this morning and tried this evening.
P is a brass pipe and W ~~the~~ platinum wire M the mouth piece and S the armature of The Receiving Instrument.

Mr. Watson was stationed in one room with the Receiving Instrument. He pressed one ear closely against S and closed his other ear with his hand. The Transmitting Instrument was placed in another room and the doors of both rooms were closed.

I then shouted into M the following sentence: "Mr. Watson — Come here — I want to

41

see you." To my delight he came and declared that he had heard and understood what I said.
I asked him to repeat the words — ~~He said~~ He answered "You said 'Mr. Watson — come here — I want to see you.'" We then changed places and I listened at S while Mr. Watson read a few passages from a book into the mouth piece M. It was certainly the case that articulate sounds proceeded from S. The effect was loud but indistinct and muffled.
If I had read beforehand the passage given by Mr. Watson I should have recognized every word. As it was I could not make out the sense — but an occasional word here and there was quite distinct.
I made ~~out~~ "to" and "out" and "further"; and finally the sentence "Mr. Bell Do you understand what I say? Do-you-un-der-stand-what-I-say" came quite clearly and intelligibly. No sound was audible when the armature S was removed.