

CS 4530

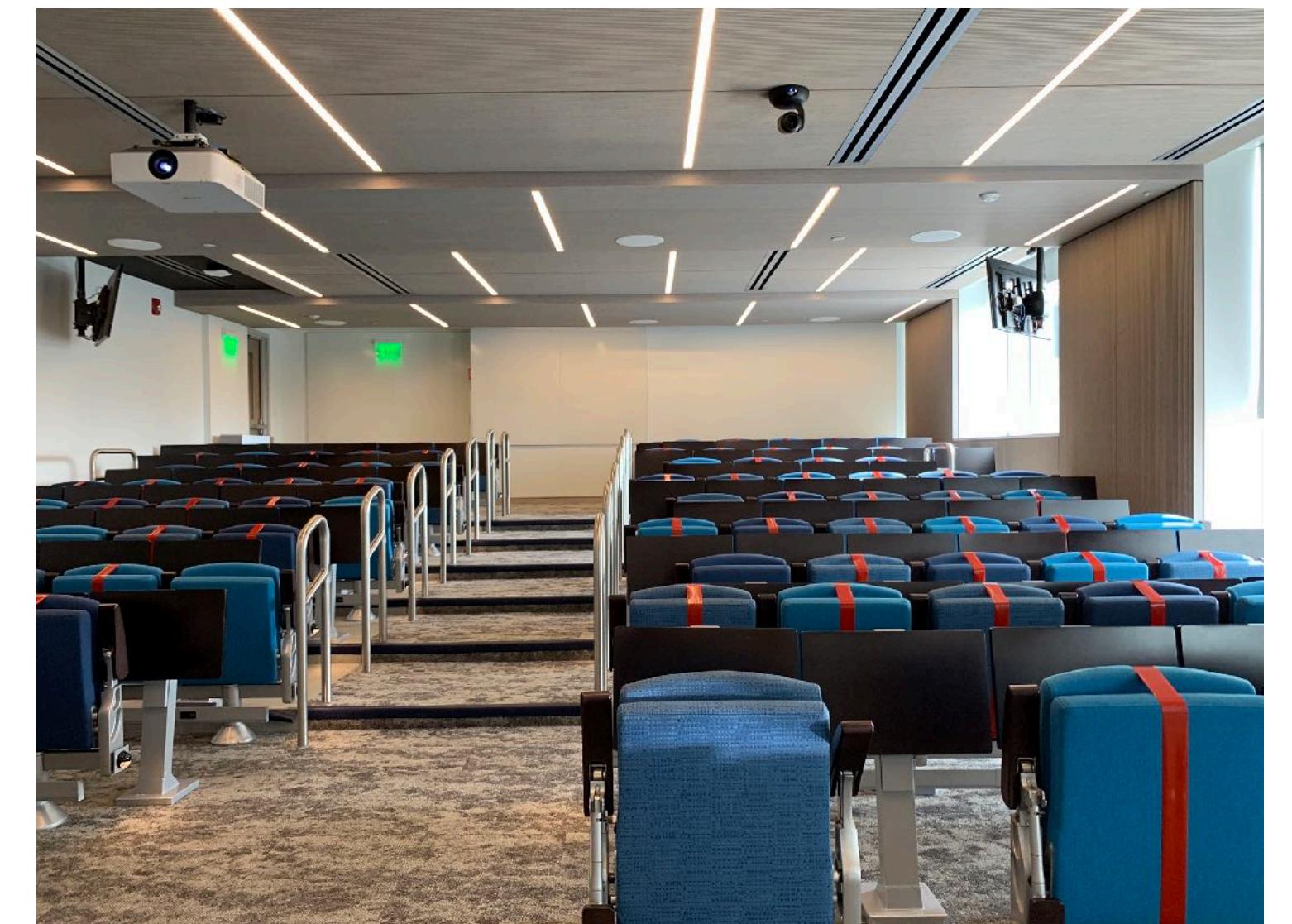
Software Engineering

Lecture 14 - Analysis/Refactoring; Covey.Town Internals

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

Zoom Mechanics

- Recording: This meeting is being recorded
- If you feel comfortable having your camera on, please do so! If not: a photo?
- I can see the zoom chat while lecturing, slack while you're in breakout rooms
- If you have a question or comment, please either:
 - “Raise hand” - I will call on you
 - Write “Q: <my question>” in chat - I will answer your question, and might mention your name and ask you a follow-up to make sure your question is addressed
 - Write “SQ: <my question>” in chat - I will answer your question, and not mention your name or expect you to respond verbally



Today's Agenda

Administrative:

Project Plan due tomorrow!

HW4 due next Friday

Today's session:

Static analysis + refactoring review + discussion

Static Analysis Review

- Find likely bugs, but programming practices (eslint + LGTM/codeql)
- Extremely difficult to *prove* that programs are correct
- This is an enormous research area

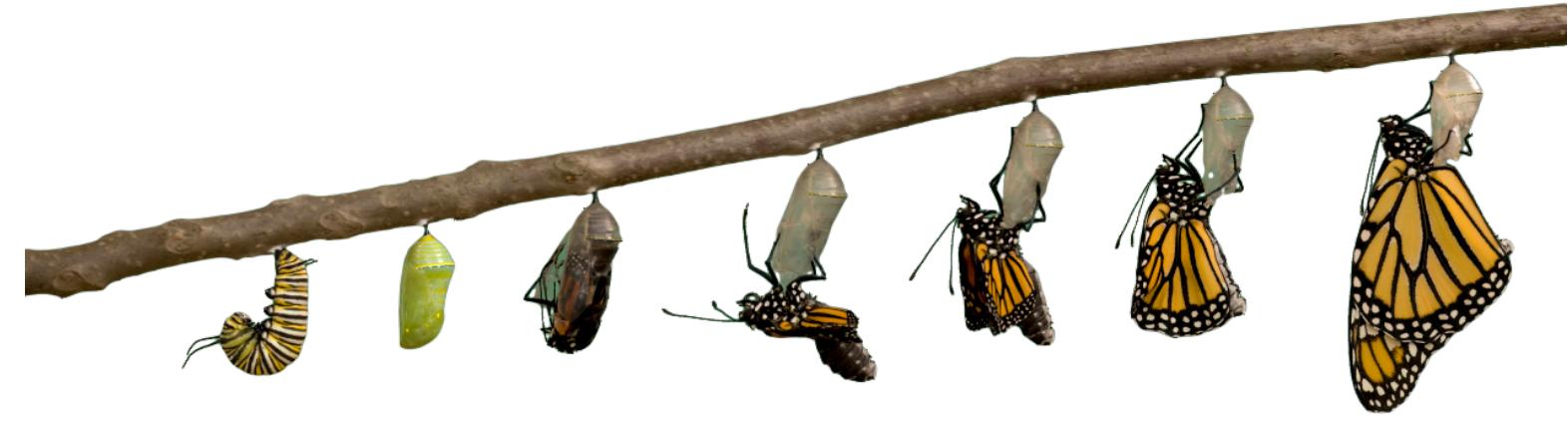
Refactoring

Martin Fowler



**“Any fool can write code that a computer can understand.
Good programmers write code that humans can understand.”**

Why Refactor?



- requirements have changed, and a different design is needed
- design needs to be more **flexible** (so new features can be added)
 - design patterns are often a target for refactoring
- address sloppiness by programmers

Example Refactoring

Consolidating duplicate conditional fragments

Original Code

```
if (isSpecialDeal()) {  
    total = price * 0.95;  
    send()  
} else {  
    total = price * 0.98;  
    send()  
}
```

Refactored Code

```
if (isSpecialDeal()) {  
    total = price * 0.95;  
} else {  
    total = price * 0.98;  
}  
send()
```

Observations

- **small incremental steps** that preserve program behavior
- most steps are so simple that they can be **automated**
 - automation limited in complex cases
- refactoring does not always proceed “in a straight line”
 - sometimes, undo a step you did earlier...
 - ...when you have insights for a better design

When to refactor?

Refactoring is incremental redesign

- Acknowledge that it will be difficult to get design right the first time
- When adding new functionality, fixing a bug, doing code review, or any time
- Refactoring evolves design in increments
- Refactoring reduces technical debt
- What do you refactor?

Code Smells

Mysterious Name

“We may fantasize about being International Men of Mystery, but our code needs to be mundane and clear”

- Martin Fowler on “Mysterious Name”

Code Smells

Shotgun Surgery

**“When the changes are all over the place,
they are hard to find, and it’s easy to miss
an important change.”**

- Martin Fowler on “Shotgun Surgery”

Code Smells

A complete list (links to book!)

Mysterious Name

Duplicated Code

Long Function

Long Parameter List

Global Data

Mutable Data

Divergent Change

Shotgun Surgery

Feature Envy

Data Clumps

Primitive Obsession

Repeated Switches

Loops

Lazy Element

Speculative Generality

Temporary Field

Message Chains

Middle Man

Insider Trading

Large Class

Alternative Classes with Different Interfaces

Data Class

Refused Bequest

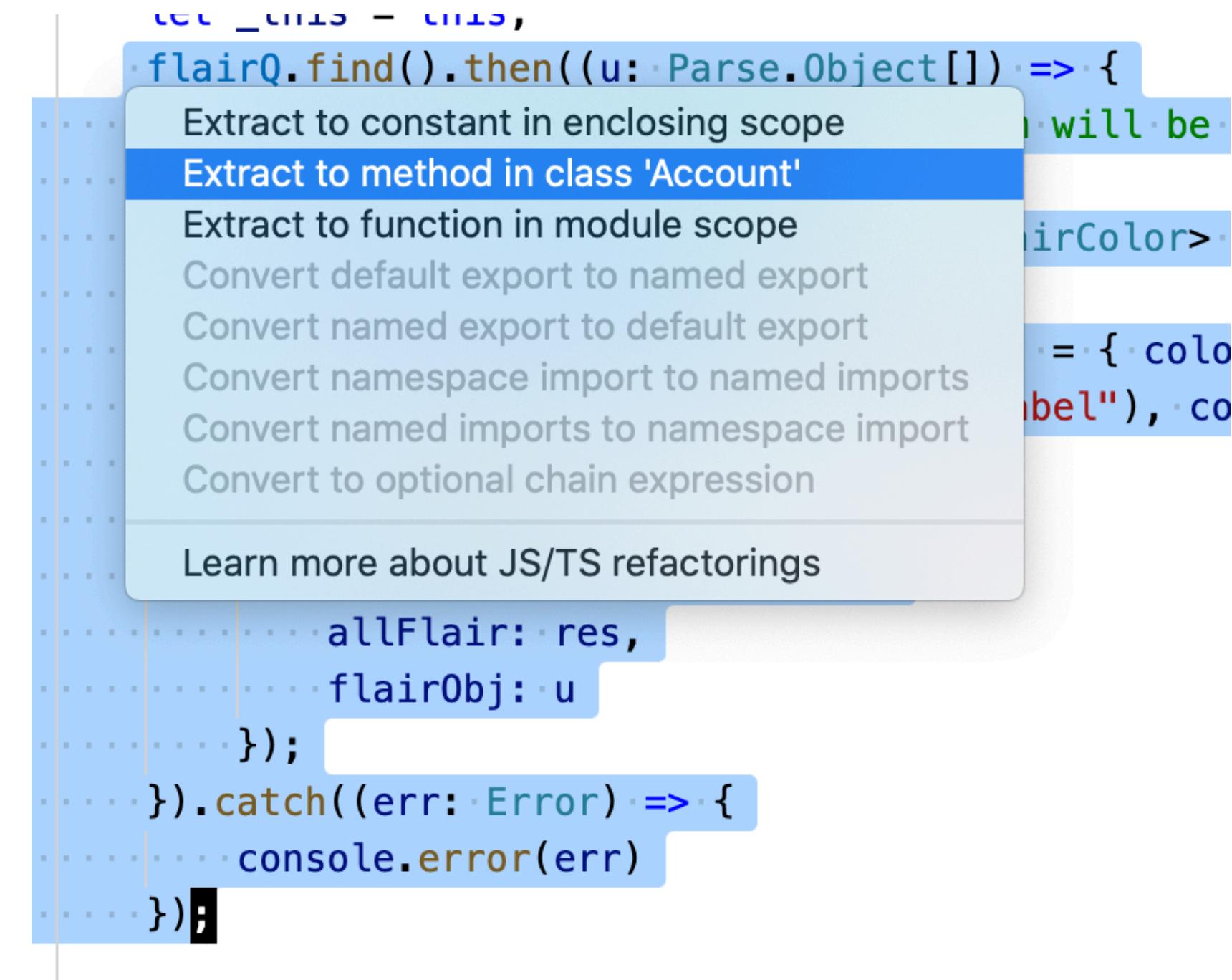
“Local” Refactorings

Rename	rename variables, fields methods, classes, packages provide better intuition for the renamed element's purpose
Extract Method	extract statements into a new method enables reuse; avoid cut-and-paste programming improve readability
Inline Method	replace a method call with the method's body often useful as intermediate step
Extract Local	introduce a new local variable for a designated expression
Inline Local	replace a local variable with the expression that defines its value
Change Method Signature	reorder a method's parameters
Encapsulate Field	introduce getter/setter methods
Convert Local Variable to Field	convert local variable to field sometimes useful to enable application of Extract Method

Type-Related Refactorings

Generalize Declared Type	replace the type of a declaration with a more general type
Extract Interface	create a new interface, and update declarations to use it where possible
Pull Up Members	move methods and fields to a superclass
Infer Generic Type Arguments	infer type arguments for “raw” uses of generic types

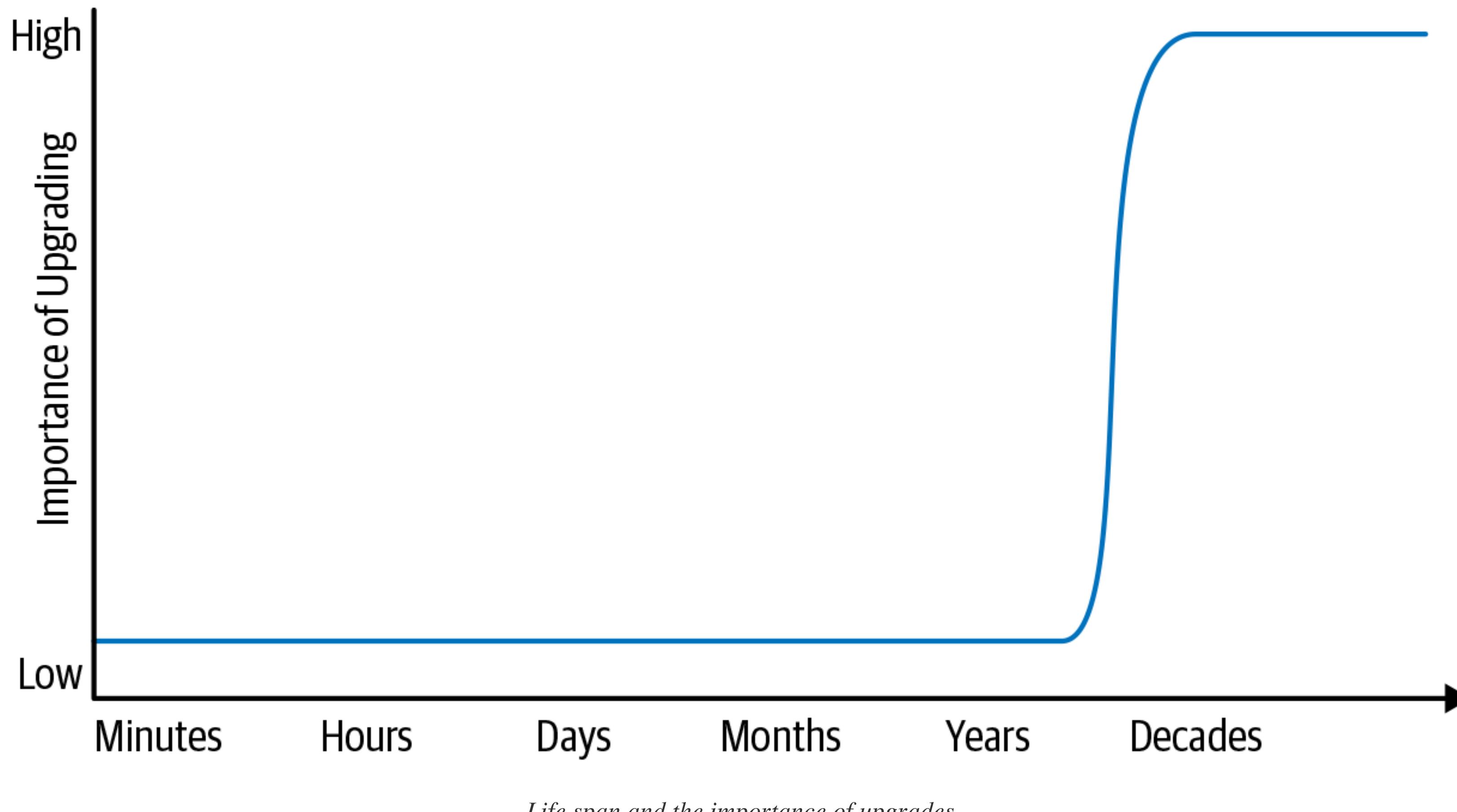
Automated Refactorings in VSC



Refactoring Risks

- Developer time is valuable: is this the best use of time *today*?
- Despite best intentions, may not be safe
- Potential for version control conflicts

Technical Debt

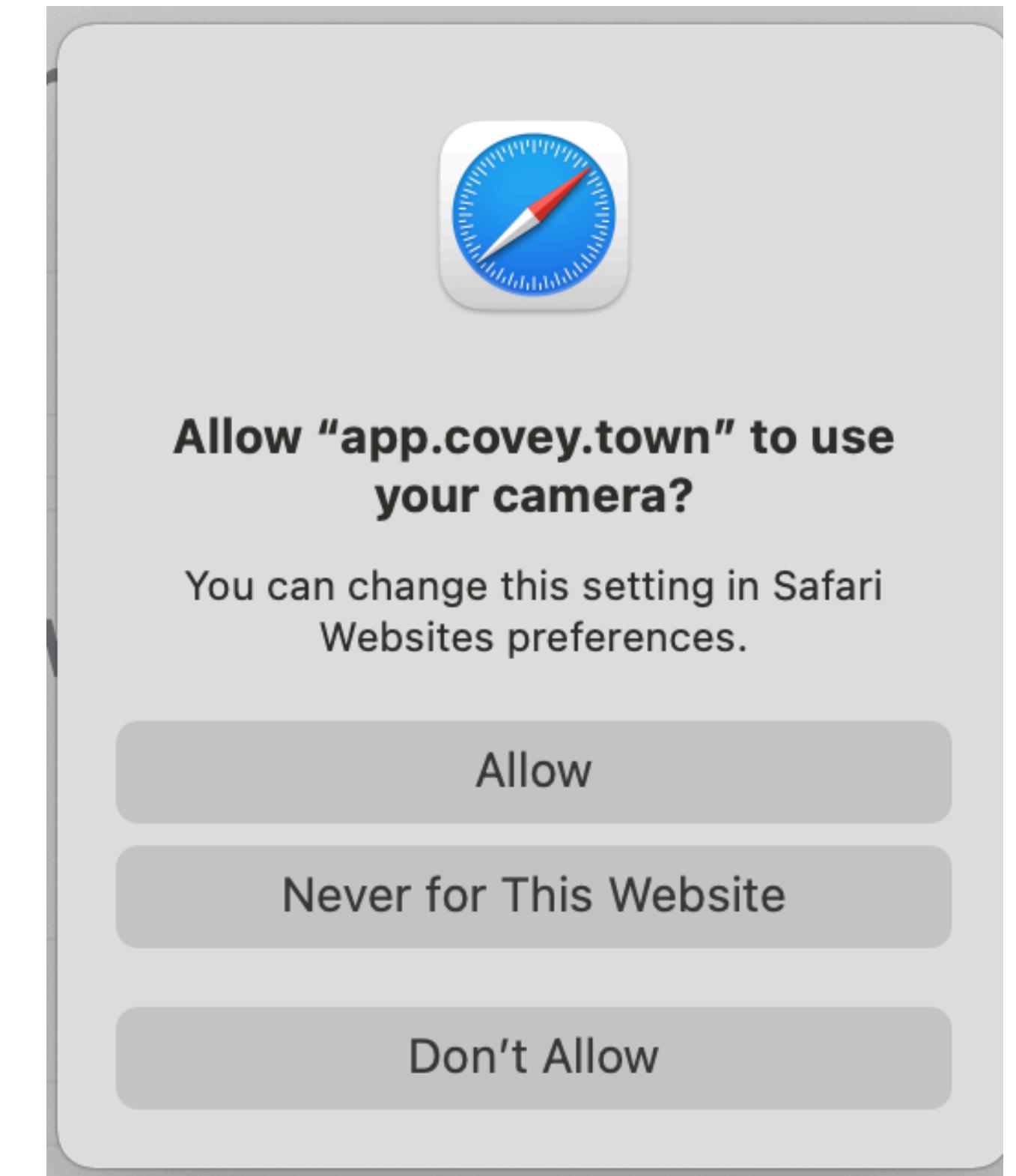


Figures: "Software Engineering at Google: Lessons Learned from Programming Over Time," Wright, Winters and Mansreck, 2020 (O'Reilly)

Web-Based Video Chat Apps

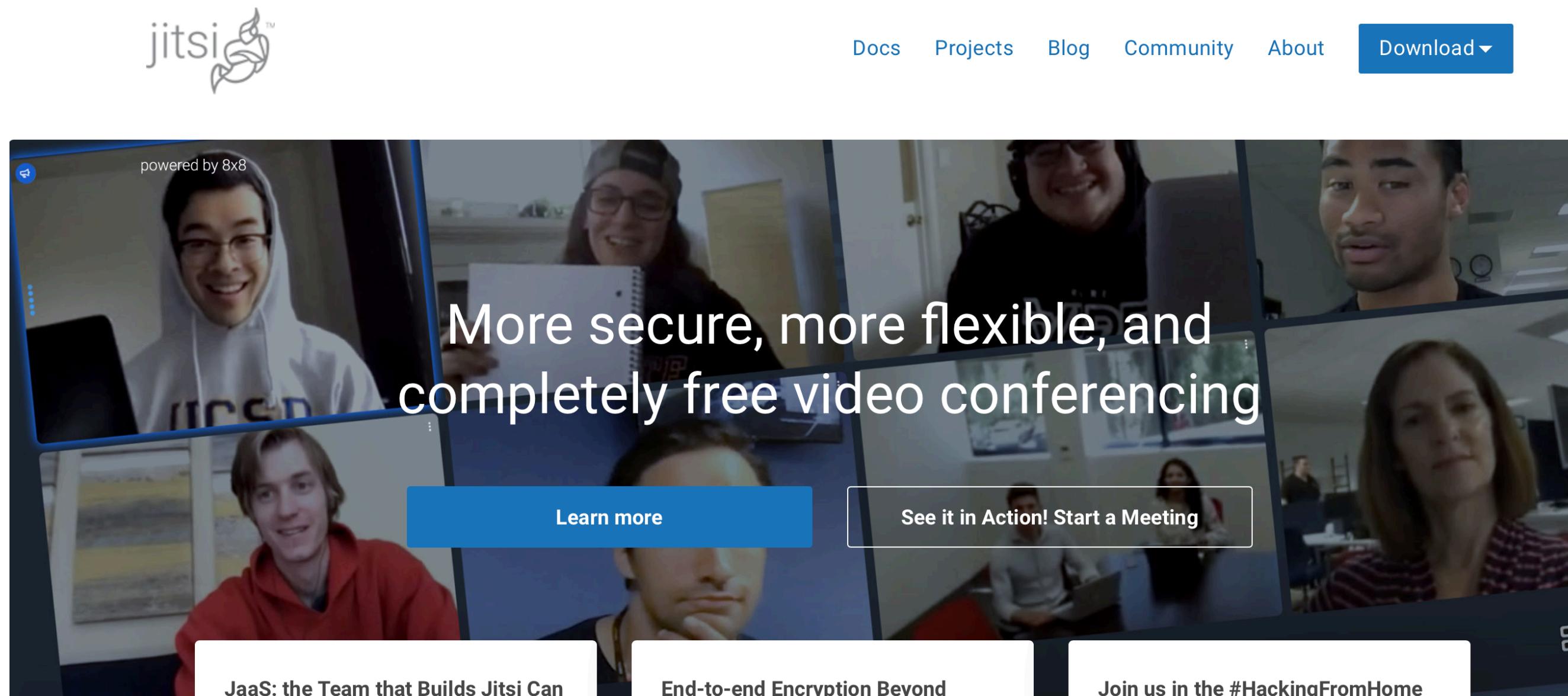
WebRTC, a standard for:

- Capturing camera + microphone with JS
- Transporting real-time audio + video between browsers
- Displaying real-time audio + video with JS
- Everything that does video chat in your browser without a plugin (everything now?) uses WebRTC



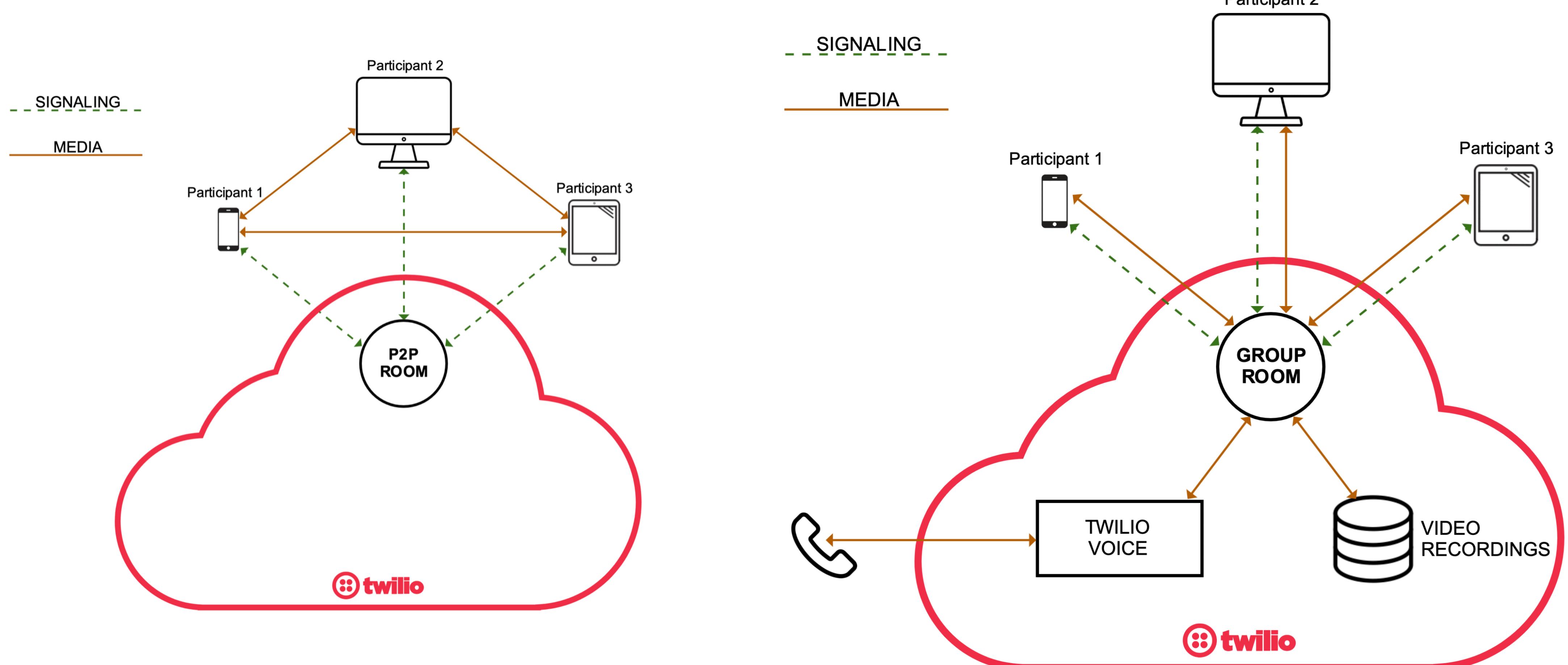
Other WebRTC services you might use

- Vonage - Like Twilio, but support calls with 1000's of participants, livestream integrations - <https://www.vonage.com/>
- Jitsi - Open source infrastructure for WebRTC, support calls with 1000's of participants, rich meeting UI <https://jitsi.org>



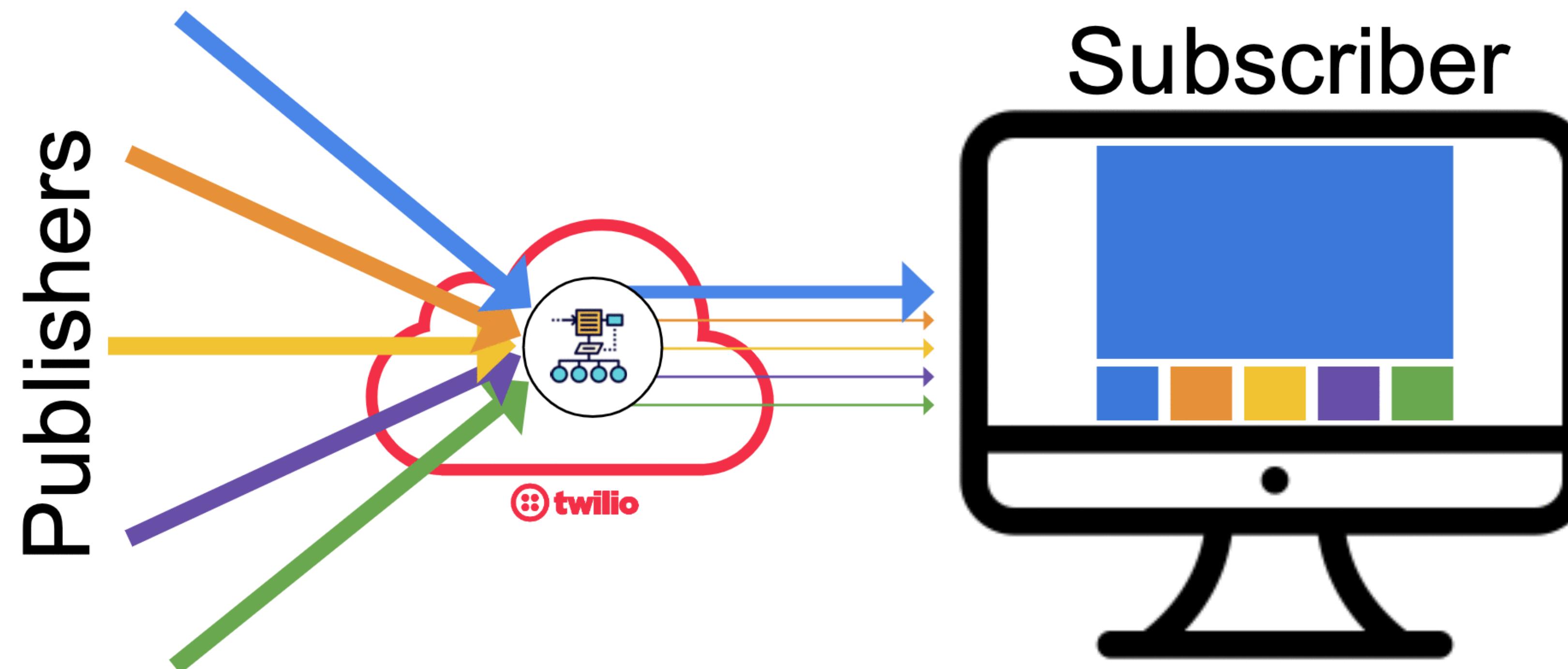
Twilio Programmable Video

Two room “topologies”: P2P + Group

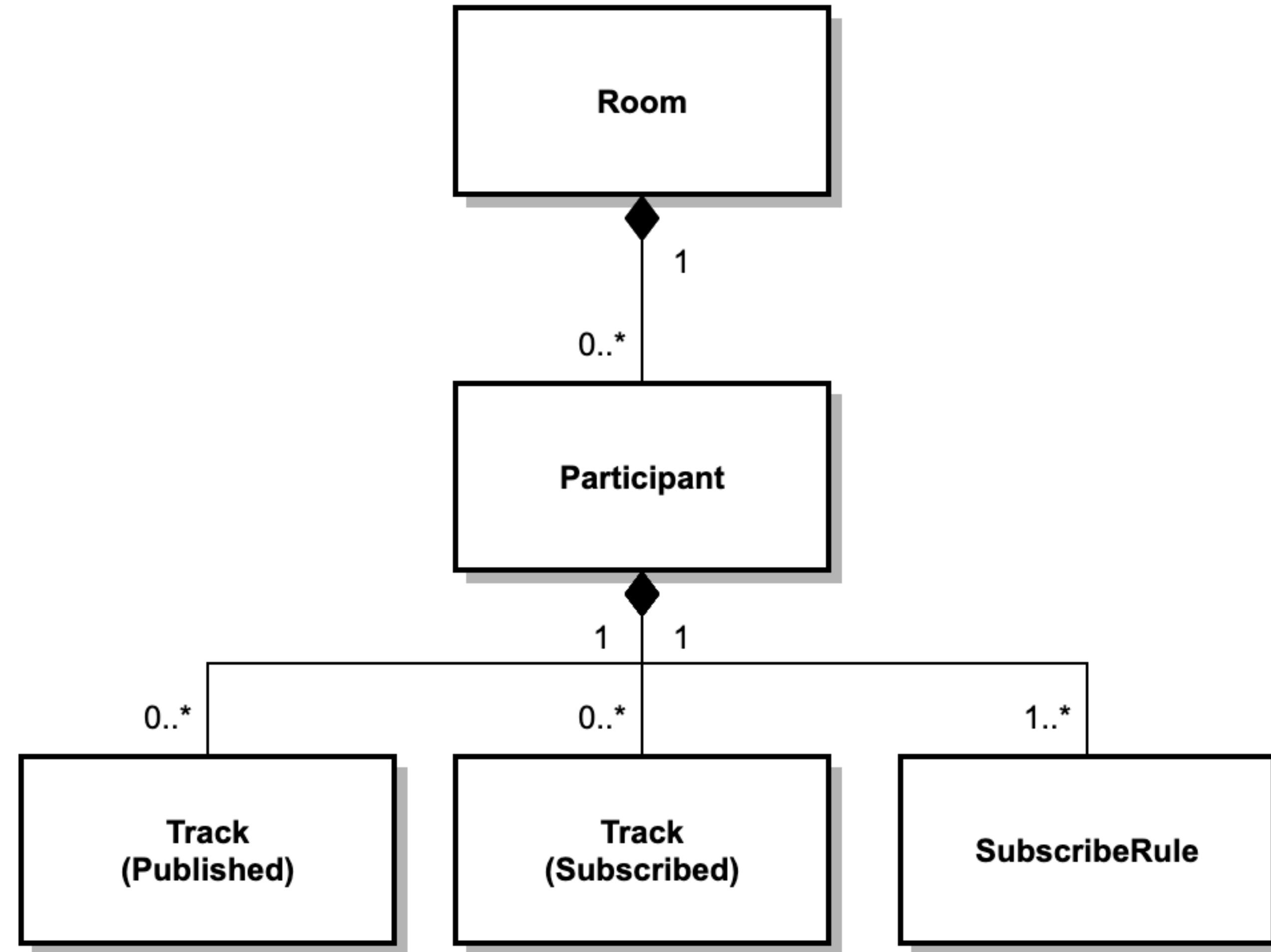


Twilio Programmable Video

Publishers + Subscribers

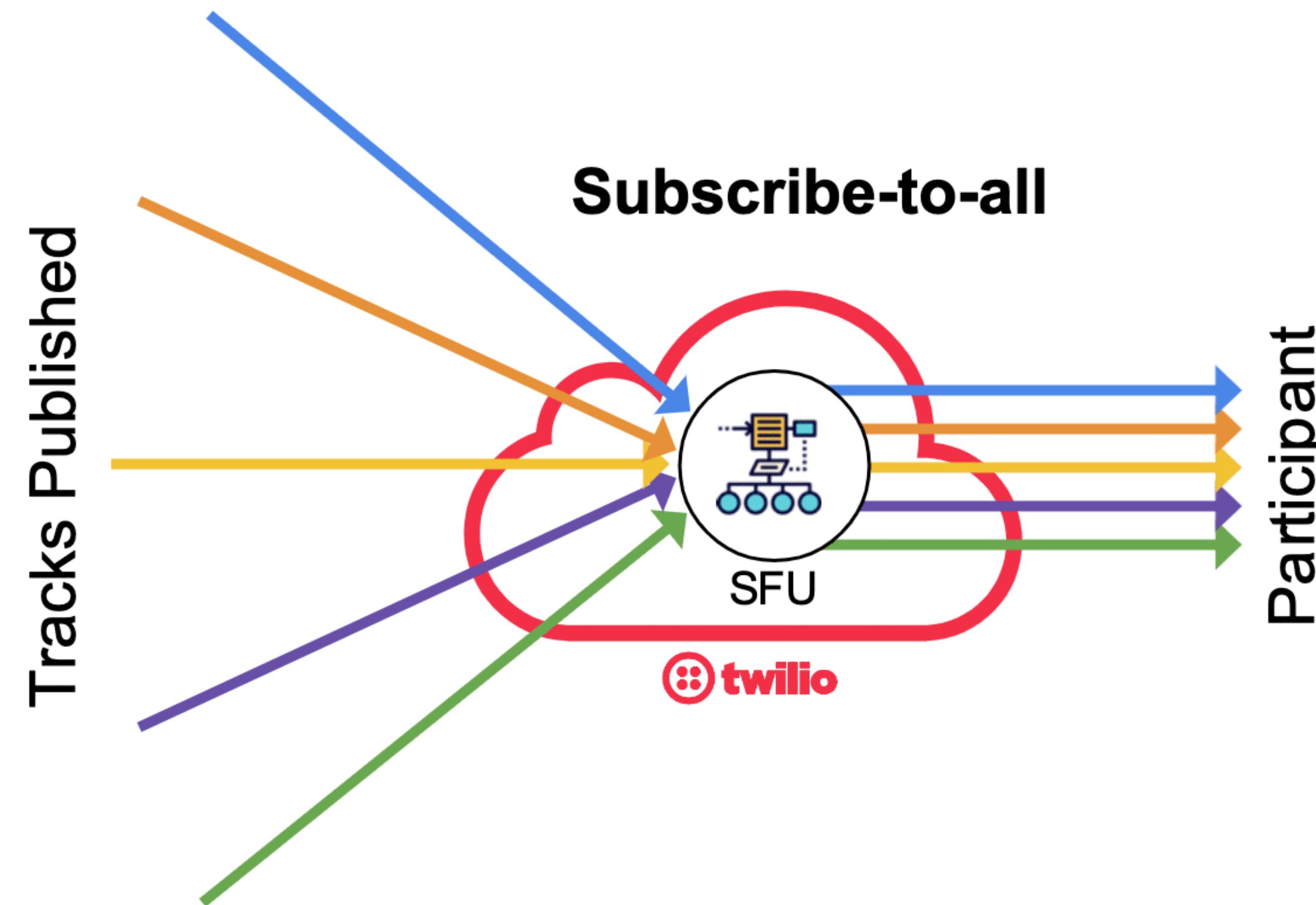


Twilio Abstractions



Twilio Programmable Video

Tracks & Subscriptions



This work is licensed under a Creative Commons Attribution-ShareAlike license

- This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>
- You are free to:
 - Share — copy and redistribute the material in any medium or format
 - Adapt — remix, transform, and build upon the material
 - for any purpose, even commercially.
- Under the following terms:
 - Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
 - No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.