

# Microservice Architectures

17-313 Fall 2022

## Inspirations:

- Martin Fowler (<http://martinfowler.com/articles/microservices.html>)
- Josh Evans @ Netflix (<https://www.youtube.com/watch?v=CZ3wluvmHeM>)
- Matt Ranney @ Uber (<https://www.youtube.com/watch?v=kb-m2fasdDY>)
- Christopher Meiklejohn & Filibuster (<http://filibuster.cloud>)

# Administrivia

- Homework 3B due Thursday (Oct 6).
- Recitation this week: midterm review (**come prepared!**)
  - Work through problems on the previous midterms – many students found this helpful.
  - Any questions on the previous midterm questions – bring them to recitation to discuss as a class.
- Midterm on October 11<sup>th</sup> (in class, regular timing).

# Learning Goals

- Contrast the monolithic application design with a modular design based on microservices.
- Reason about how architectural choices affect software quality and process attributes.
- Reason about tradeoffs of microservices architectures.

# **Before we get to microservices...**

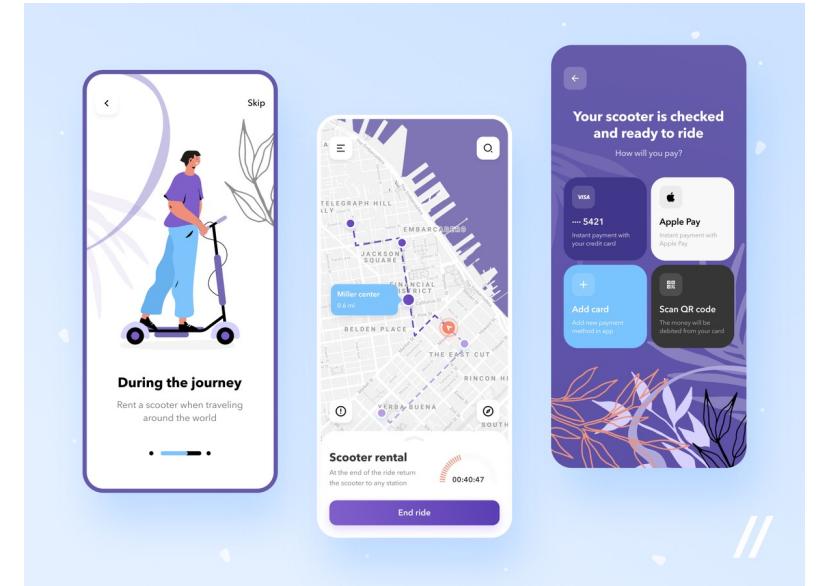
# How might these apps be architected?



The Netflix homepage features a large banner for the movie "THE HAUNTING OF BLY MANOR". Below the banner, there are buttons for "Play Episode", "More info", and "Watch together". A section titled "Popular on Netflix" shows thumbnails for "MURDOCH MYSTERIES", "VIKINGS", "RuPaul's Drag Race", "New Girl", and "Modern Family". Another section titled "Trending Now" shows thumbnails for "HAUNTING OF HILL HOUSE", "modernfamily", "LA RÉVOLUTION", "the office", "LAST KINGDOM", and "COMM".

**Sinomix Docs** interface:

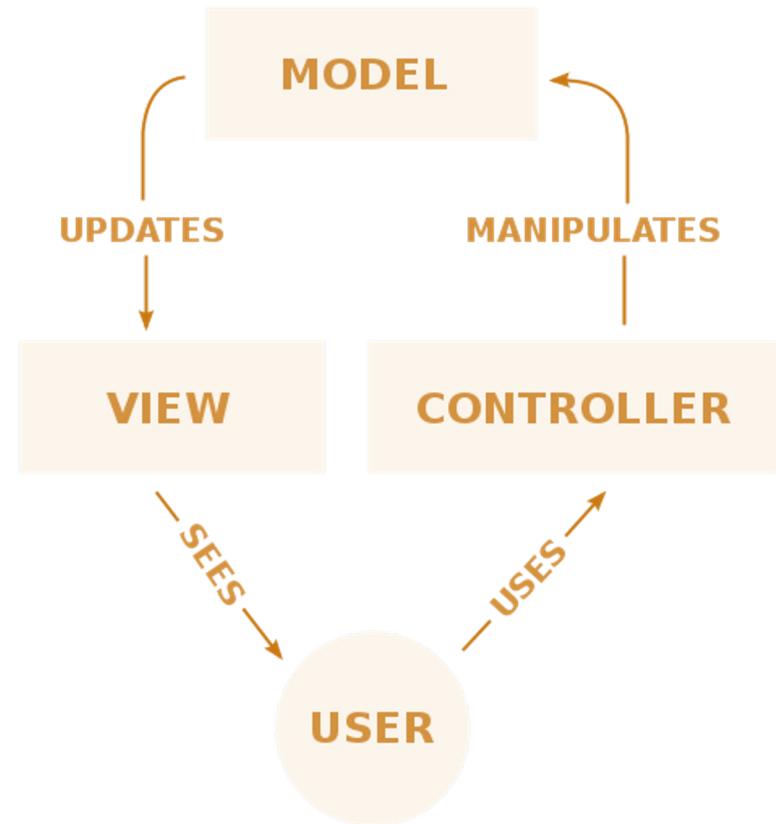
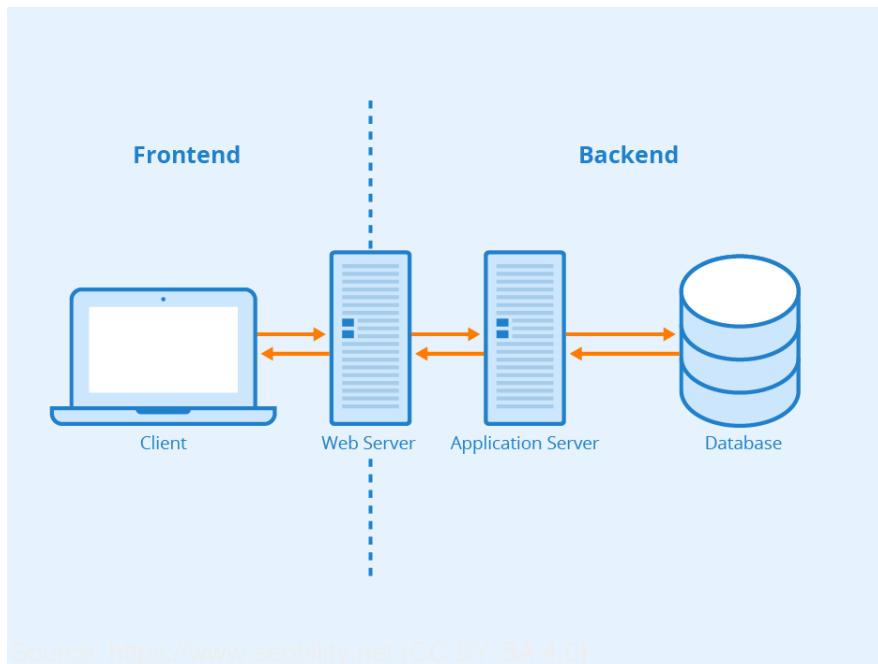
- Top navigation: Home, Docs, Events, Tags, Users & Groups.
- Document list for "Tourism":
  - Title: "Tourism" (by admin) - Creation Date: 11/18/17
  - Contributors: admin
  - Content: "No comments on this document yet" (with "Add a comment" button).
  - Files: 29799.pdf, 2993\_ho\_00\_p\_2048x1136.jpg, Lyon-immobilien-neuf-01.jpg, Lyon-Riverfront.jpg.
  - Actions: Export, Delete, Edit.
- Left sidebar: Tags, Search, Filter, Sort, Create Document, Add a document.
- Bottom navigation: Previous, Next, 10 per page, 299 documents found.



The Facebook news feed shows various posts:

- Andy Chung:** Update Status, Add Photos/Video.
- Vivian Wang:** "Colorful day on the boardwalk!" (with a photo of a colorful Ferris wheel).
- Alex Ristevski and 1 other:** 2 events this week.
- TRENDING:** Paul Rudd: Lip Sync Battle with Paul Rudd, Ben & Jerry's Ben and Jerry's Announces Four New 'Core' Flavors, Pepe de Lucía: Renowned Spanish flamenco guitarist Pepe de Lucía dies.
- PEOPLE YOU MAY KNOW:** Greg Marrs, Mike Rumble, Julie Zhuo.

# Monolithic styles: Client-server or MVC



# Monoliths make trade-offs on software quality

Several consequences of this architecture on:

- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Evolution
- Testability
- Ownership

The screenshot shows a search results page for the term "Tourism". At the top, there's a search bar with the word "Tourism" and a "Search" button. Below the search bar, the results are listed in a table format. The columns are "TITLE" and "CREATION DATE". The results include:

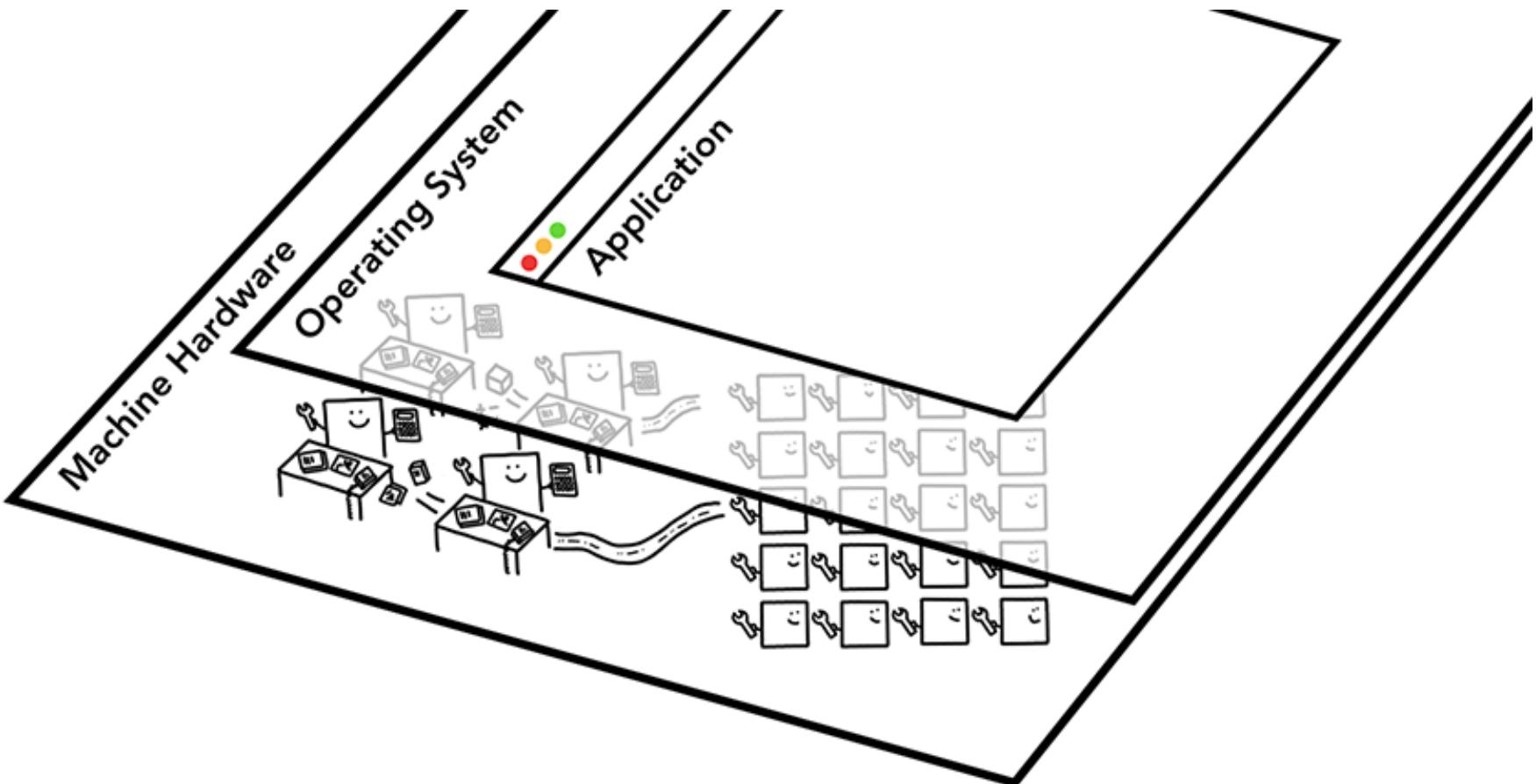
- Authoritarianism (1)
- Auto-reification (1)
- Discrimination based on skin color (1)
- Collection (1)
- Indigo Era (economist) (1)
- Dhali syndrome (1)
- Korean ethnic nationalism (1)
- Individualism (1)
- Anaphorism (1)
- Muse (1)

At the bottom of the page, there are navigation links for "Previous" (page 1), "Next" (page 2), and "10 per page". It also shows "71MB (0.7%) used on 10.000MB" and "299 documents found".

The top half of the image shows a Facebook news feed. On the left, there's a sidebar with "Andy Chung" and "Edit Profile". The main feed shows posts from "Vivian Wang" (Just now), "Matt Viscomi" (5 mins), and "Andy Chung" (1 min). The post from Vivian Wang includes a photo of a colorful Ferris wheel. The bottom half of the image shows a document viewer interface for a file named "Tourism". The interface has tabs for "Content", "Workflow", "Permissions", and "Activity". It shows a preview of the document, a list of contributors, and a section for comments. There are four thumbnail images of buildings or landscapes.

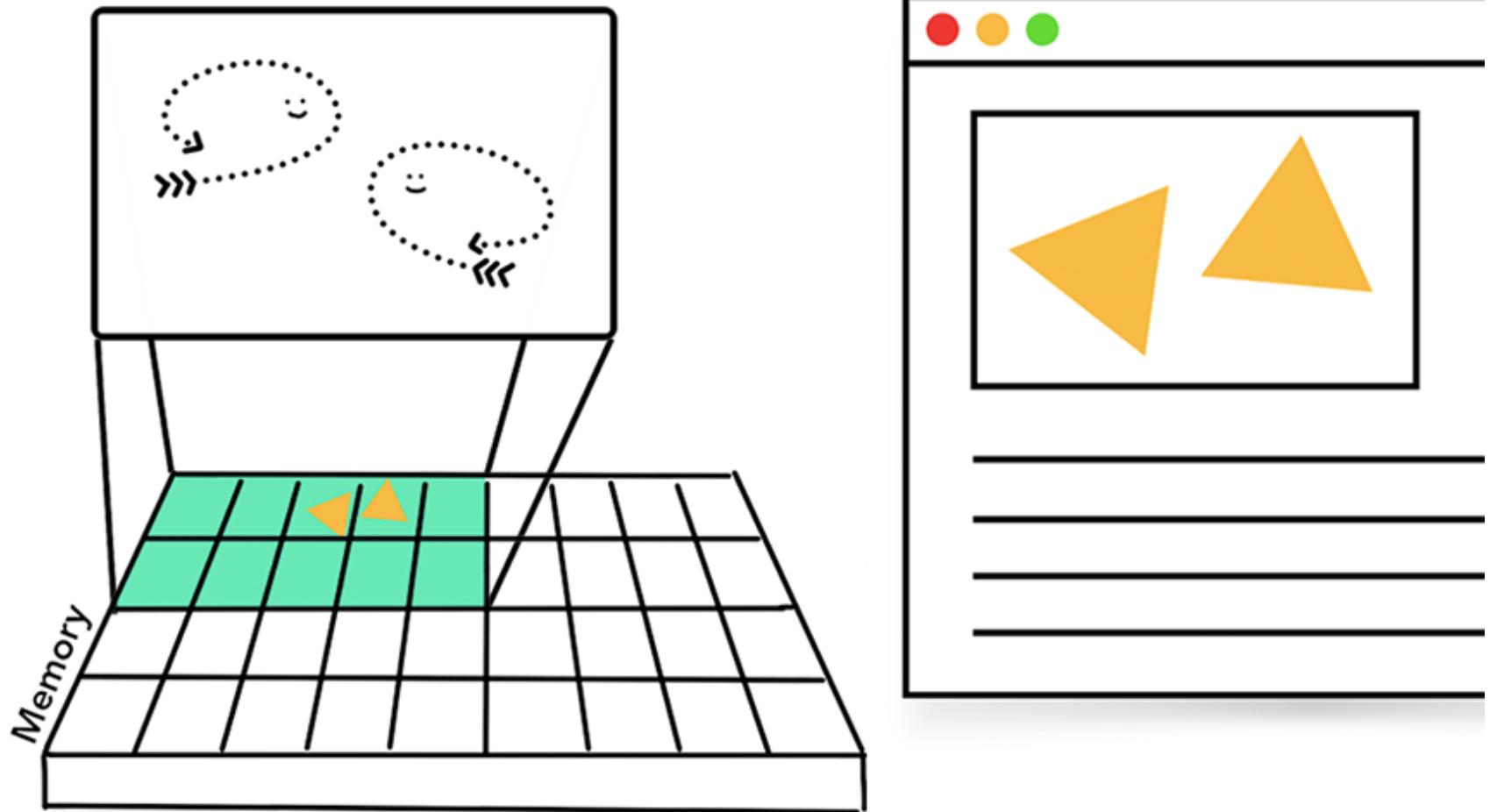
# **Service-based architecture – Chrome**

# Web Browsers



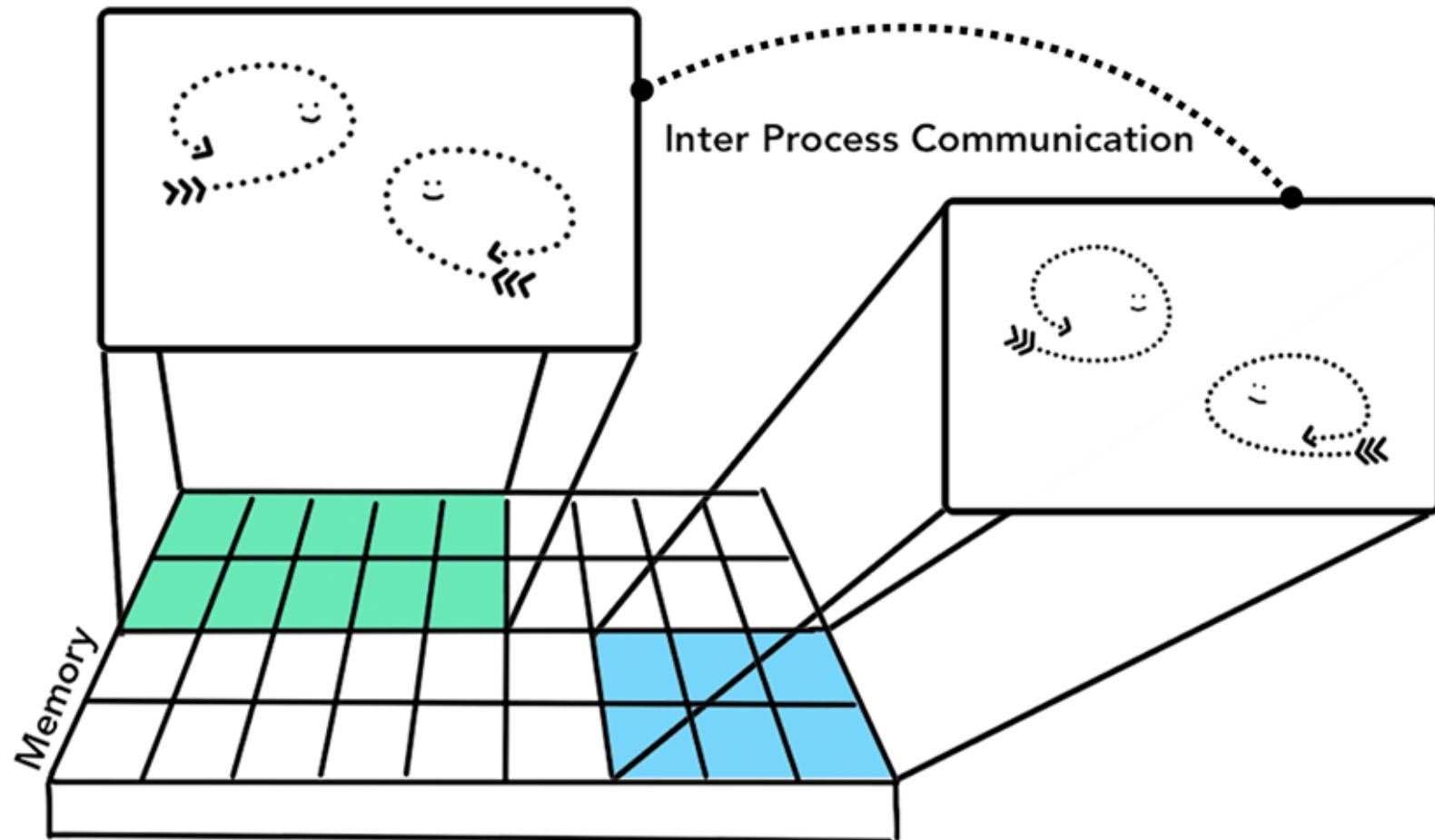
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

# Browser: A multi-threaded process



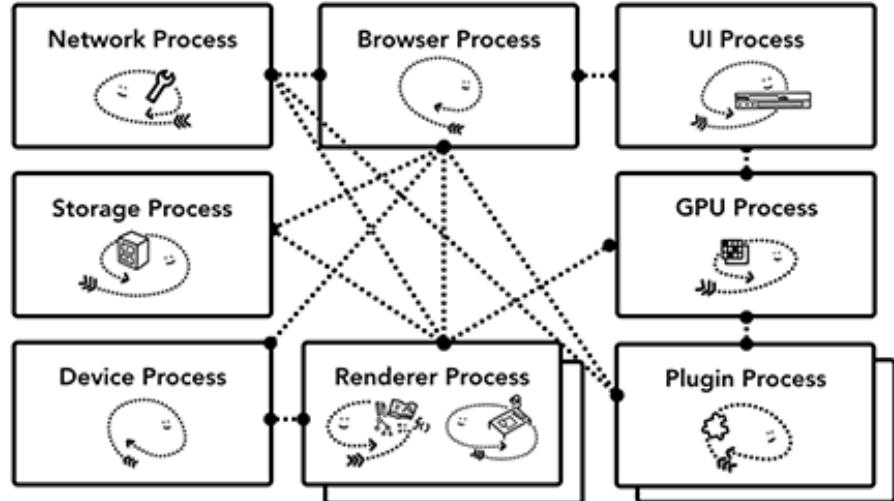
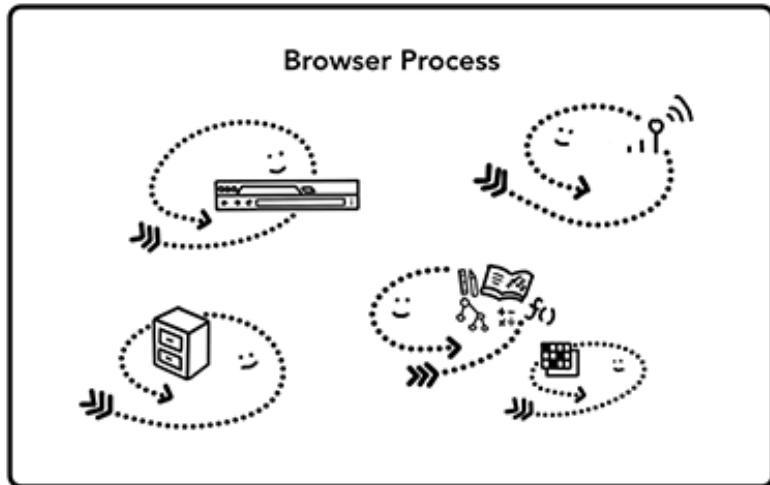
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

# Multi-process browser with IPC



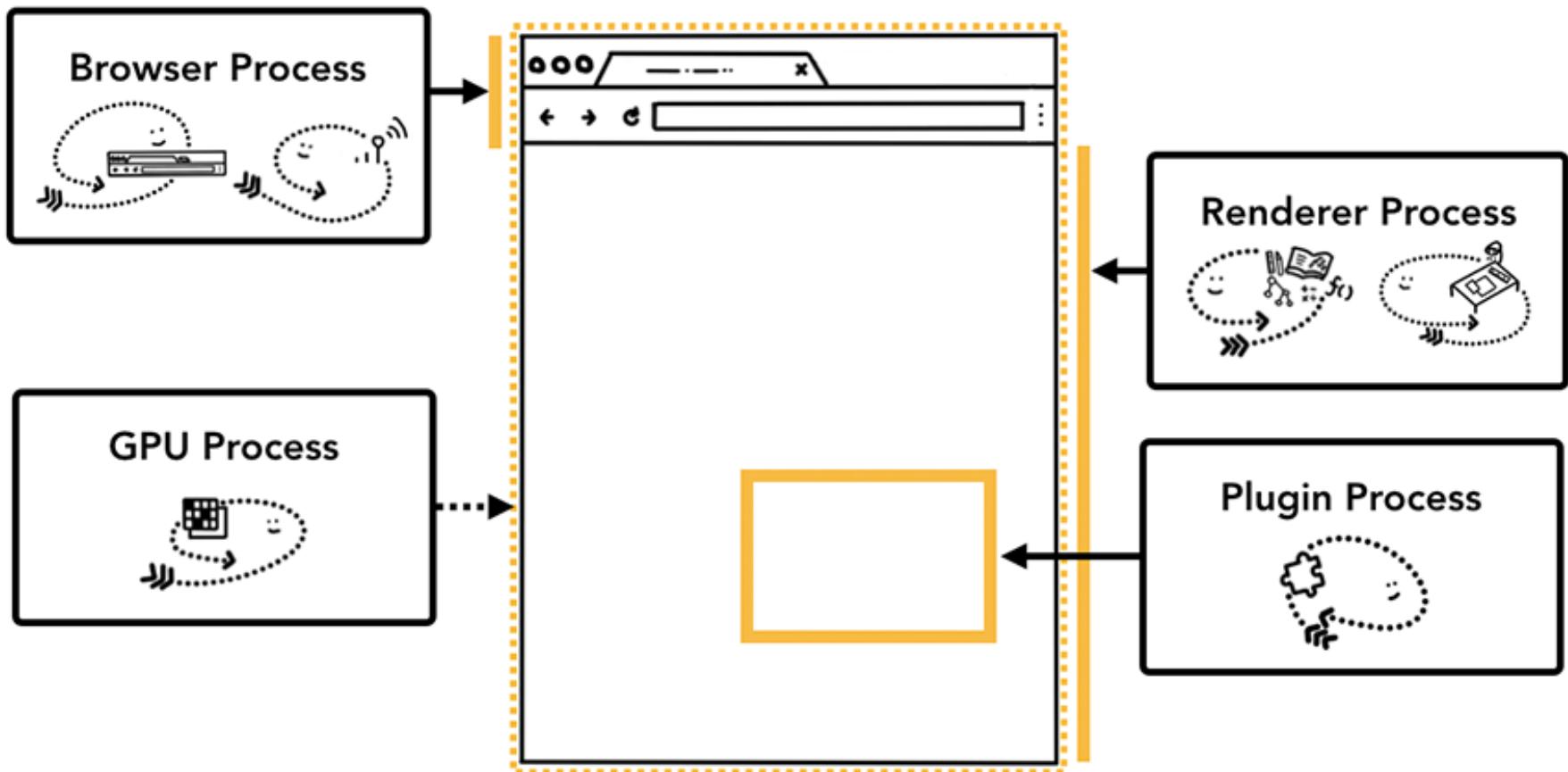
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

# Browser Architectures



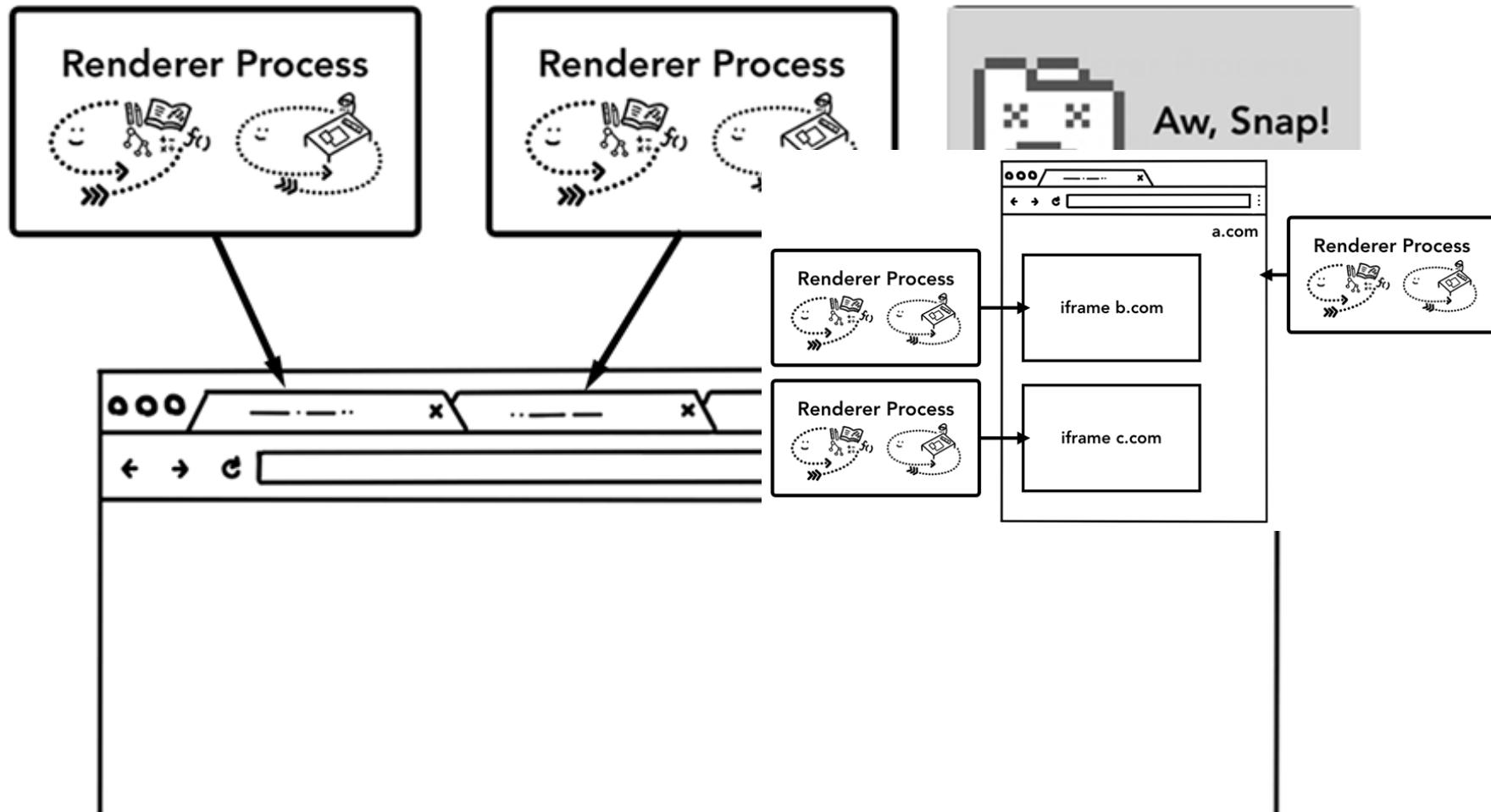
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

# Service-based browser architecture



Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

# Service-based browser architecture



Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

# Navigating to a web site uses service requests



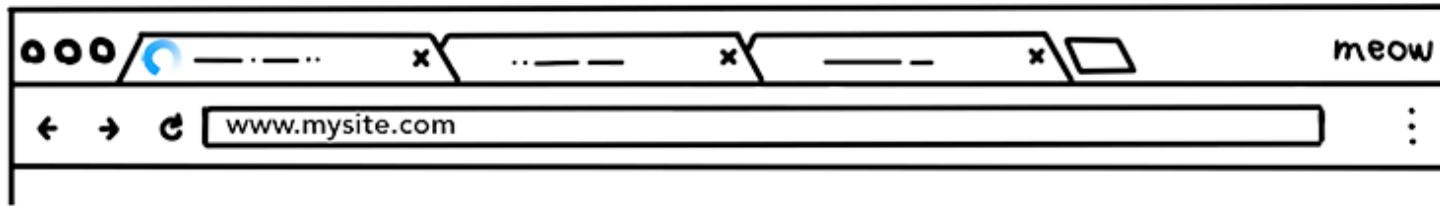
## Browser Process



Is this a search query or URL?

It's a URL!

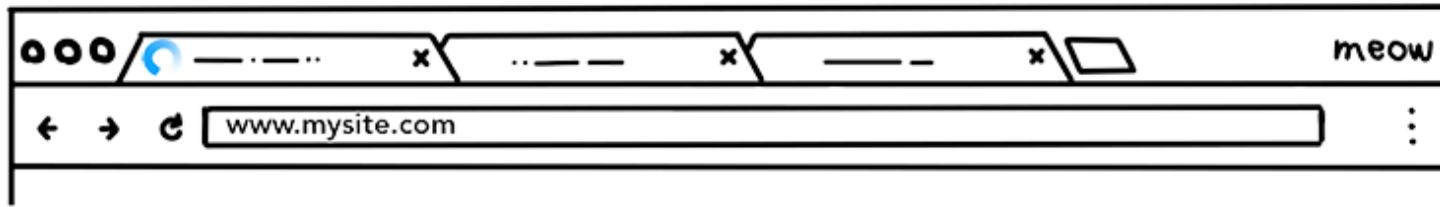
# Navigating to a web site uses service requests



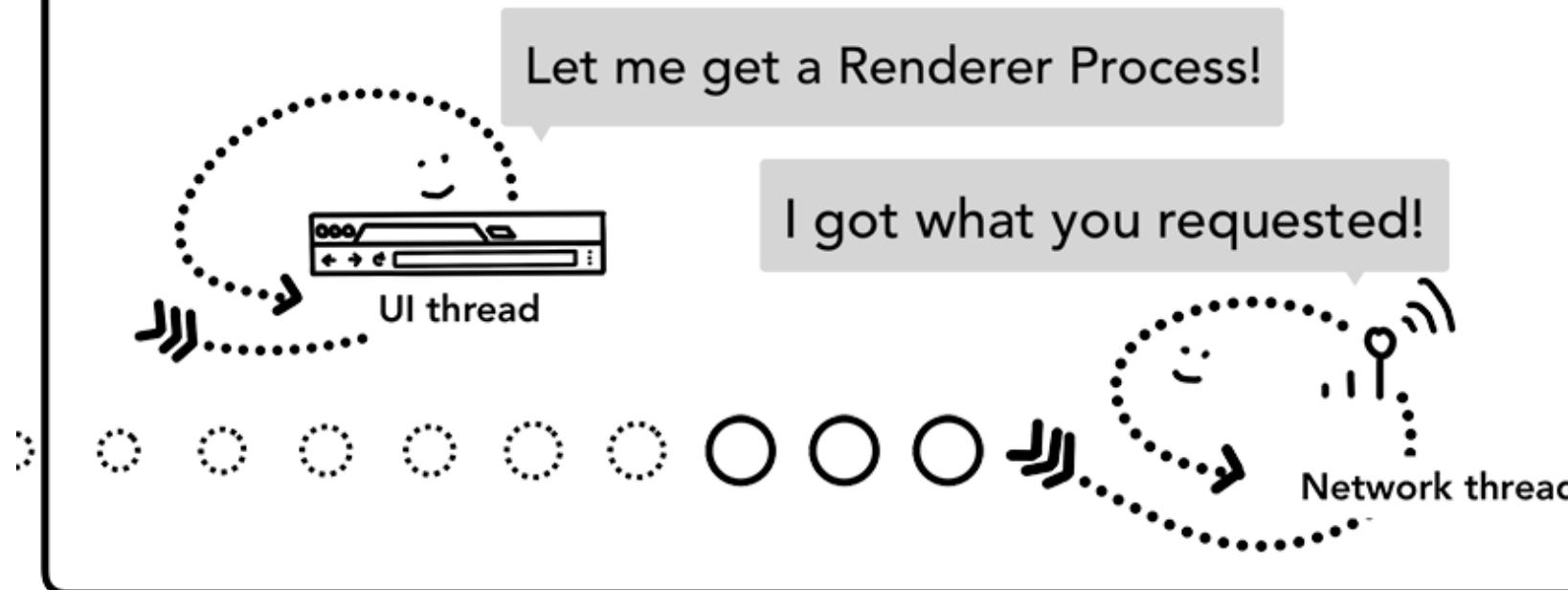
## Browser Process



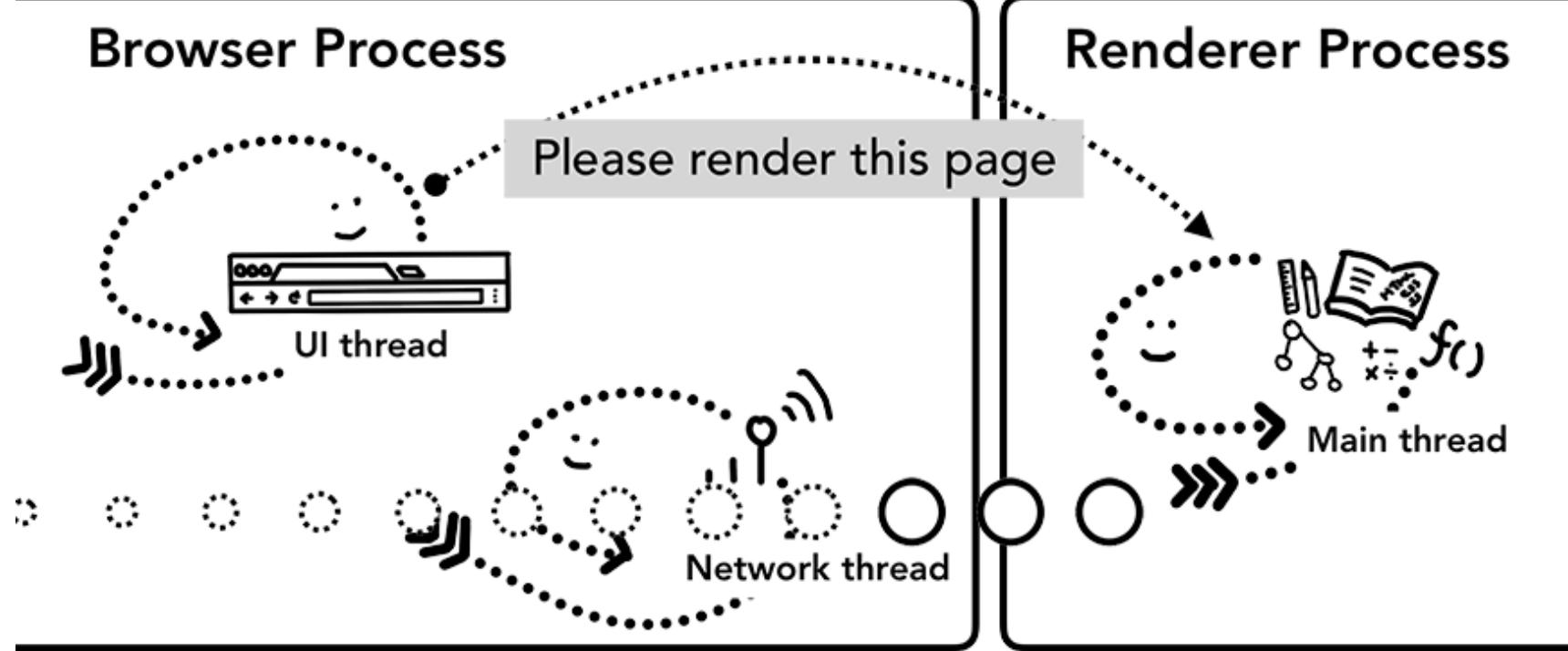
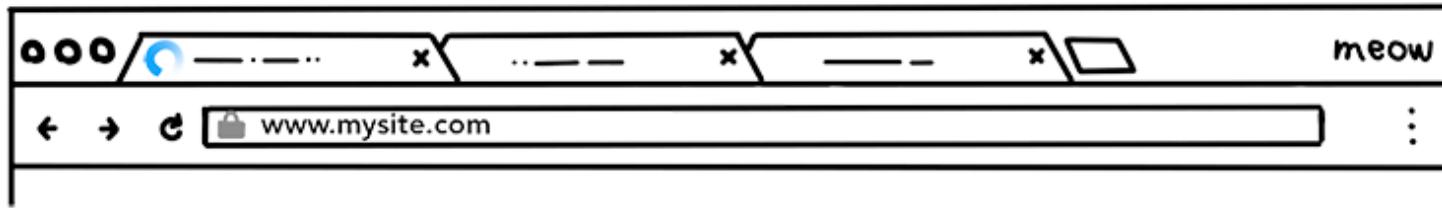
# Navigating to a web site uses service requests



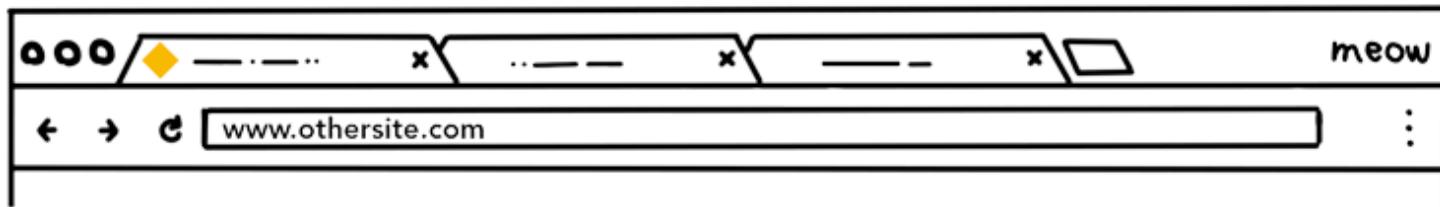
## Browser Process



# Navigating to a web site uses service requests



# Navigating to a web site uses service requests



## Browser Process

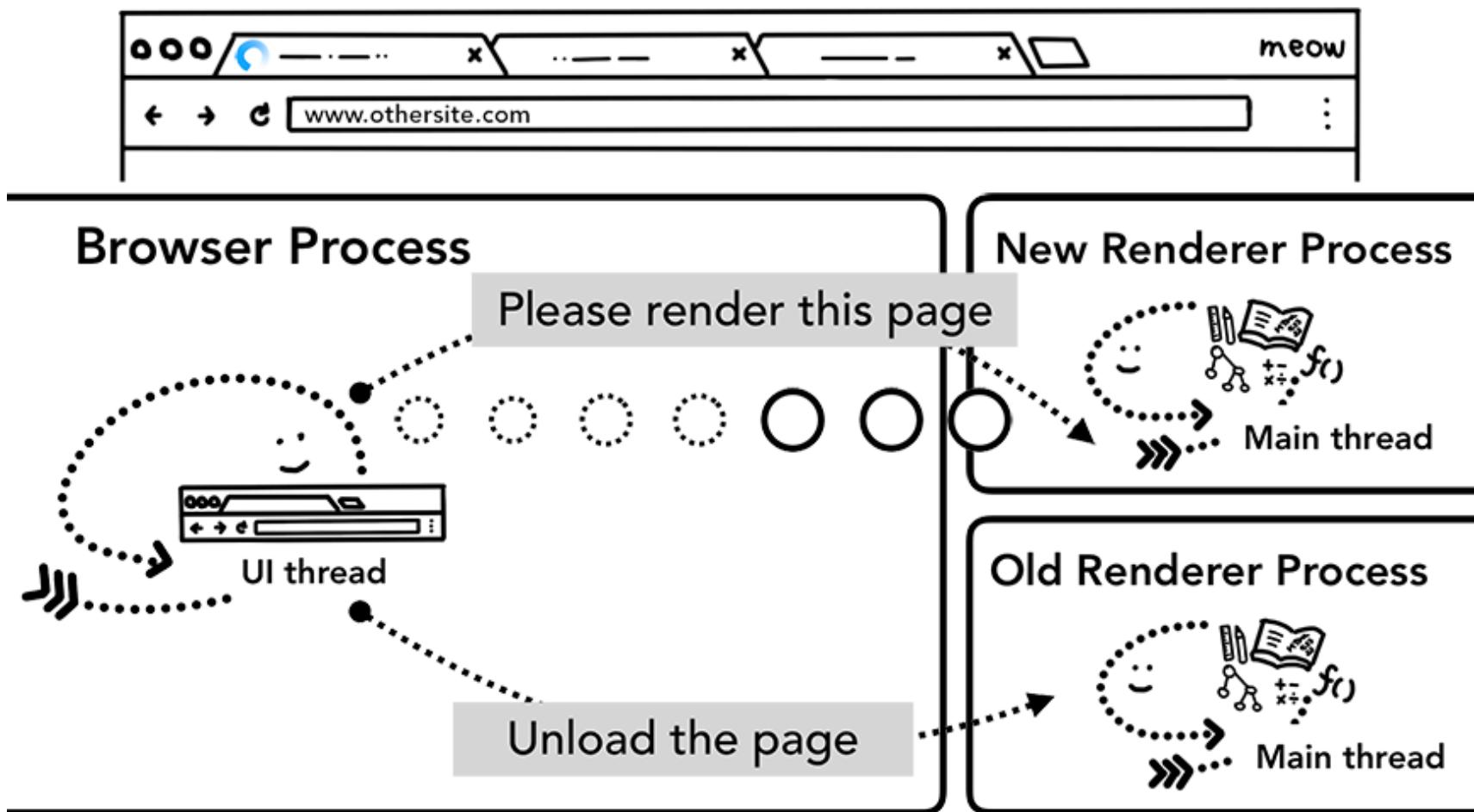


## Renderer Process



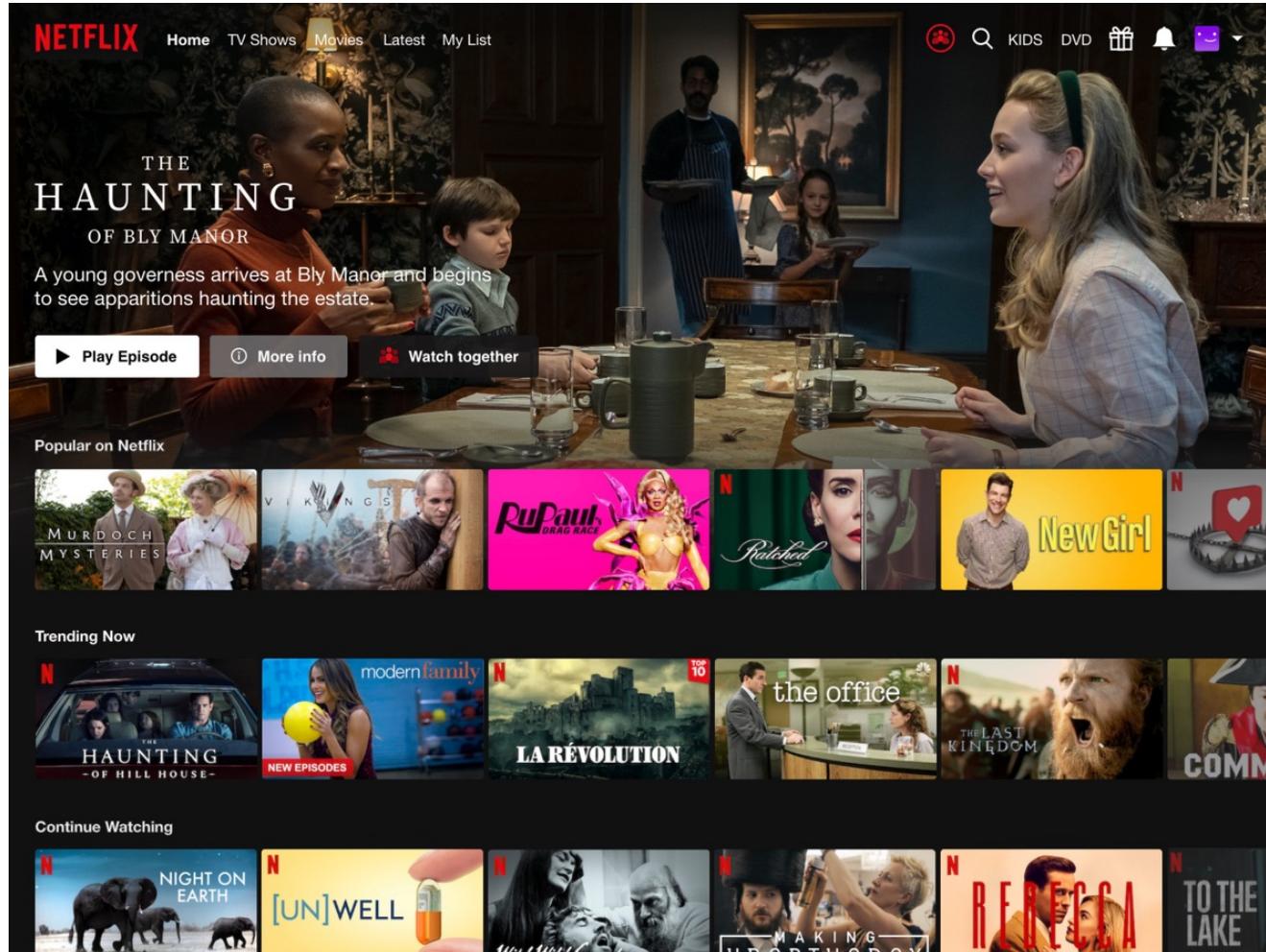
Hey, I'm about to navigate away.  
Do you need to handle `beforeunload`?

# Navigating to a web site uses service requests

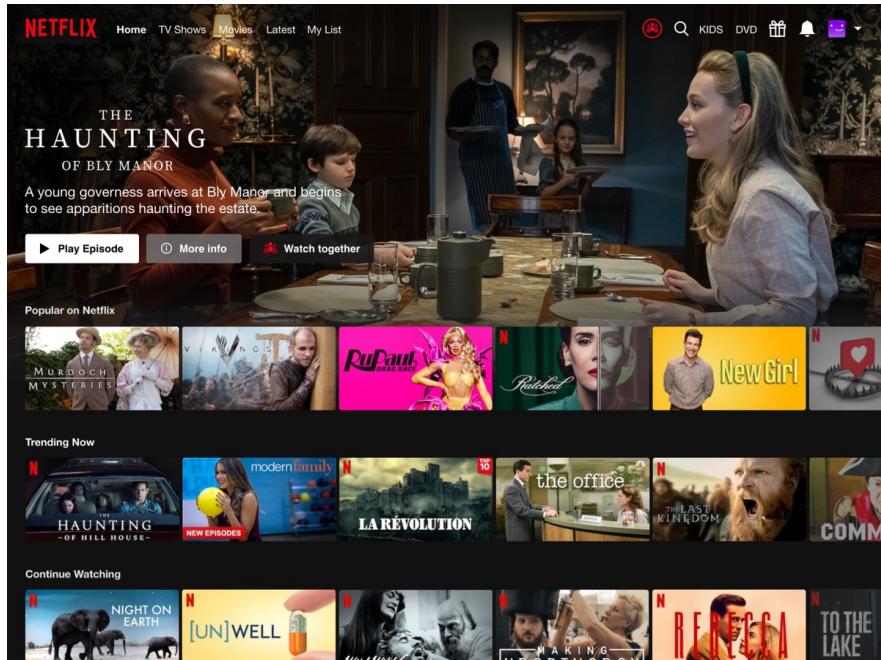


# **Microservice architecture – Netflix**

# Netflix



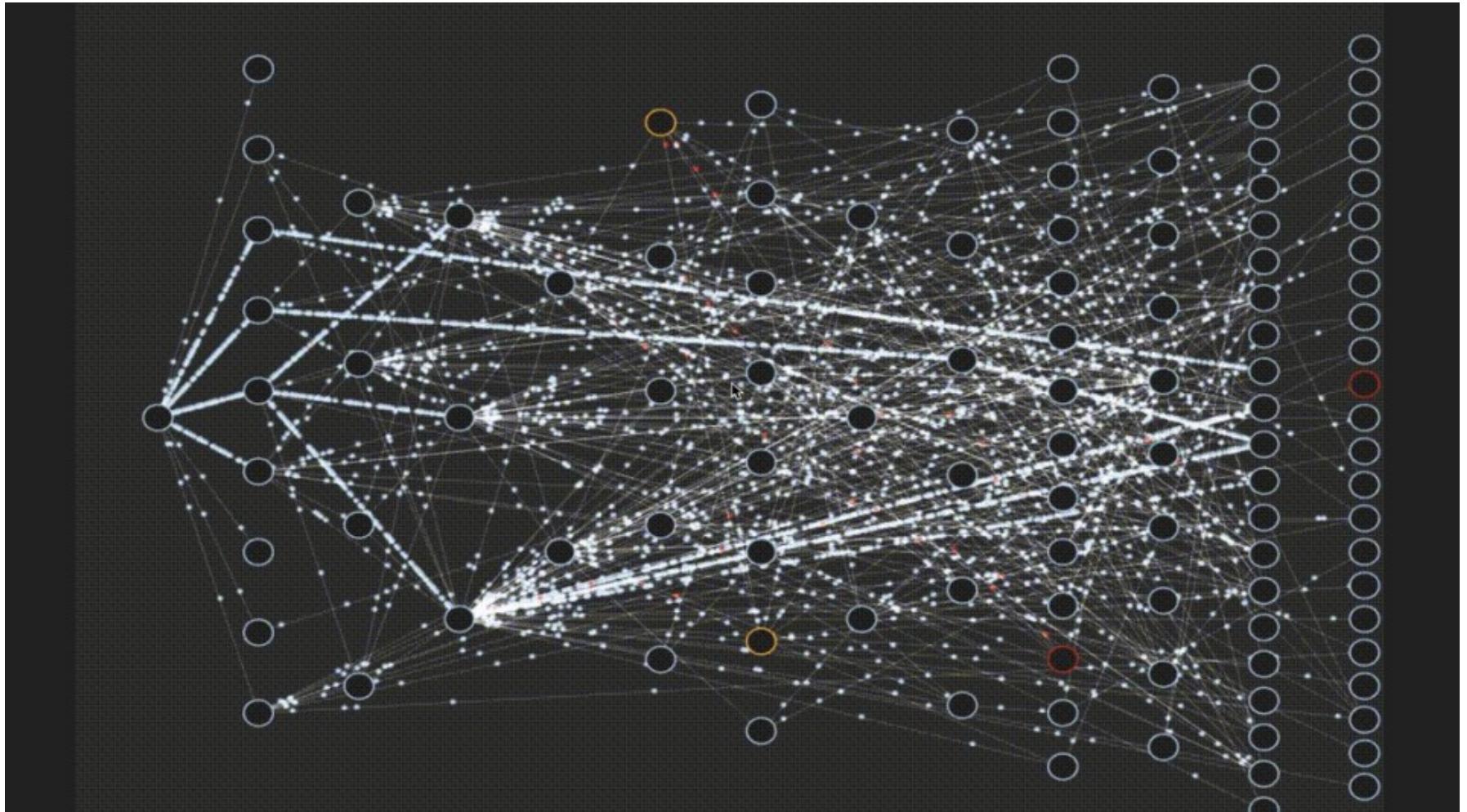
# Netflix Microservices – App Boot



- Recommendations
- Trending Now
- Continue Watching
- My List
- Metrics

(as of 2016)

# Netflix Microservices – One Request



(as of 2016)

<https://www.youtube.com/watch?v=CZ3wluvvmHeM>

# Who uses Microservices?



UBER

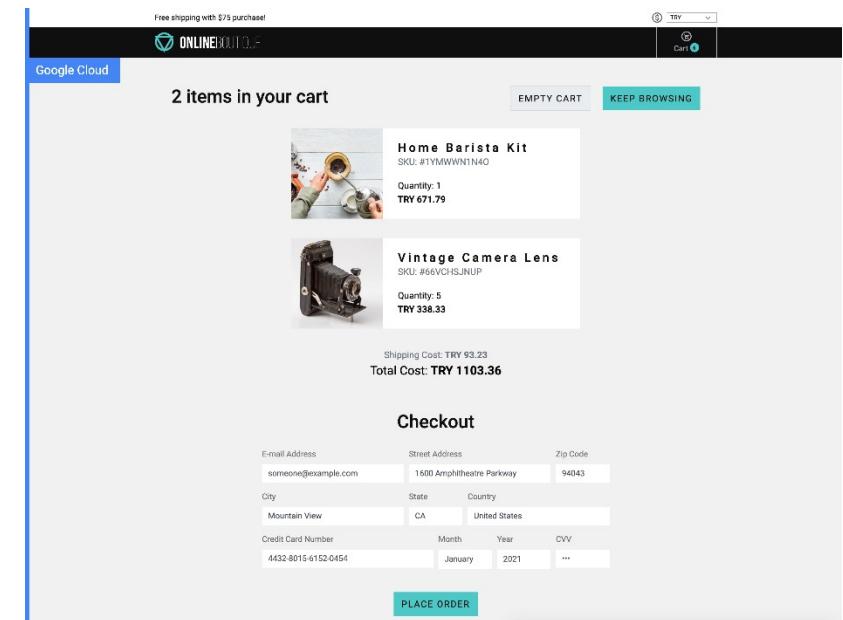
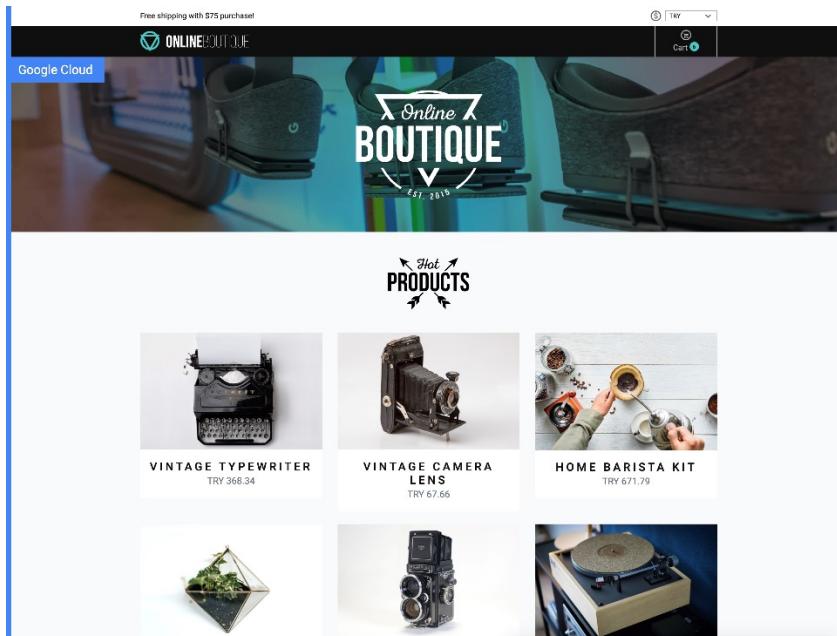


GROUPON®



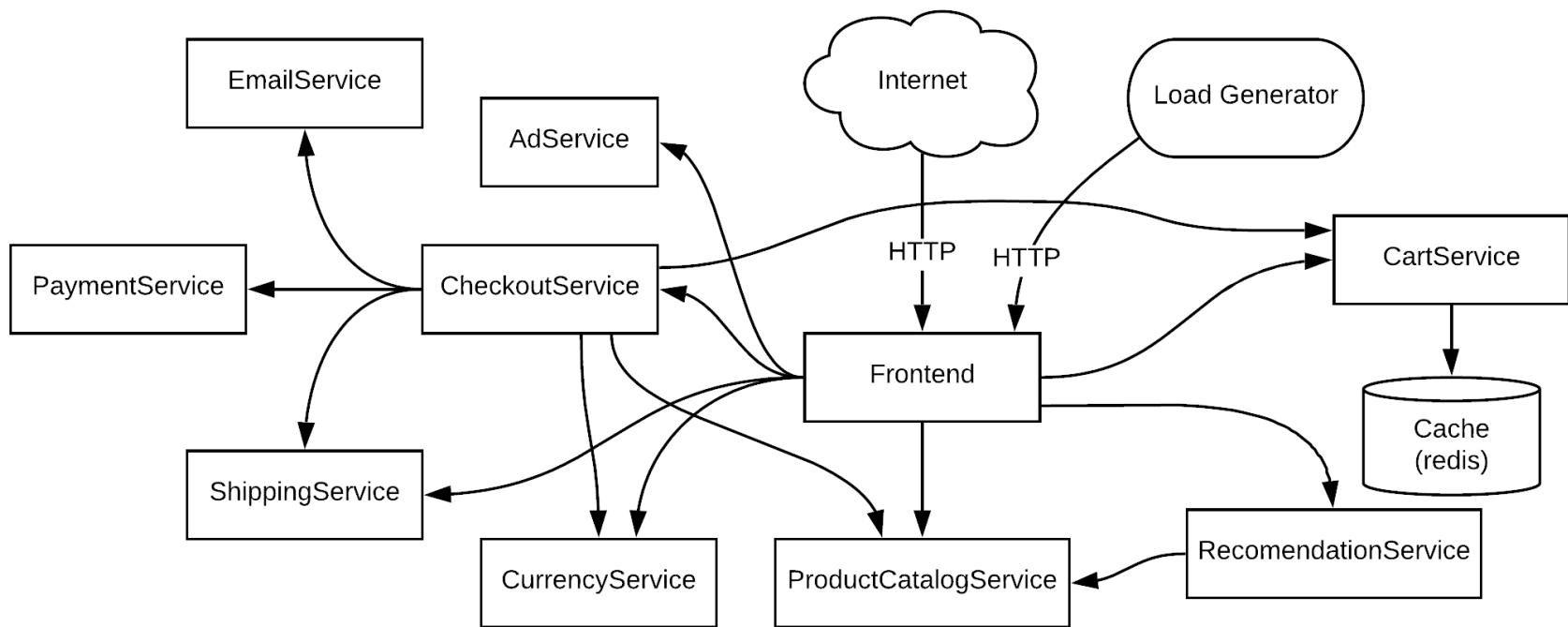
# **Microservices – The Hipster Shop Example**

# Hipster Shop: Guess some microservices



<https://onlineboutique.dev>

# Hipster Shop Microservice Architecture



<https://github.com/GoogleCloudPlatform/microservices-demo>

# Microservices

What are the consequences of this architecture? On:

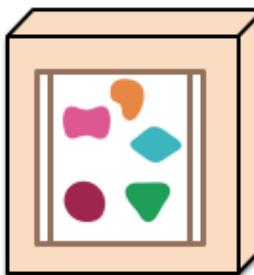
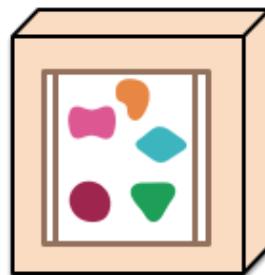
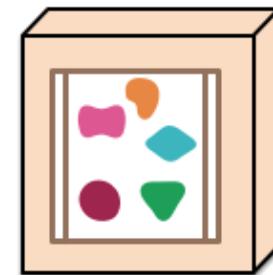
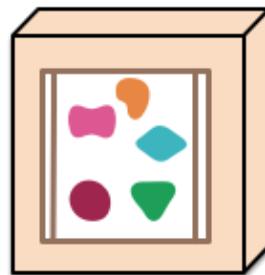
- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Evolution
- Testability
- Ownership
- Data Consistency

# Scalability

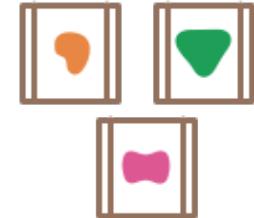
*A monolithic application puts all its functionality into a single process...*



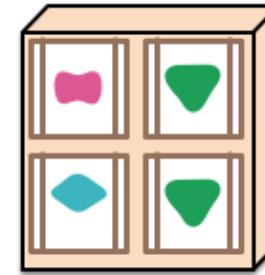
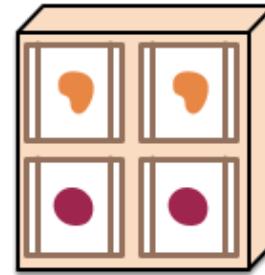
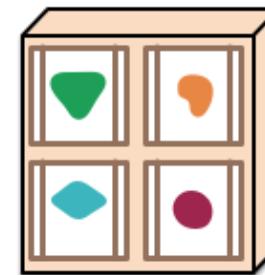
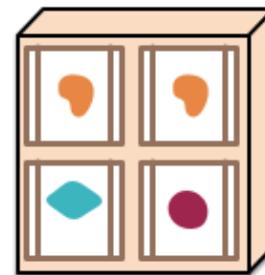
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*

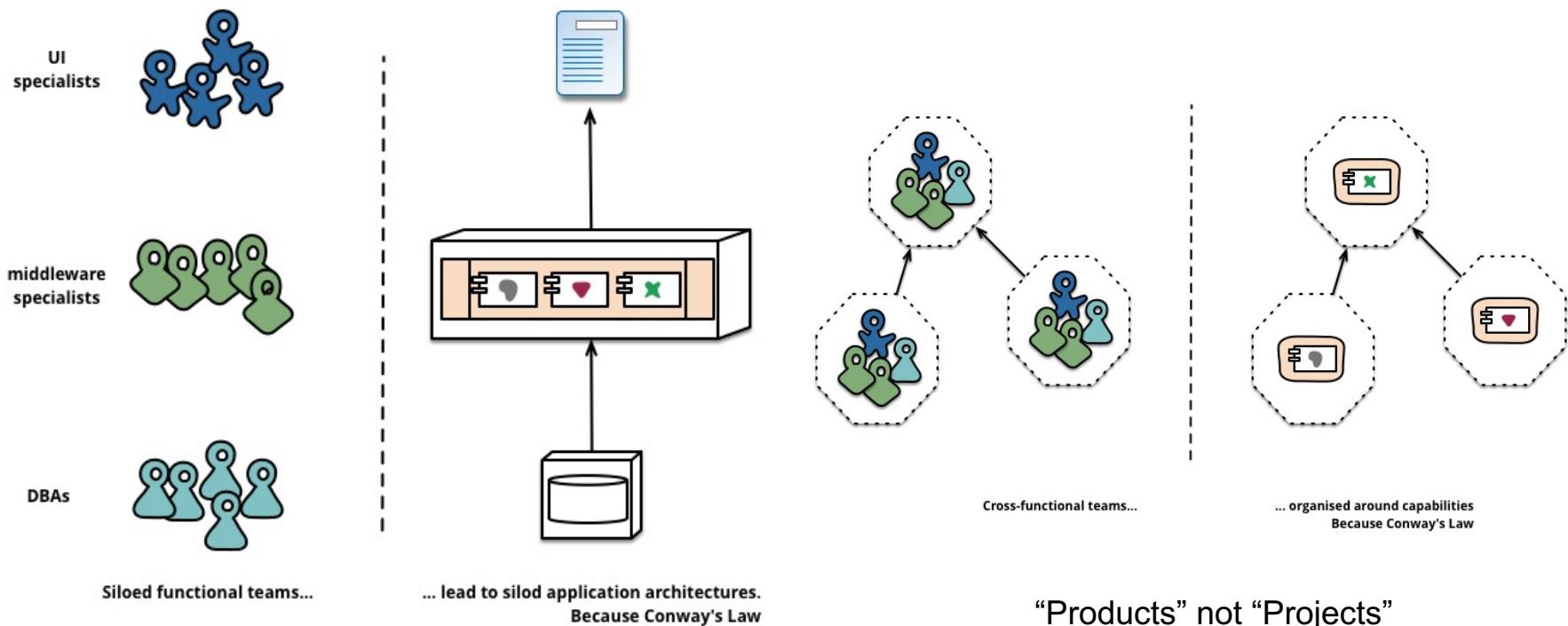


*... and scales by distributing these services across servers, replicating as needed.*



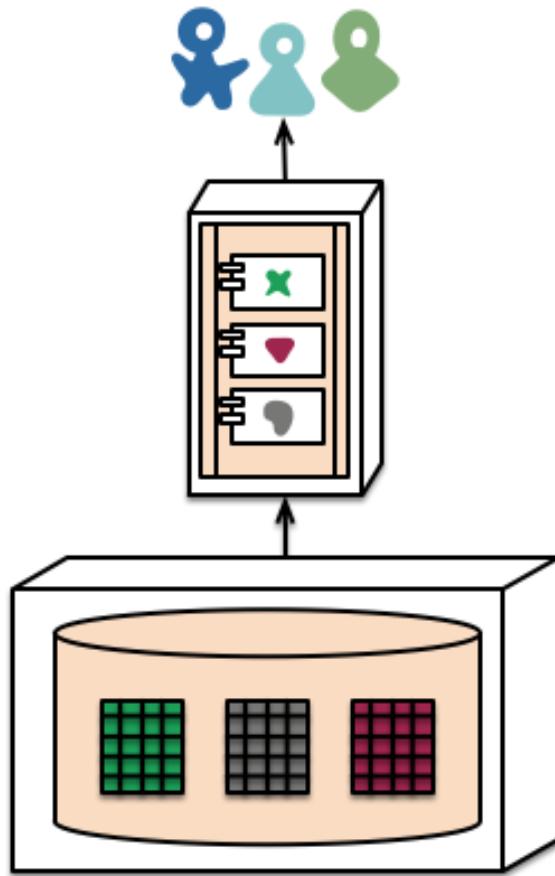
Source: <http://martinfowler.com/articles/microservices.html>

# Team Organization (Conway's Law)

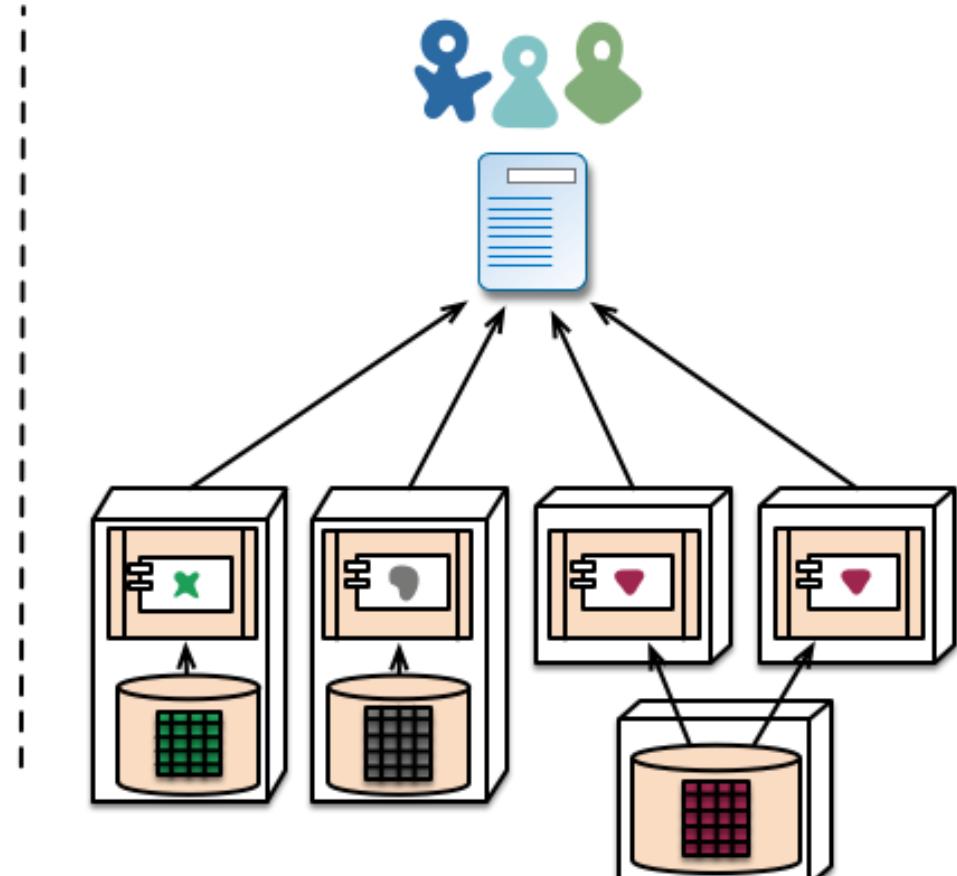


Source: <http://martinfowler.com/articles/microservices.html>

# Data Management and Consistency



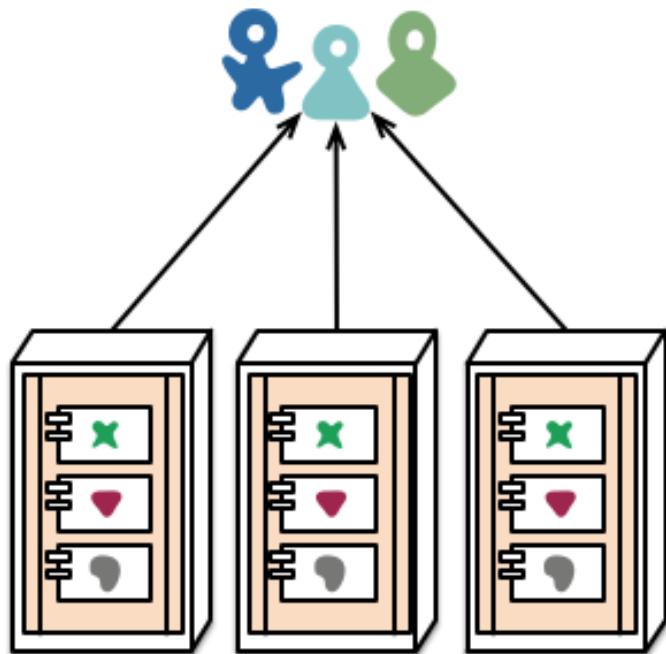
monolith - single database



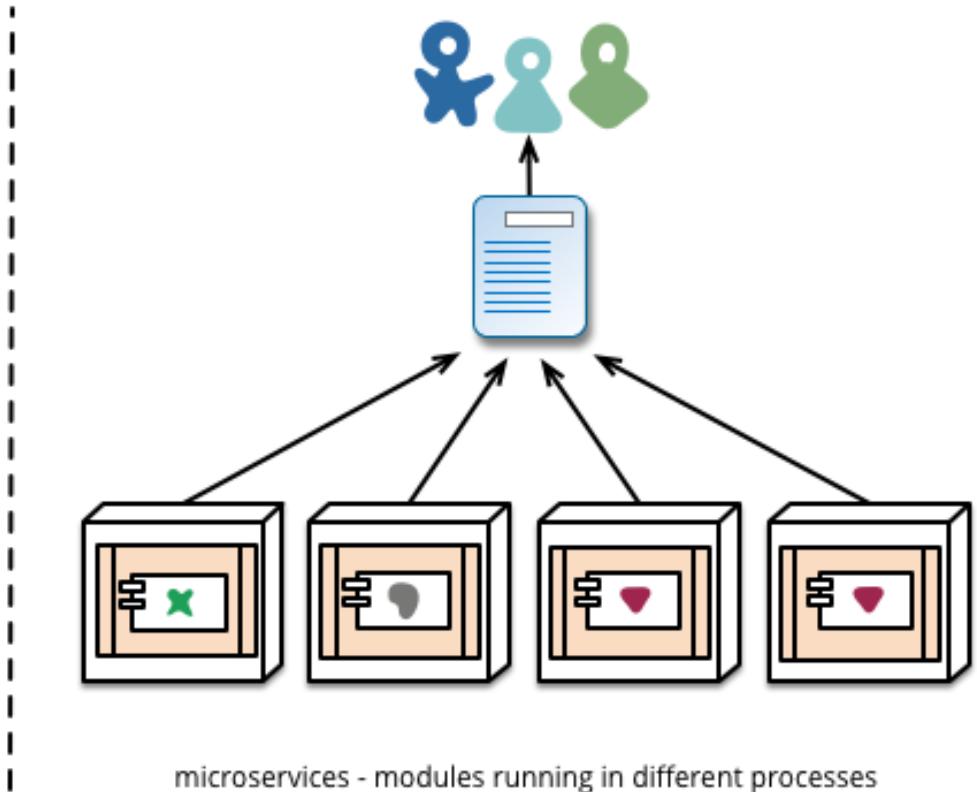
microservices - application databases

Source: <http://martinfowler.com/articles/microservices.html>

# Deployment and Evolution



monolith - multiple modules in the same process



microservices - modules running in different processes

Source: <http://martinfowler.com/articles/microservices.html>

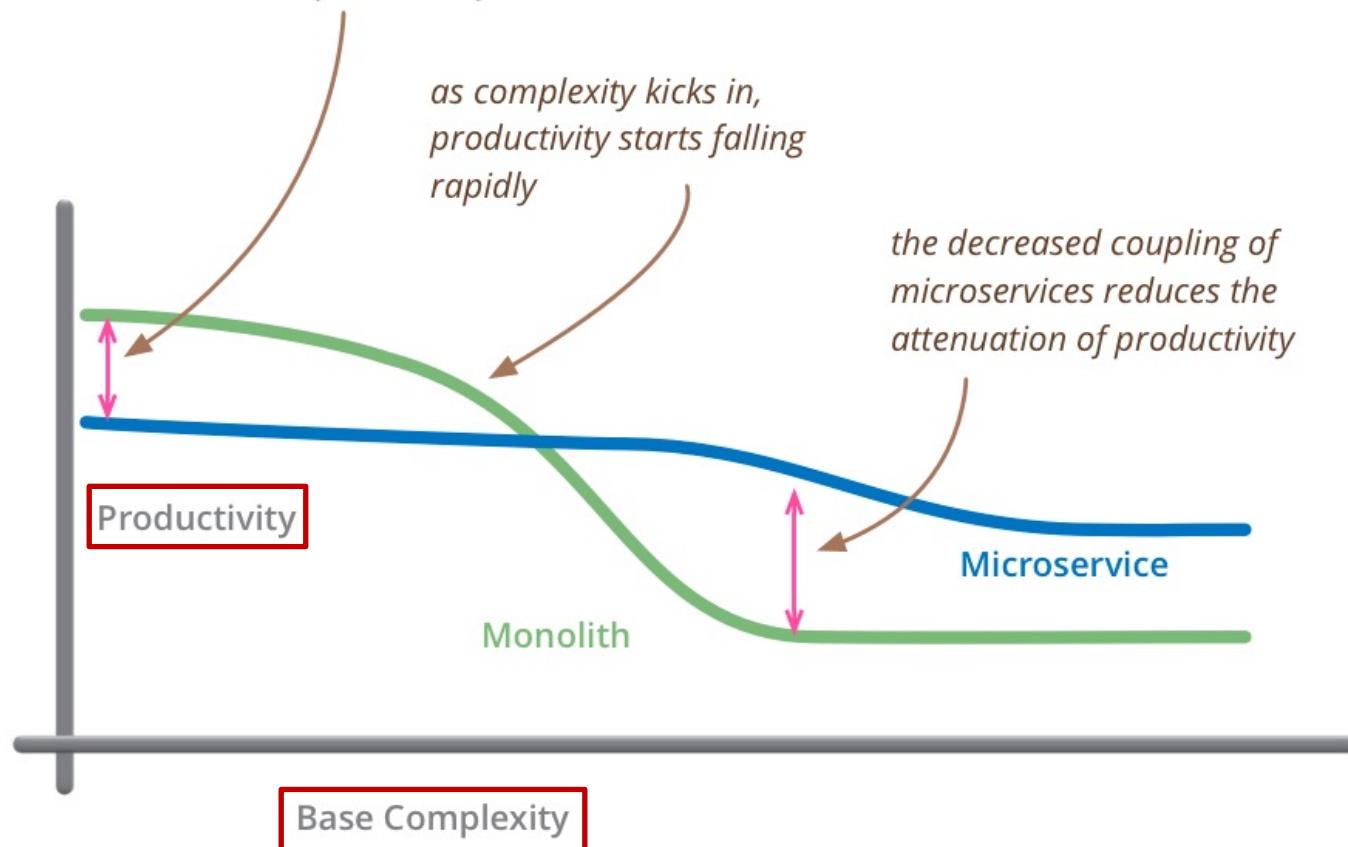
# Microservices

- Building applications as suite of small and easy to replace services
  - fine grained, one functionality per service  
(sometimes 3-5 classes)
  - composable
  - easy to develop, test, and understand
  - fast (re)start, fault isolation
  - modelled around business domain
- Interplay of different systems and languages
- Easily deployable and replicable
- Embrace automation, embrace faults
- Highly observable

# Are microservices always the right choice?

# Microservices overhead

*for less-complex systems, the extra baggage required to manage microservices reduces productivity*



# Microservice challenges

- Complexities of distributed systems
  - network latency, faults, inconsistencies
  - testing challenges
- Resource overhead, RPCs
  - Requires more thoughtful design (avoid "chatty" APIs, be more coarse-grained)
- Shifting complexities to the network
- Operational complexity
- Frequently adopted by breaking down monolithic application
- HTTP/REST/JSON communication
  - Schemas?

# Serverless



# Serverless (Functions-as-a-Service)

- Instead of writing minimal services, write just functions
- No state, rely completely on cloud storage or other cloud services
- Pay-per-invocation billing with elastic scalability
- Drawback: more ways things can fail, state is expensive
- Examples:  
AWS Lambda, CloudFlare workers, Azure Functions
- What might this be good for?

# More in: API testing and DevOps

