Async programming: GCD

The four horsemen of asynchronicity: Sync, Async, Serial and Concurrent

by Fernando Olivares

Agenda

1-hour session

- Author Introduction
- Parallelism vs. Concurrency
- What is GCD?
- Live coding
- Conclusion & Where to go from here
- Q&A

Author Introduction

Fernando

- ~10 years of experience
- Worked at small startups (1SecondEveryday) to publicly traded companies (J2 Global Inc.)
- Instructor at Big Nerd Ranch, Bloc bloc.io, Lambda School
- Won a few awards: The Storyteller Within (Apple), ERA Accelerator Top 10 (ERA NY)
- Product and Project experience
- iOS-only
- @fromJrToSr

Synchronous programming

A long time ago... in a taco truck far, far away.

Rock Stars

Super heroes

Ordering

Taco Truck

Synchronous programming

First come, first serve

Super heroes

Rock Stars

Ordering

Taco Truck

Synchronous programming

Fist come, first serve

- Disadvantages:
  - We only have one chef.

Super heroes

Rock Stars

Ordering

Taco Truck

Synchronous programming

Fist come, first serve

- Disadvantages:
  - We only have one chef.
  - If a someone takes too long, the rest of the line suffers.

Super heroes

Rock Stars

Taco Truck

Synchronous programming

Fist come, first serve

- Possible solution:
  - Prepare the dishes partially.
  - Order only matters within a party.

Super heroes

Rock Stars

Ordering

Taco Truck

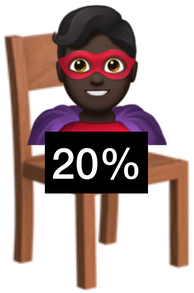
Synchronous programming

Fist come, first serve

- Possible solution:
  - Prepare the dishes partially.
  - Order only matters within a party.

Super heroes

Rock Stars



Ordering

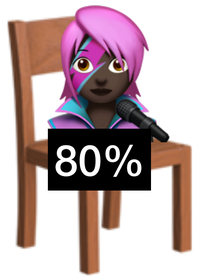
Taco Truck

Synchronous programming

Illusion of one chef serving different parties.

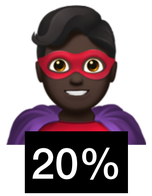
Super heroes

Rock Stars



Ordering

Taco Truck



Synchronous programming

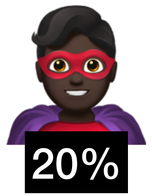
Illusion of one chef serving different parties.

Super heroes

Rock Stars

Ordering

Taco Truck

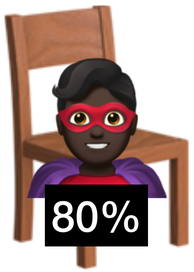


Synchronous programming

Illusion of one chef serving different parties.

Super heroes

Rock Stars



Ordering

Taco Truck

Synchronous programming

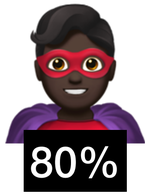
Illusion of one chef serving different parties.

Super heroes

Rock Stars

Ordering

Taco Truck



Synchronous programming

Everyone ate tacos and left.

Ordering

Taco Truck

Synchronous programming

A long time ago... in a single-core computer.

Algorithm A

Algorithm B

Thread

Core

Synchronous programming

FIFO - First in, First out

Algorithm B

Algorithm A

Thread

Core

Synchronous programming

FIFO - First in, First out

- Disadvantages:
  - We only have one core.

Algorithm B

Algorithm A

Thread

Core

Synchronous programming

FIFO - First in, First out

- Disadvantages:
  - We only have one core.
  - If some code takes too long, the app freezes.

Algorithm B

Algorithm A

Core

Synchronous programming

FIFO - First in, First out

- Possible solution:
  - Execute parts of the code partially. (Pseudoparallelism)
  - The order only matters within an algorithm.

Algorithm B

Algorithm A

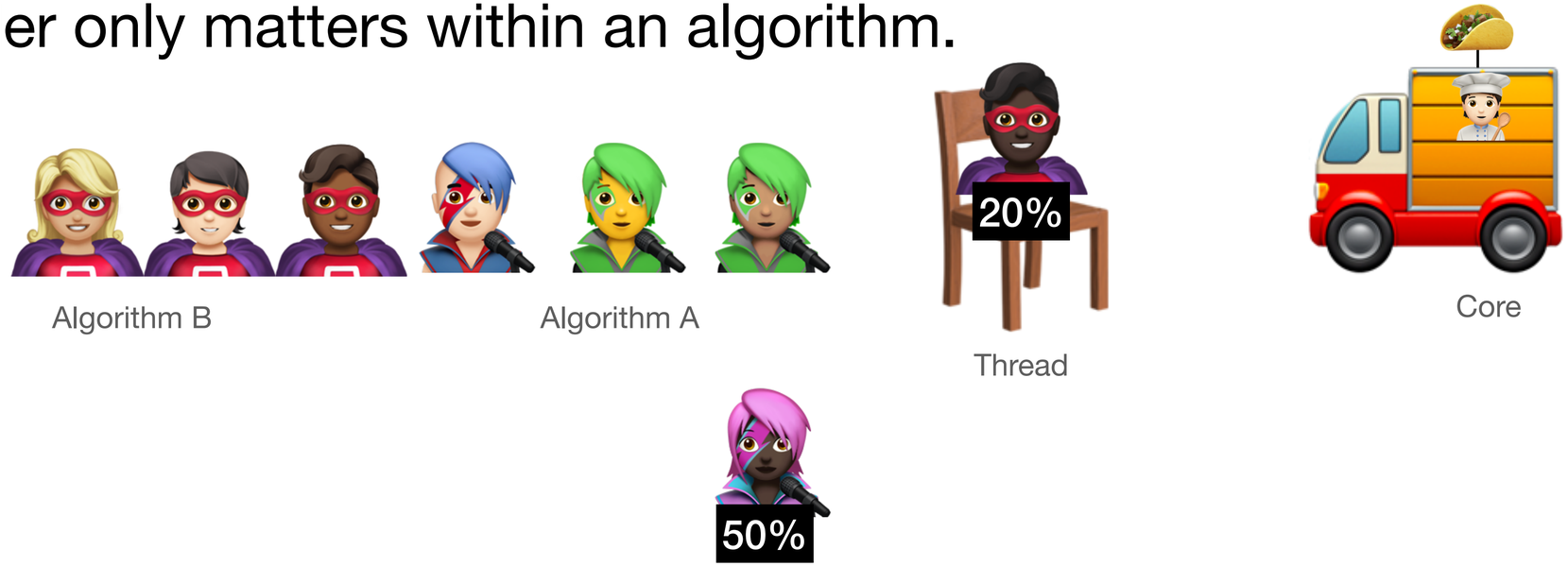
Thread

Core

Synchronous programming

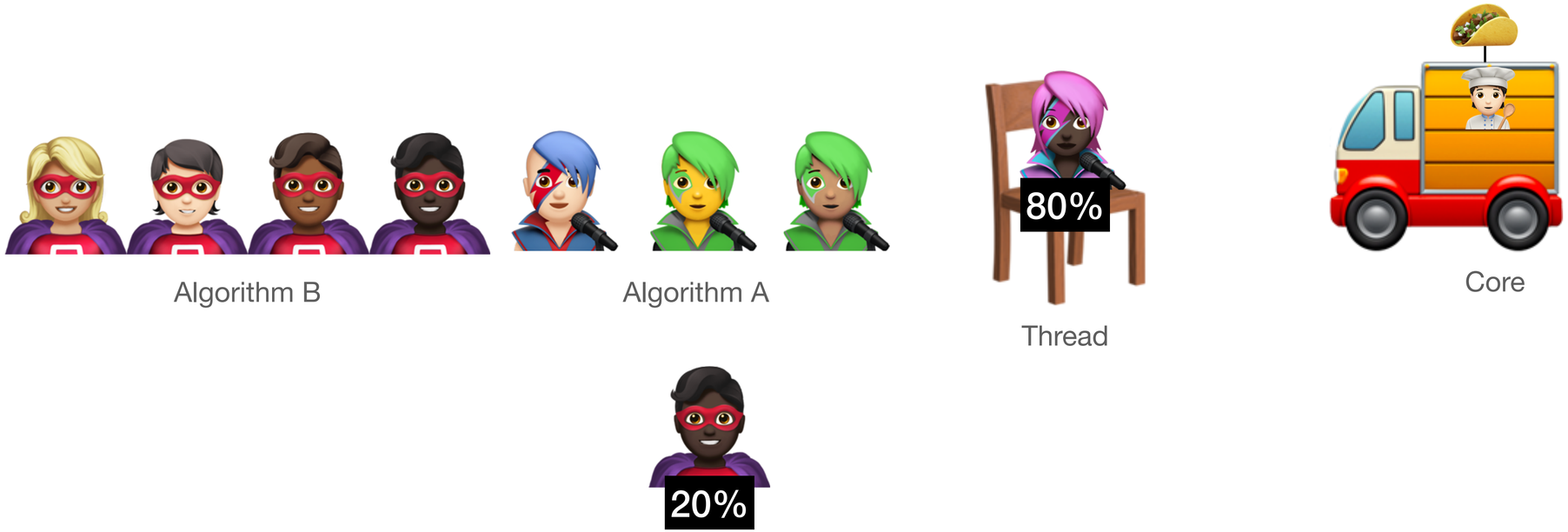
FIFO - First in, First out

- Possible solution:
  - Execute parts of the code partially.
  - The order only matters within an algorithm.



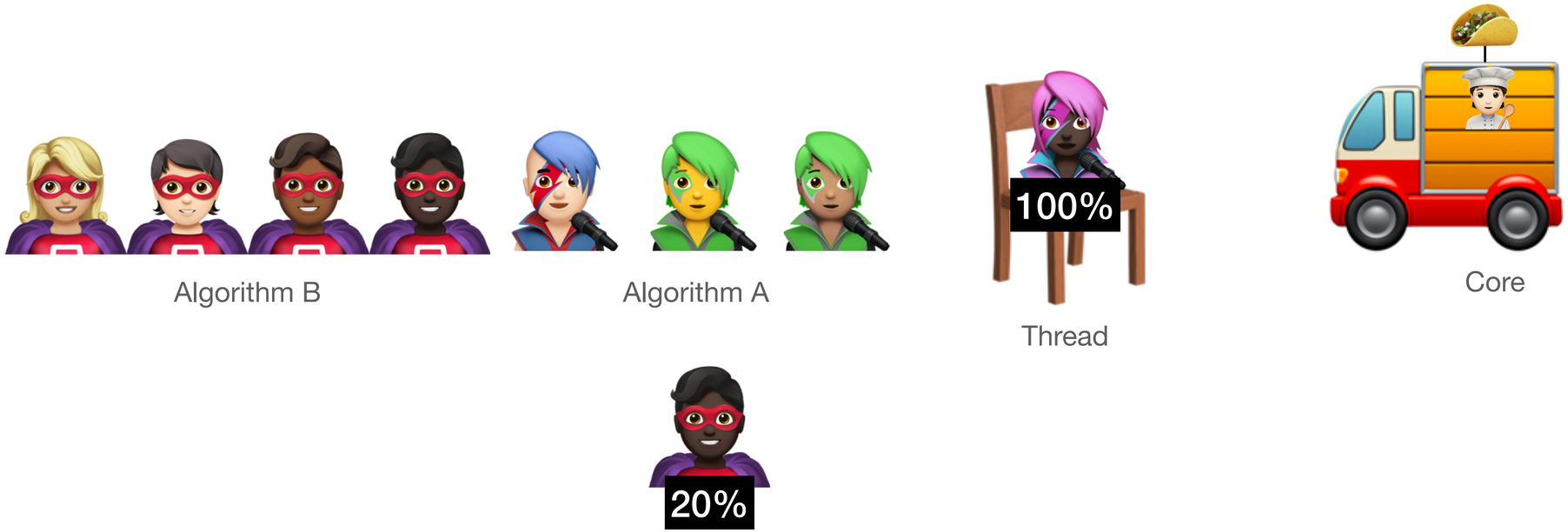
Synchronous programming

Illusion of one core performing several algorithms.



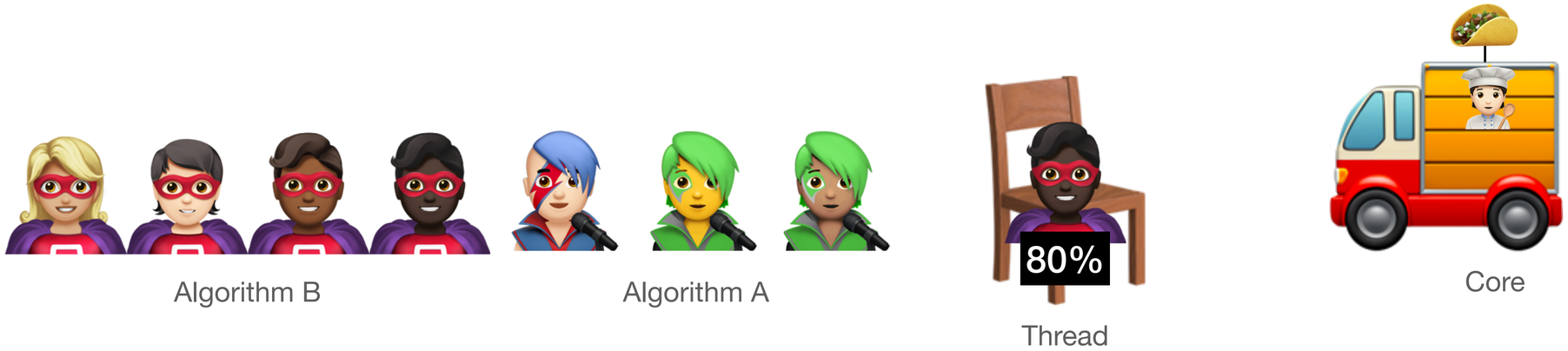
Synchronous programming

Illusion of one core performing several algorithms.



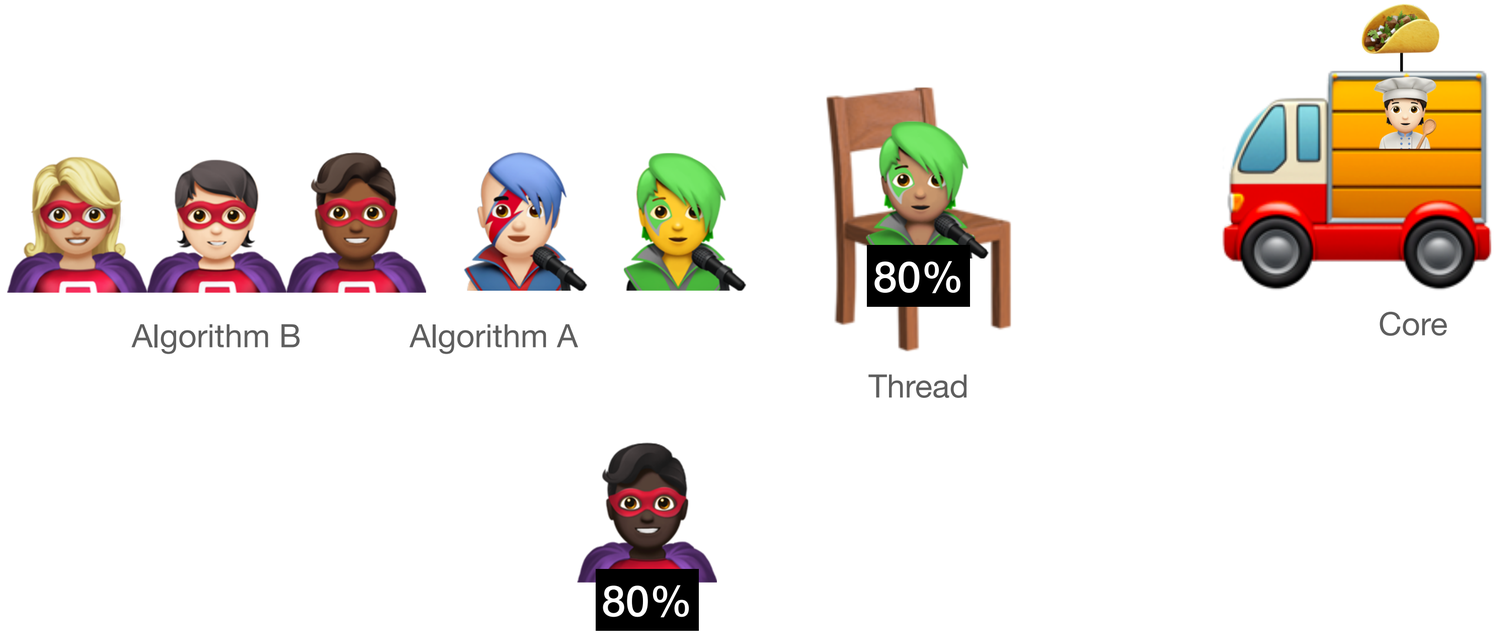
Synchronous programming

Illusion of one core performing several algorithms.



Synchronous programming

Illusion of one core performing several algorithms.



Synchronous programming

All code has been executed.

Thread

Core

Parallelism vs. Concurrency

Fernando

- **Pseudo-parallelism** is the ilusion that multiple tasks can be executed at the same time.
  - This is the case for single-thread environments.
- True parallelism means that multiple tasks can be executed at the same time.
  - This requires a multi-threaded environment.
- Concurrency means that a task can be completed in any order without affecting the outcome.
  - This excludes algorithms that must be sequentially executed.

Asynchronous programming

Tacos today.

Rock Stars

Host(ess)

You decide where each party is going.

Reservation

Taco Truck

Super Heroes

Walk-in

Taco Truck

Asynchronous programming

First come, first serve

Super Heroes (Walk-in) Rock Stars (Reservation)

Reservation

Walk-in

Core

Core

Asynchronous programming

First come, first serve

Super Heroes (Walk-in)

Reservation

Walk-in

Core

Core

Asynchronous programming

First come, first serve

Super Heroes (Walk-in)

Reservation

Walk-in

Core

Core

Asynchronous programming

First come, first serve

Super Heroes (Walk-in)

Reservation

Walk-in

Core

Core

Asynchronous programming

First come, first serve

Super Heroes (Walk-in)

Reservation

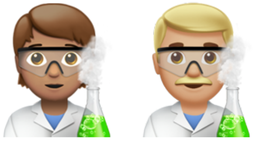
Walk-in

Core

Core

Asynchronous programming

First come, first serve



Scientists (Reservation)

Super Heroes (Walk-in)

Reservation

Walk-in

Core

Core

Asynchronous programming

First come, first serve

Scientists (Reservation)

Super Heroes (Walk-in)

Reservation

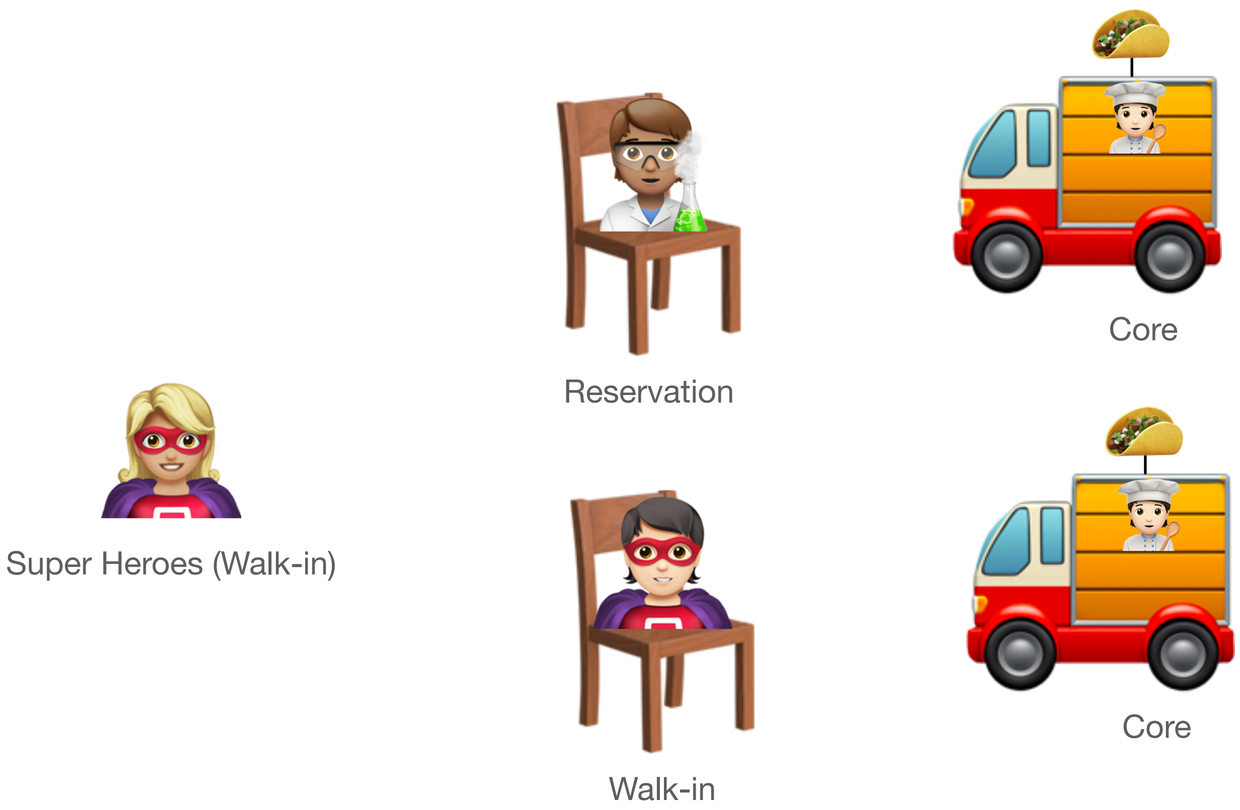
Core

Walk-in

Core

Asynchronous programming

First come, first serve



Asynchronous programming

First come, first serve

Reservation

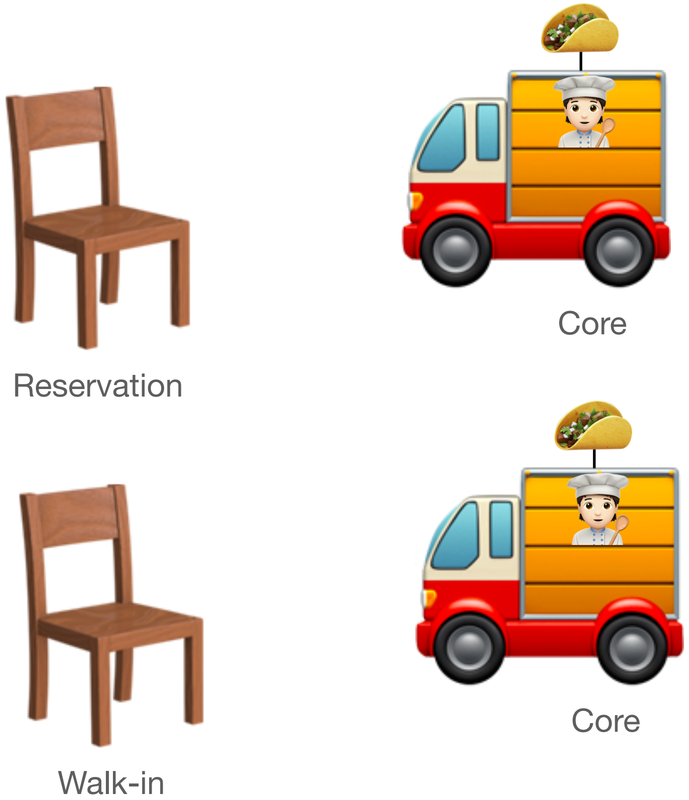
Core

Walk-in

Core

Asynchronous programming

All done.



Asynchronous programming

Threading today.

UI updates, animations, drawing

Dev

You decide where each algorithm is going.

Main Thread

Core

JSON Parsing

Background Thread

Core

Asynchronous programming

The UI must always be updated on the main thread.

JSON Parsing (Background thread) UI (Main Thread)

Main Thread

Core

Background Thread

Core

Asynchronous programming

The UI must always be updated on the main thread.

JSON Parsing (Background Thread)

Main Thread

Background Thread

Core

Core

Asynchronous programming

The UI must always be updated on the main thread.

JSON Parsing (Background Thread)

Main Thread

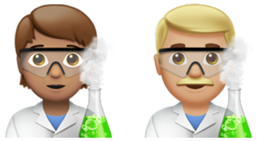
Background Thread

Core

Core

Asynchronous programming

The UI must always be updated on the main thread.



Animations (Main Thread)

JSON Parsing (Background Thread)

Main Thread

Background Thread

Core

Core

Asynchronous programming

The UI must always be updated on the main thread.

Scientists (Main Thread)

Super Heroes
(Background Thread)

Main Thread

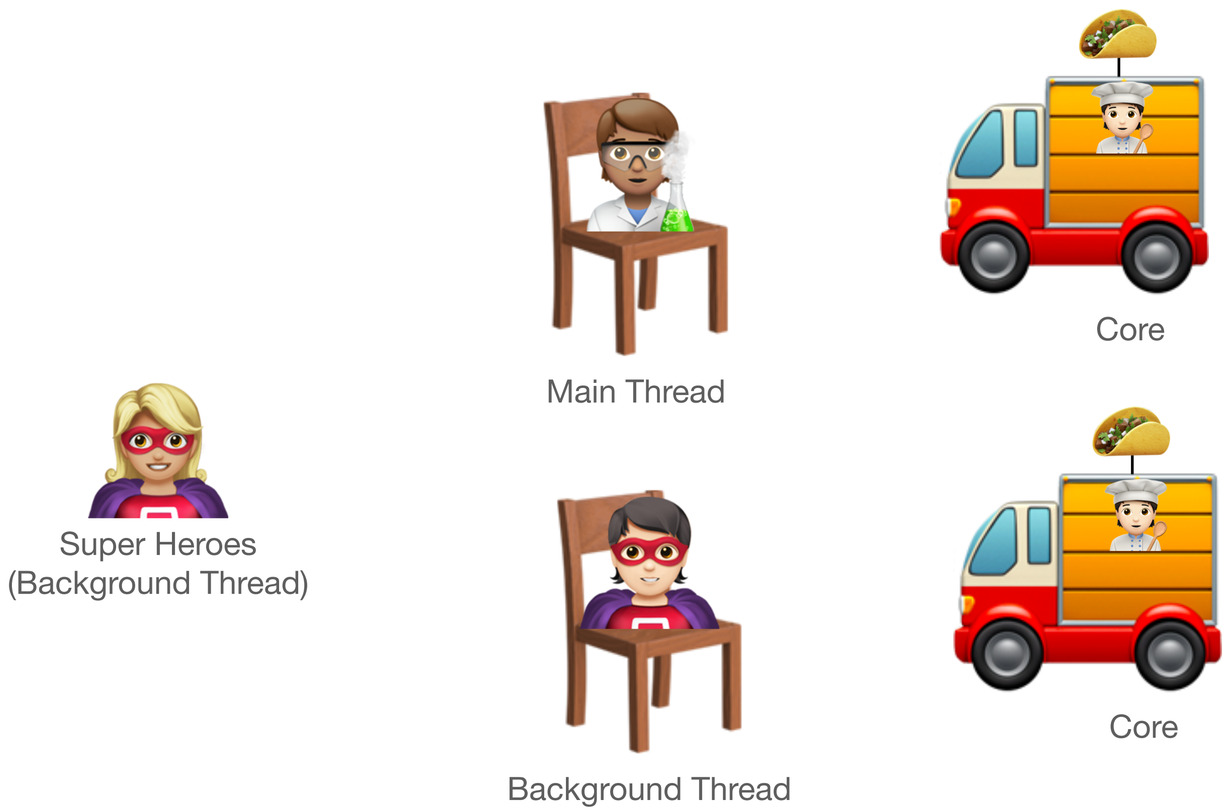
Core

Background Thread

Core

Asynchronous programming

The UI must always be updated on the main thread.



Asynchronous programming

The UI must always be updated on the main thread.

Main Thread

Core

Background Thread

Core

Asynchronous programming

The UI must always be updated on the main thread.

Main Thread

Core

Background Thread

Core

Parallelism vs. Concurrency

Fernando

- Pseudo-parallelism is the ilusion that multiple tasks can be executed at the same time.
  - This is the case for single-thread environments.
- **True parallelism** means that multiple tasks can be executed at the same time.
  - This requires a multi-threaded environment.
- Concurrency means that a task can be completed in any order without affecting the outcome.
  - This excludes algorithms that must be sequentially executed.

Asynchronous programming

What if programming is hard?

UI updates, animations (and JSON parsing?)

JSON Parsing (with animations?)

Dev

Humans aren't good at predicting who will take a long time ordering when the orders are mixed.

Main Thread

Background Thread

Background Thread

Core

Core

Core

Asynchronous programming

"We can solve any problem by introducing an extra level of indirection."

UI updates, animations (and JSON parsing?)

JSON Parsing (with animations?)

Dev

Humans aren't good at predicting who will take a long time ordering when the orders are mixed.

Main Thread

Core

Background Thread

Core

Background Thread

Core

Asynchronous programming

"We can solve any problem by introducing an extra level of indirection."

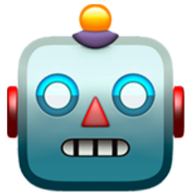
UI updates, animations (and JSON parsing?)

JSON Parsing (with animations?)

Dev

Main Queue

Background Queue



GCD

Main Thread

Background Thread

Background Thread

Core

Core

Core

What is GCD?

In theory, it's easy concurrency

- Official definition: "provides comprehensive support for concurrent code execution on multicore hardware."[1]
- Handles tasks (i.e. closures) and passes them along to queues that execute them in synchronous or asynchronous order.
- Queues can be serial or concurrent.
- Used extensively for long-running tasks that would block the main queue (responsible for drawing).

[1] - https://apple.github.io/swift-corelibs-libdispatch/

Live Demo

Good ol' main queue.

Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

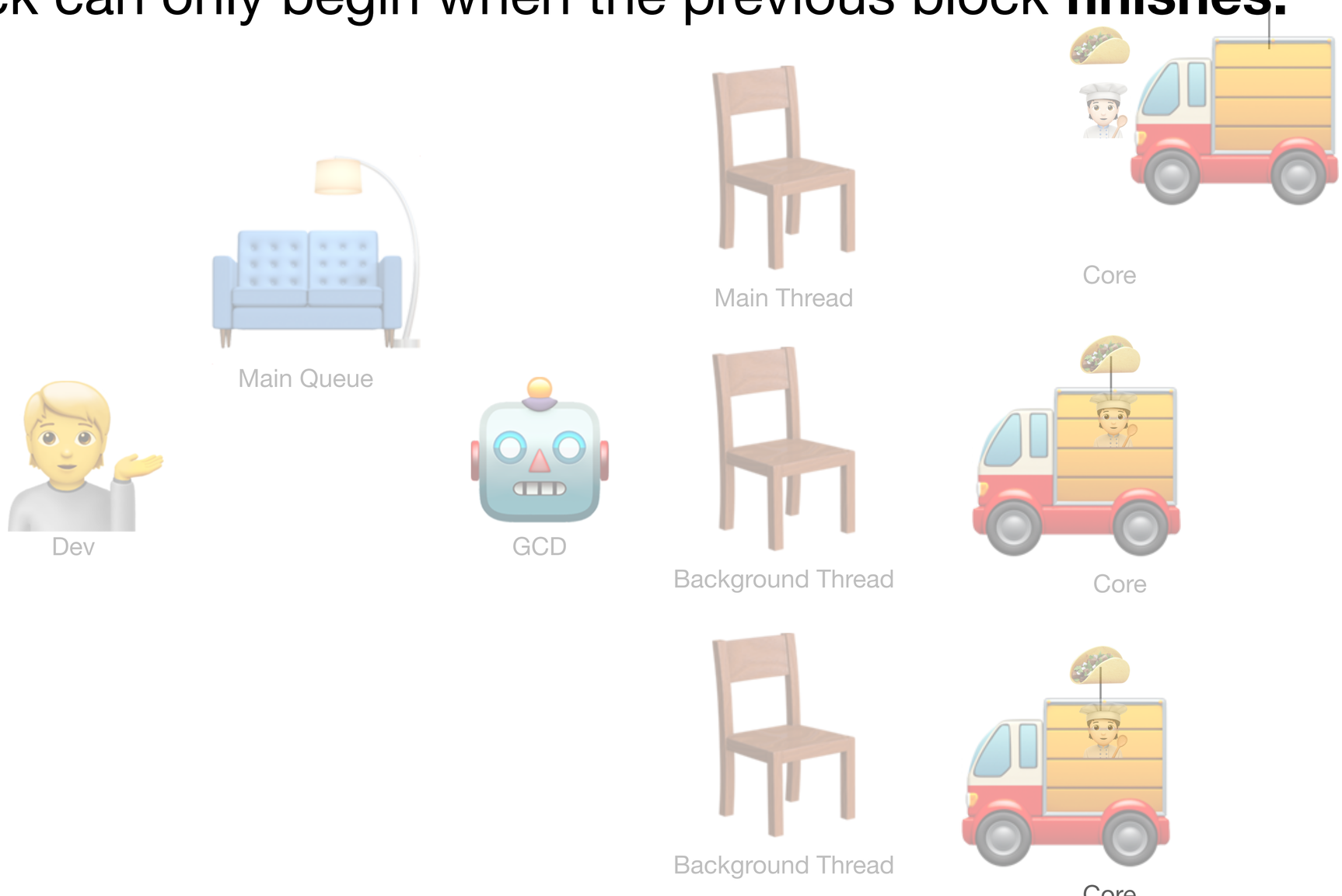
Code

0 `load entry index = 0`

1 `load entry index = 1`

...

9 `load entry index = 9`

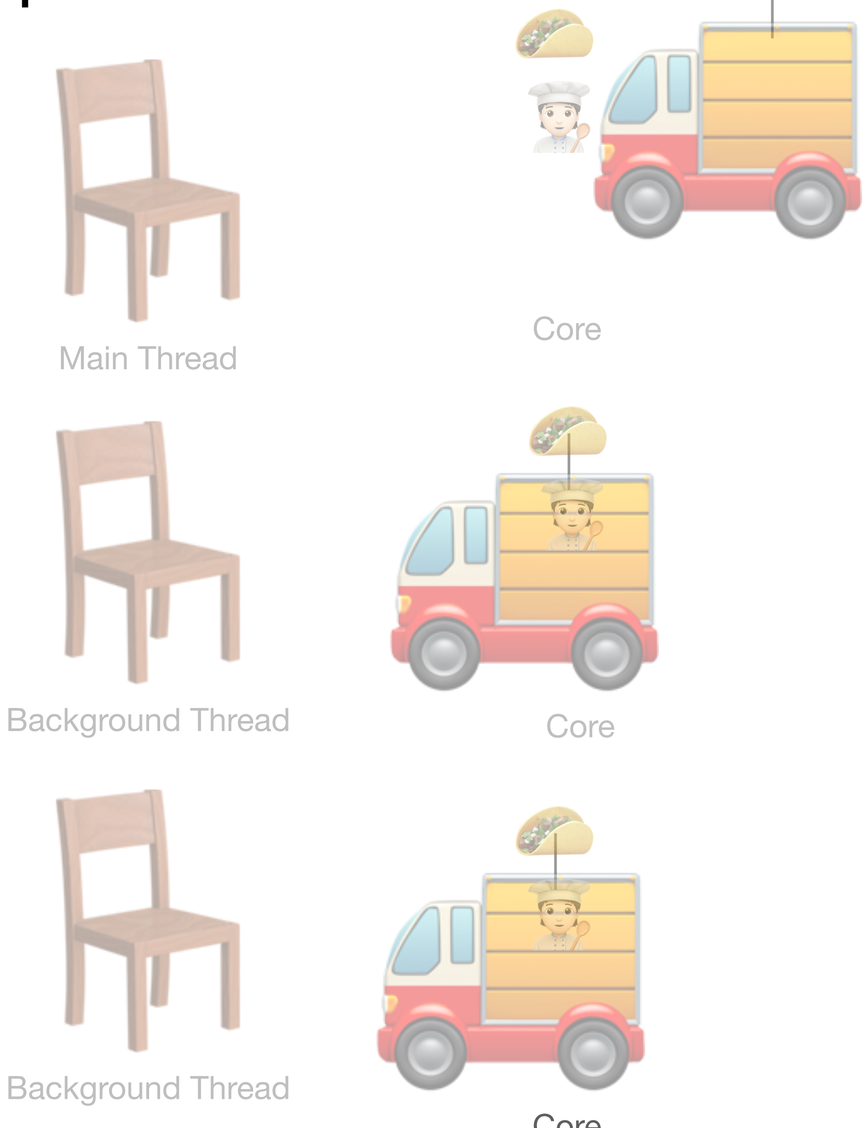


Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

Code

```
0  load entry index = 0
1  load entry index = 1
...
9  load entry index = 9
```

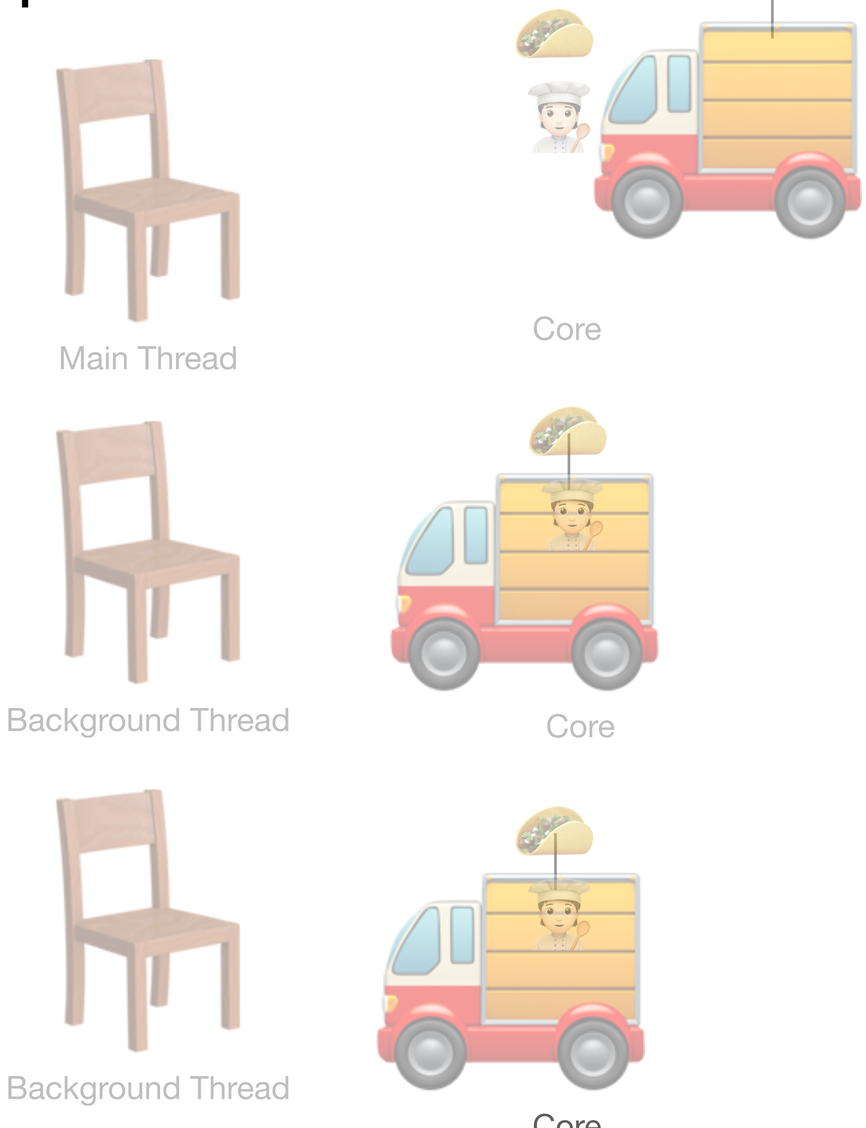


Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

Code

```
0  load entry index = 0
1  load entry index = 1
...
9  load entry index = 9
```



Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

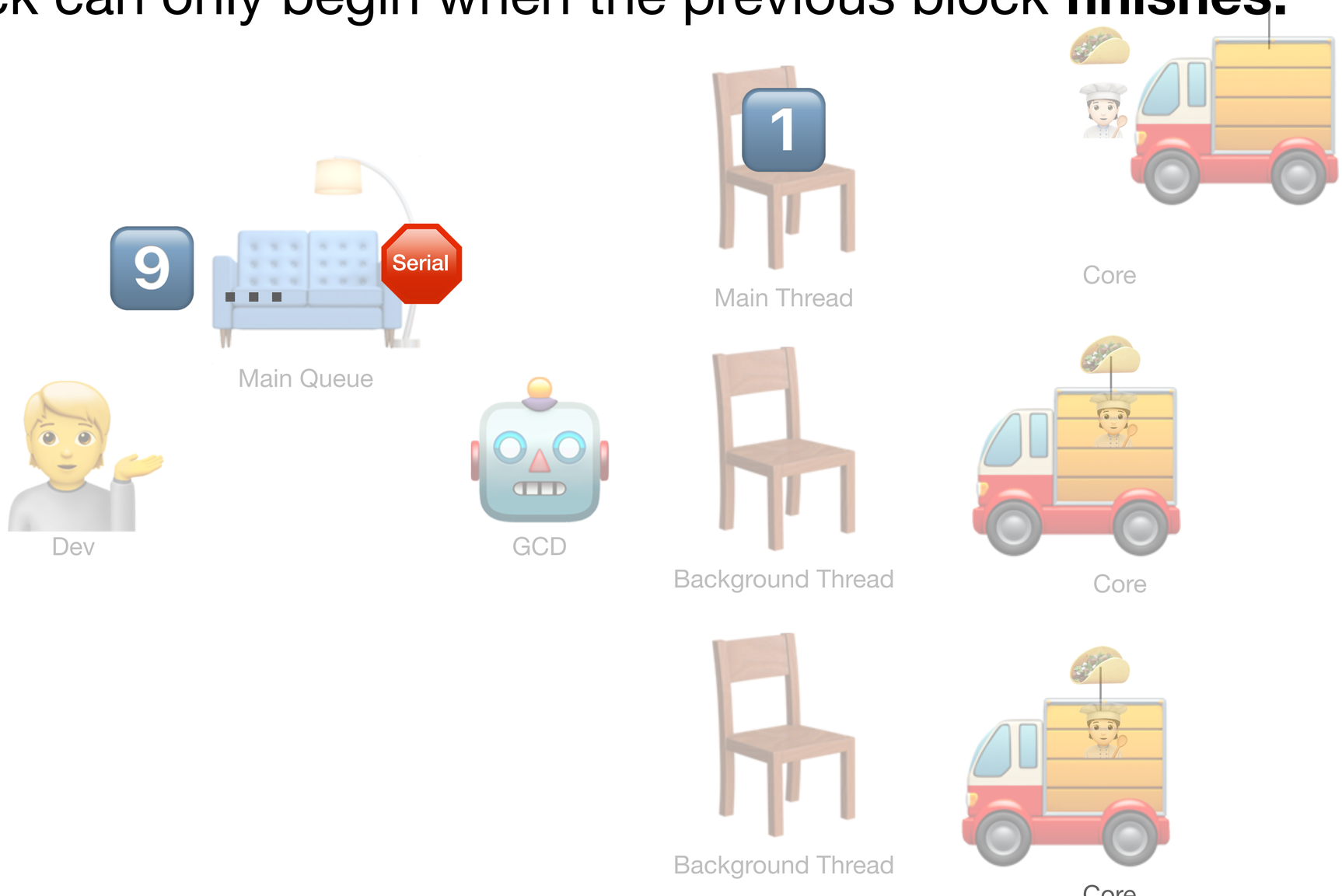
Code

0 `load entry index = 0`

1 `load entry index = 1`

...

9 `load entry index = 9`



Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

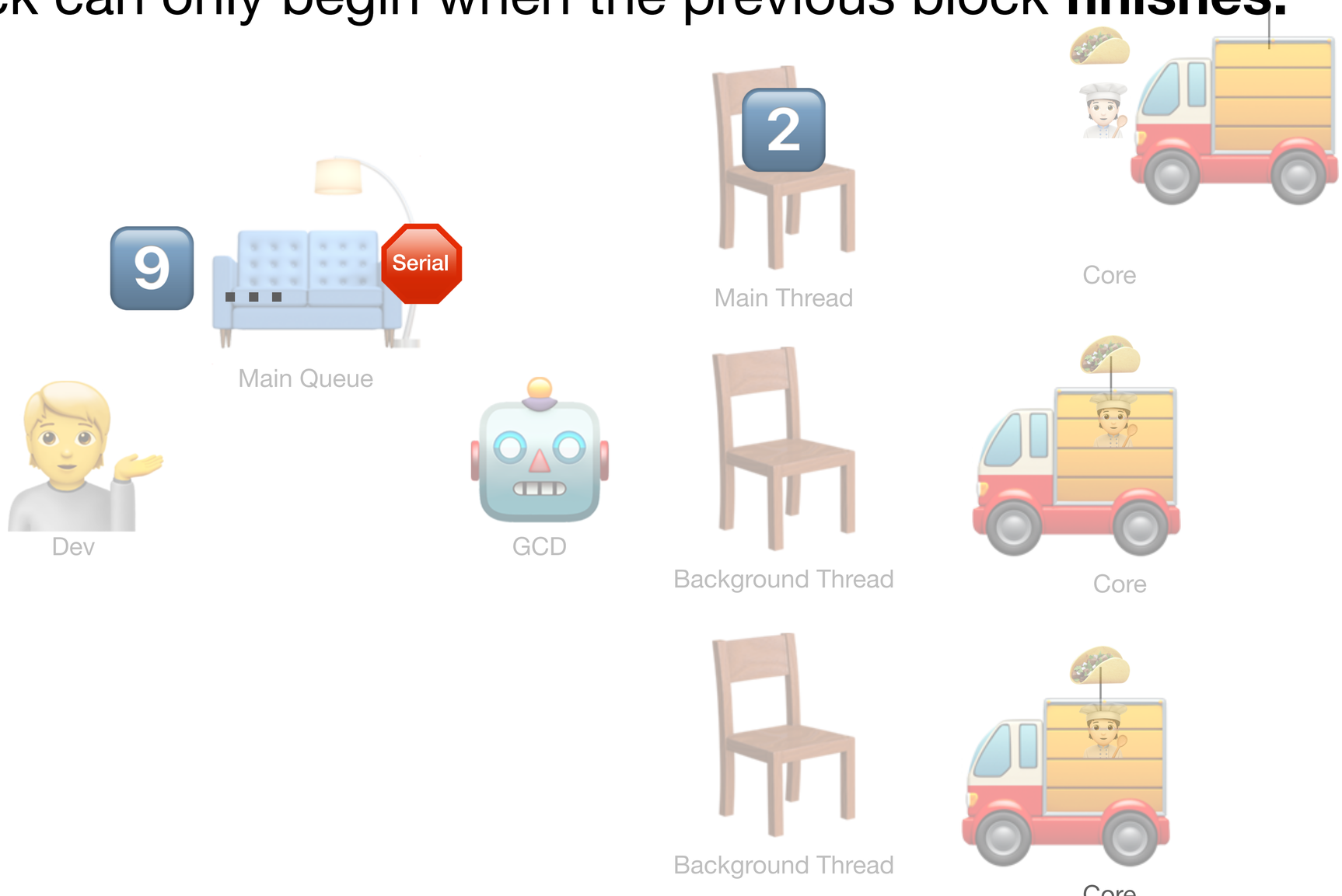
Code

0 `load entry index = 0`

1 `load entry index = 1`

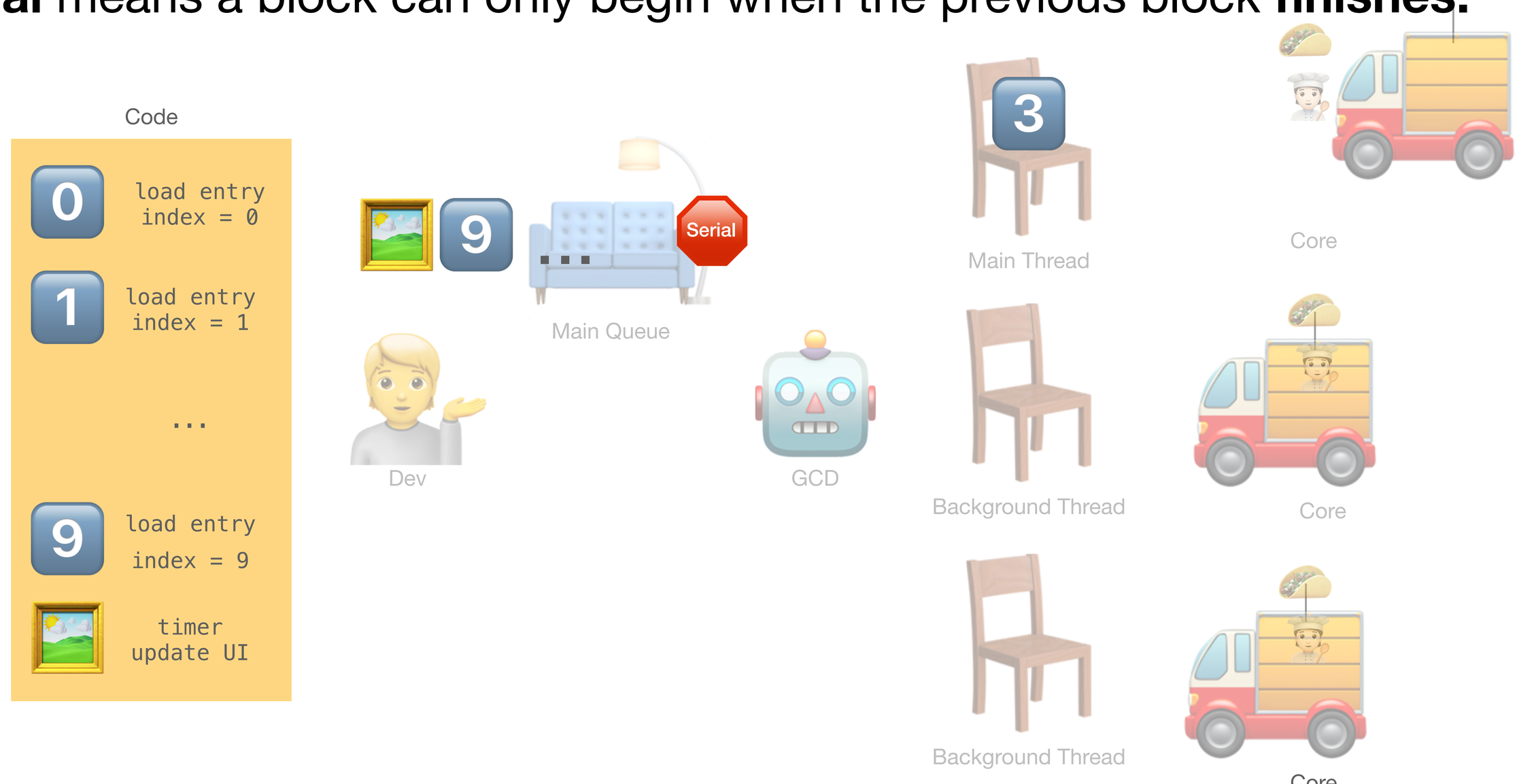
...

9 `load entry index = 9`



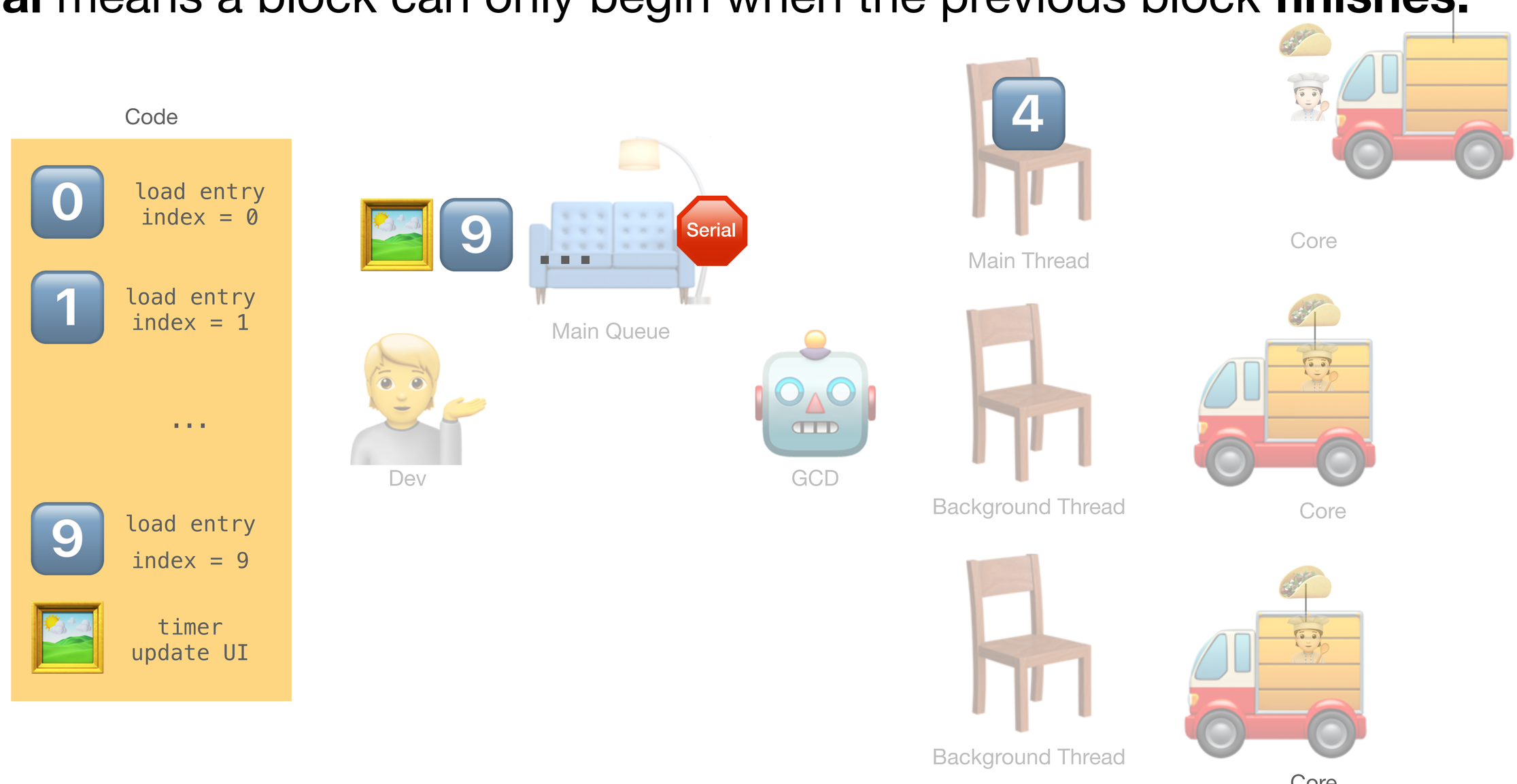
Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**



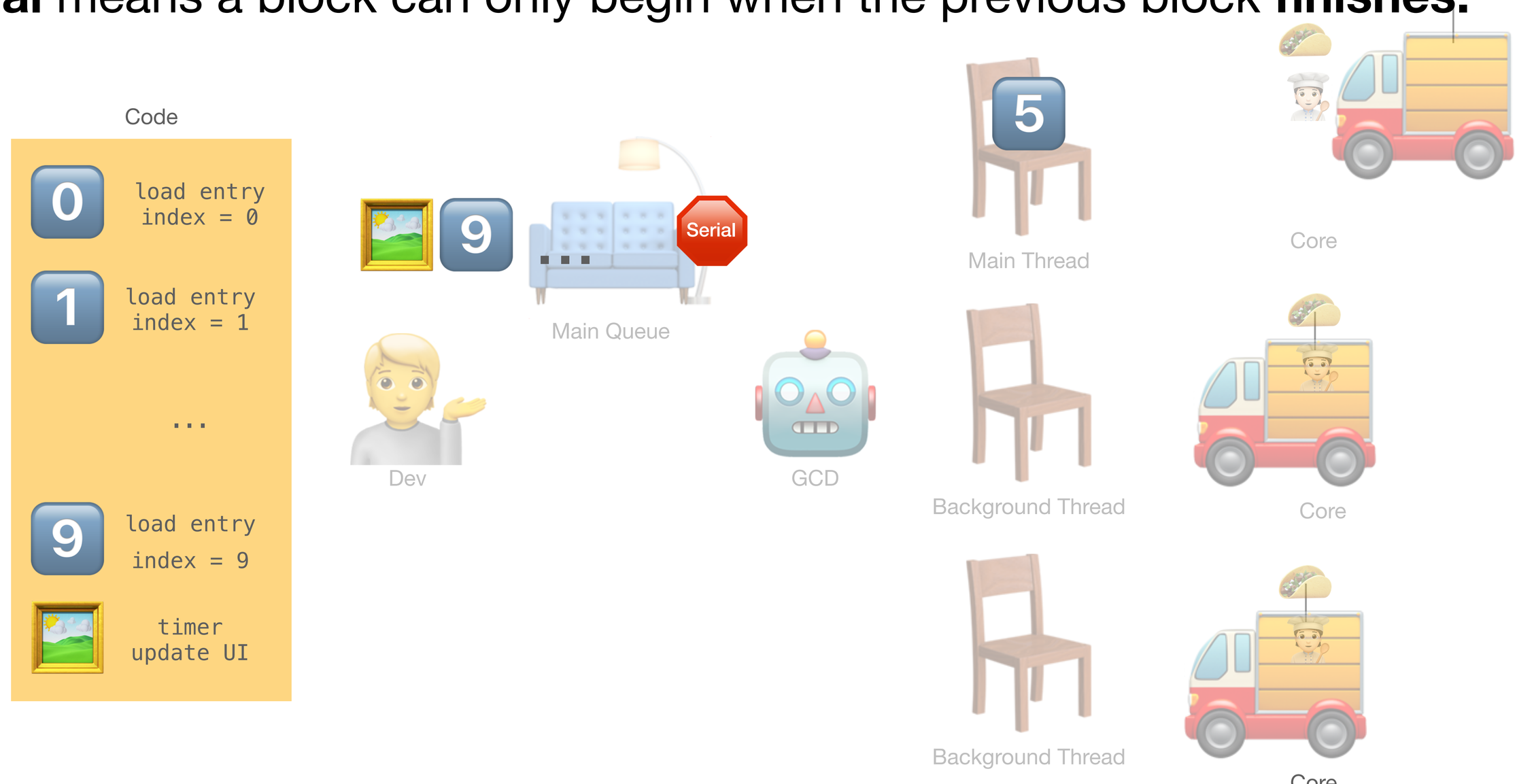
Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**



Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

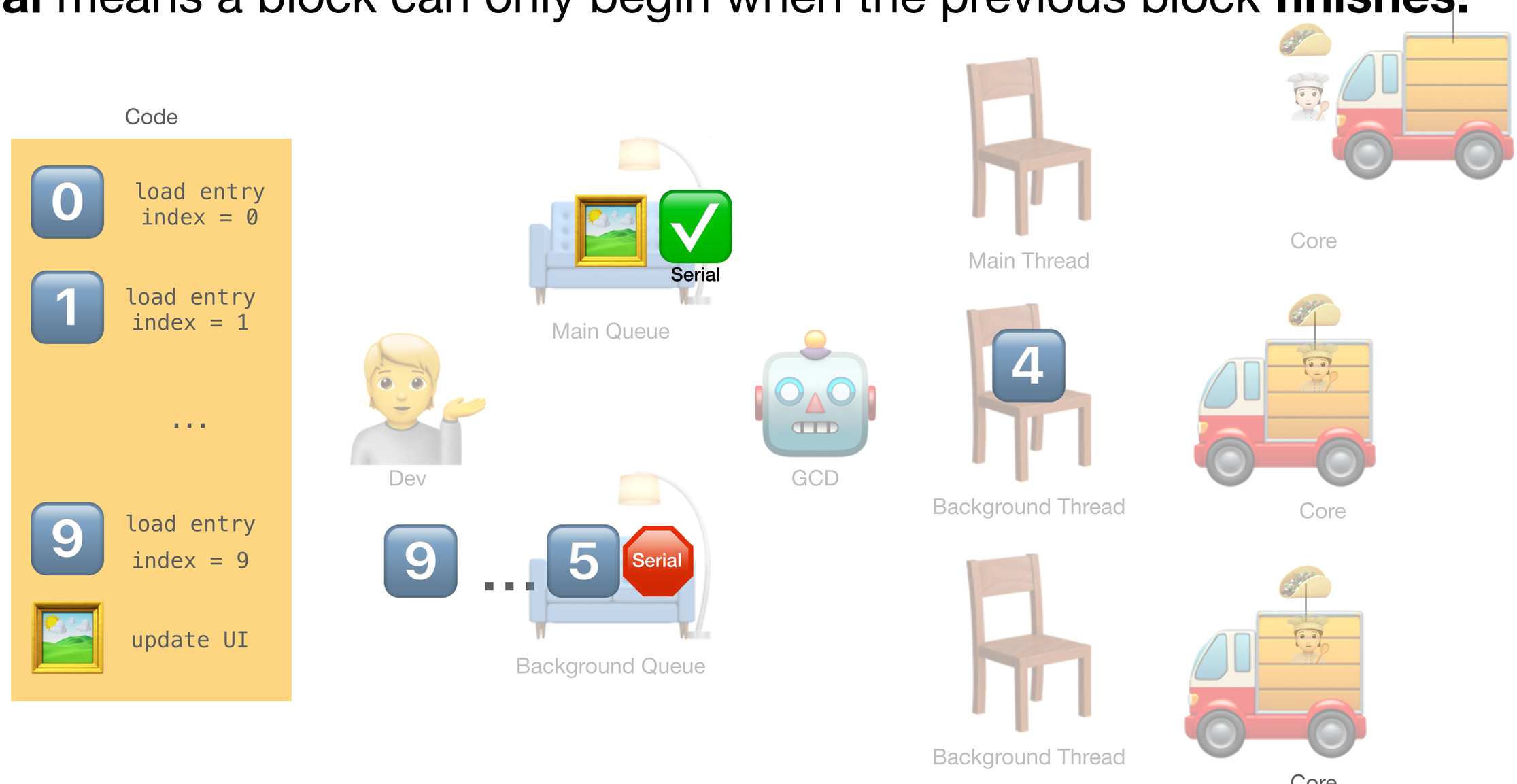


Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

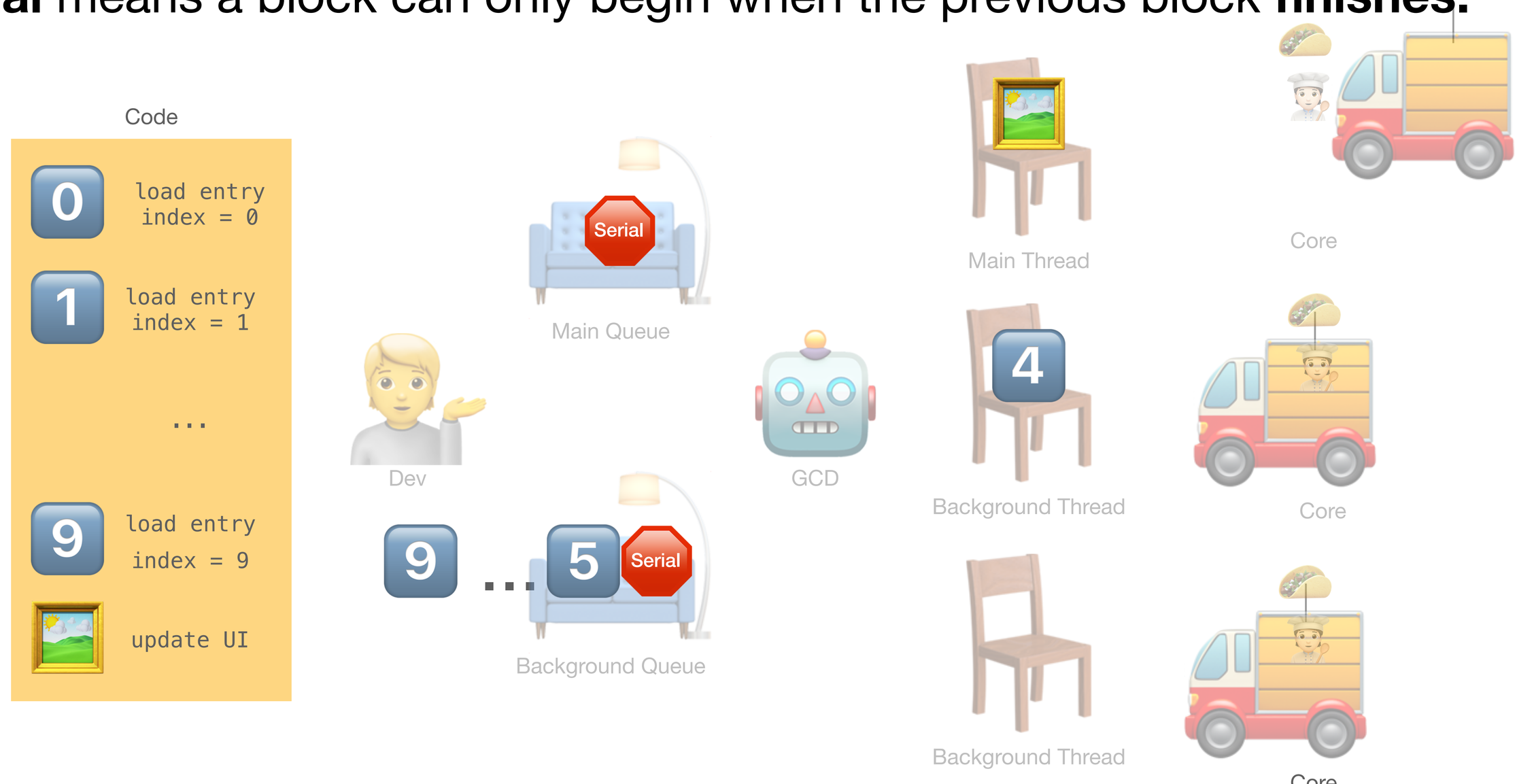
Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**



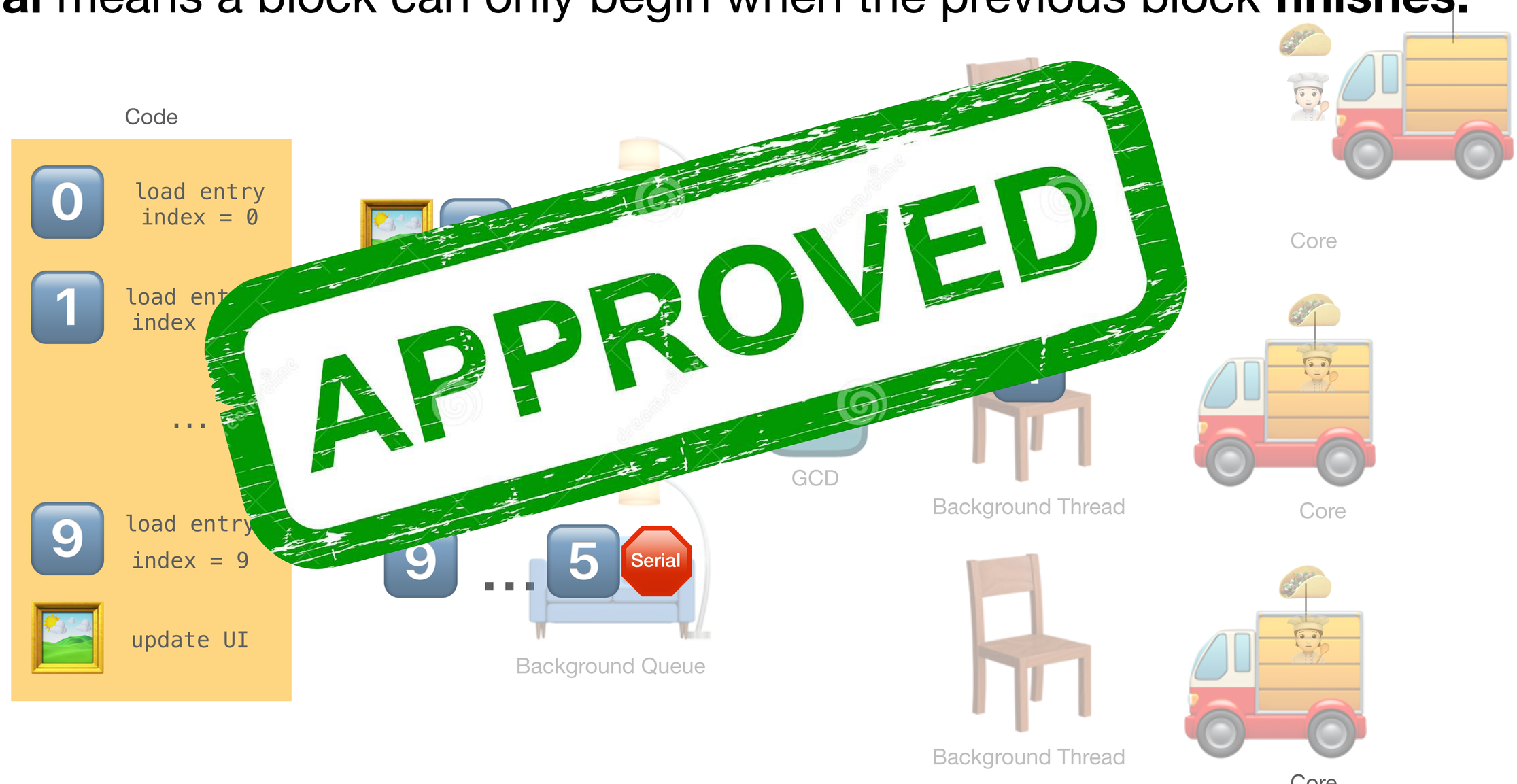
Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**



Single Serial Queue

Serial means a block can only begin when the previous block **finishes.**

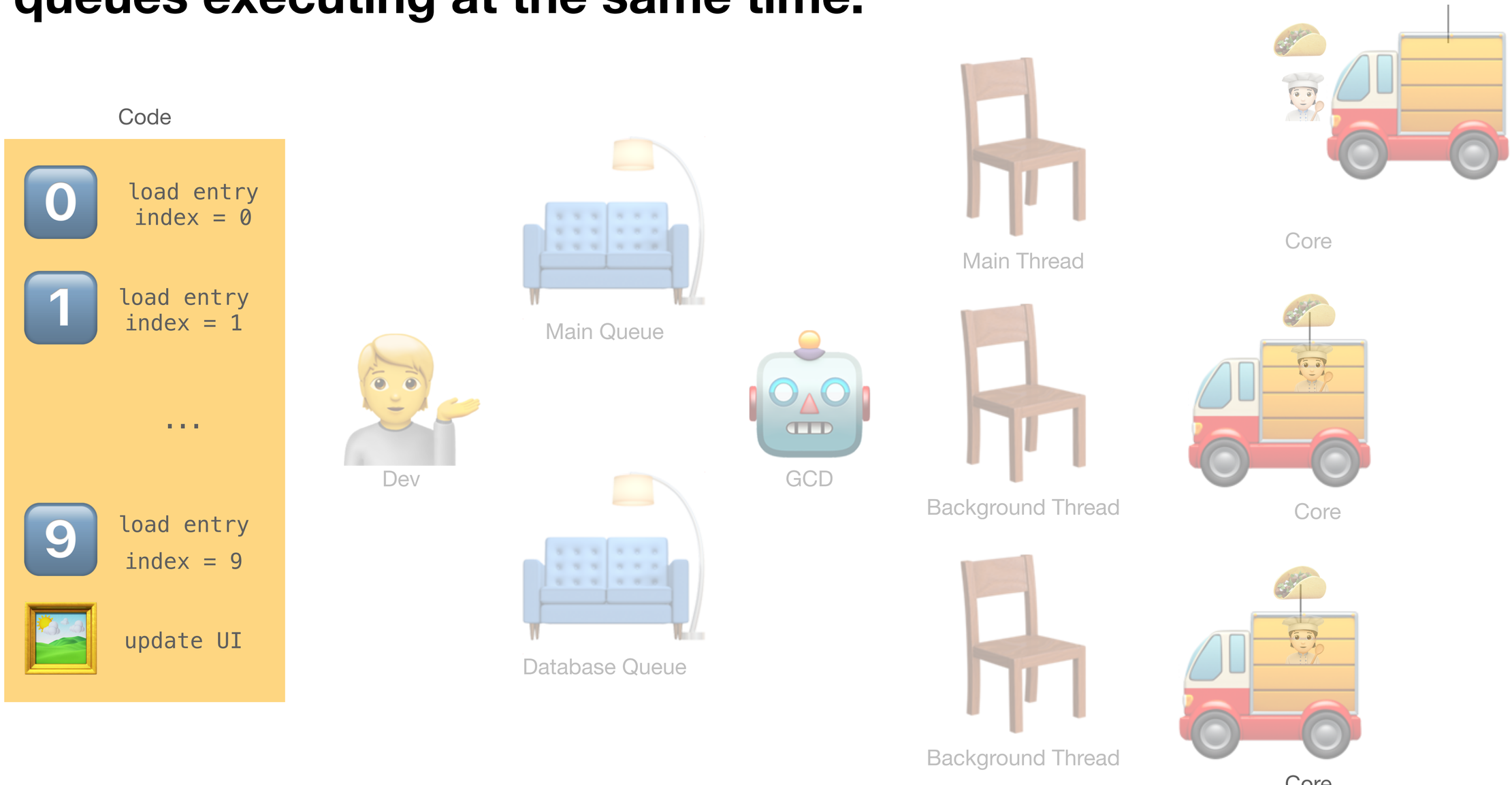


Live Demo

Adding a **background** queue

Parallel queues

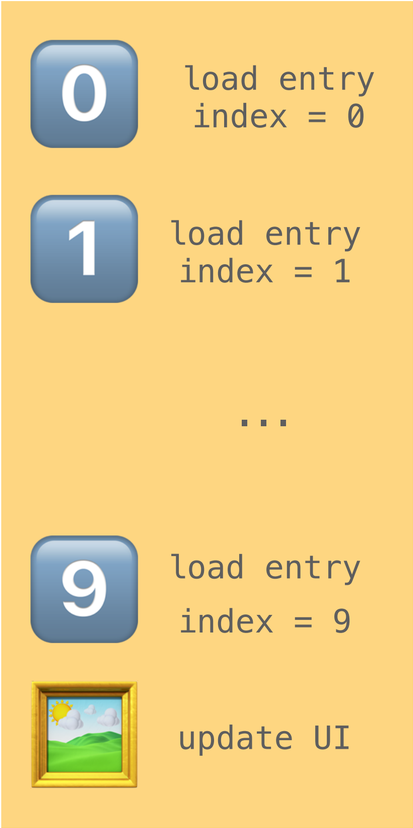
Two queues executing at the same time.

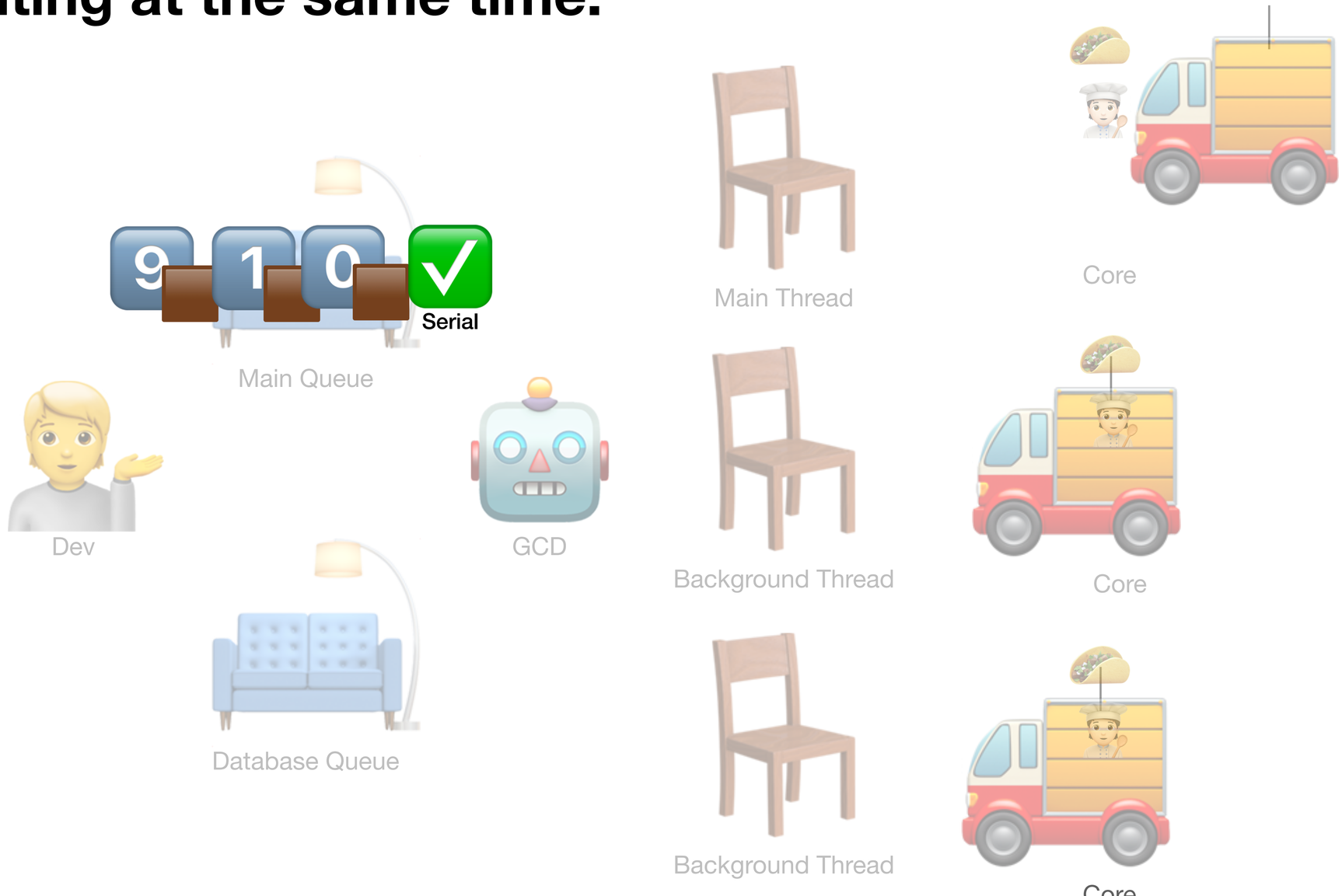


Parallel queues

Two queues executing at the same time.

Code





Parallel queues

Two queues executing at the same time.

Code

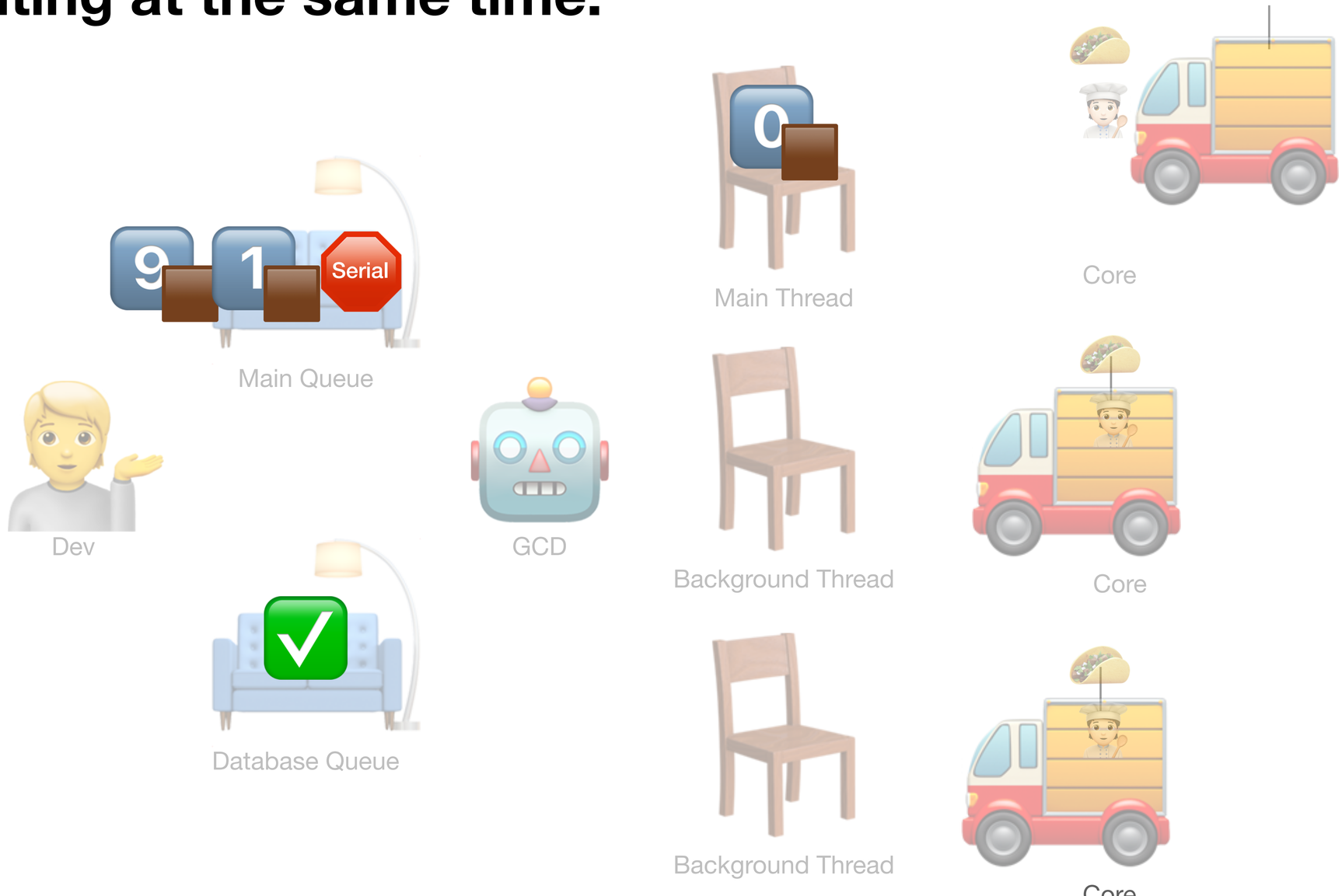
0 `load entry index = 0`

1 `load entry index = 1`

...

9 `load entry index = 9`

`update UI`



Parallel queues

Two queues executing at the same time.

Code

0 `load entry index = 0`

1 `load entry index = 1`

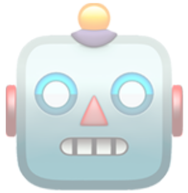
...

9 `load entry index = 9`

`update UI`

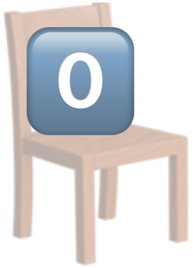
Main Queue

Dev

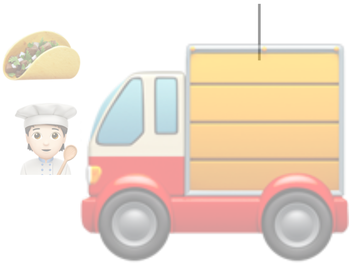


GCD

Database Queue



Main Thread



Core

Background Thread

Core

Background Thread

Core

Parallel queues

Two queues executing at the same time.

Code

0 `load entry index = 0`

1 `load entry index = 1`

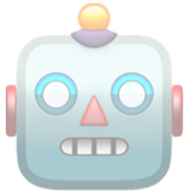
...

9 `load entry index = 9`

`update UI`

Main Queue

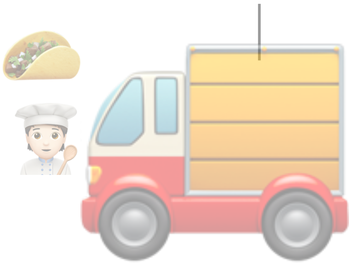
Dev



GCD

Database Queue

Main Thread



Core

Background Thread

Core

Background Thread

Core

Parallel queues

Two queues executing at the same time.

Code

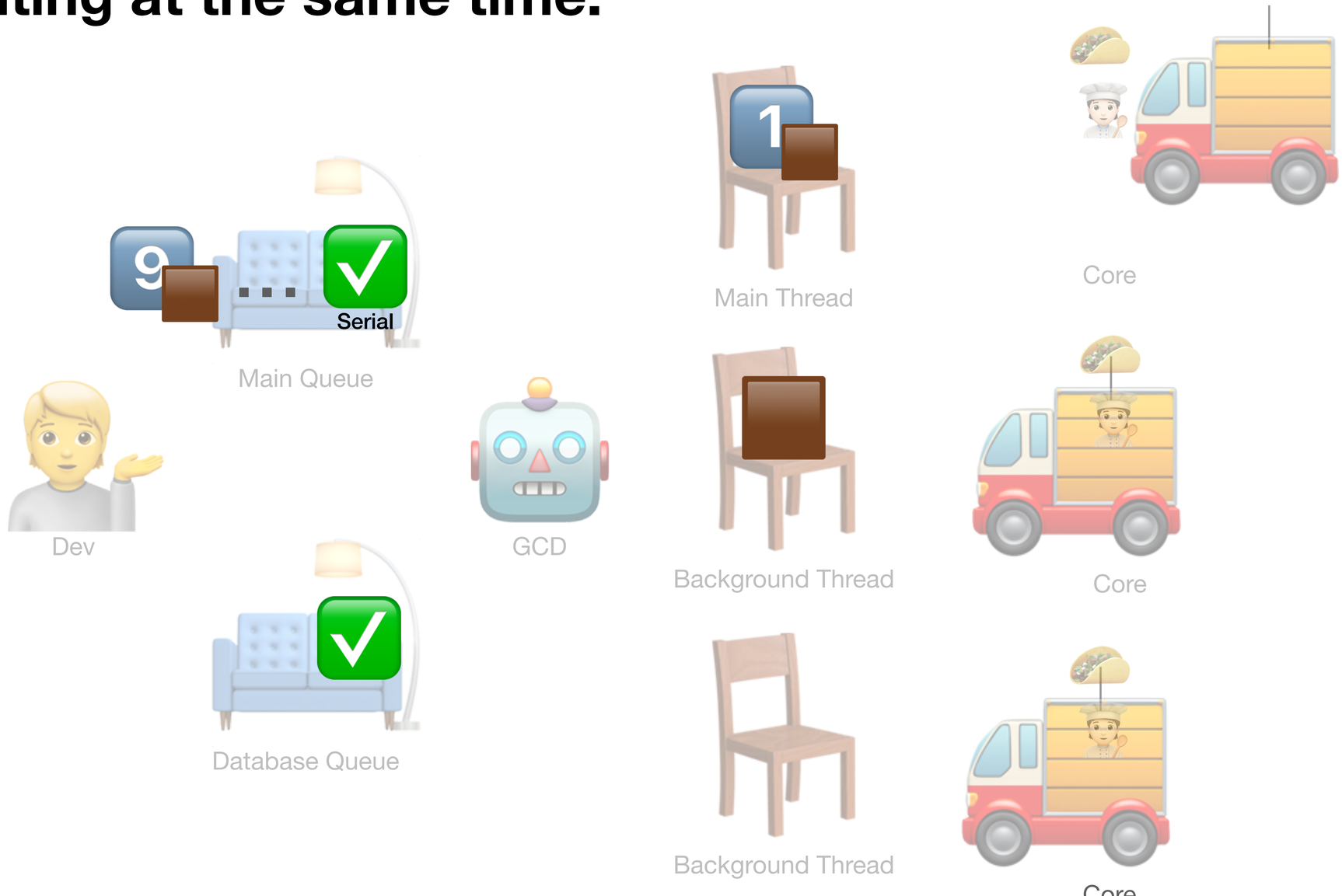
0 `load entry index = 0`

1 `load entry index = 1`

...

9 `load entry index = 9`

`update UI`

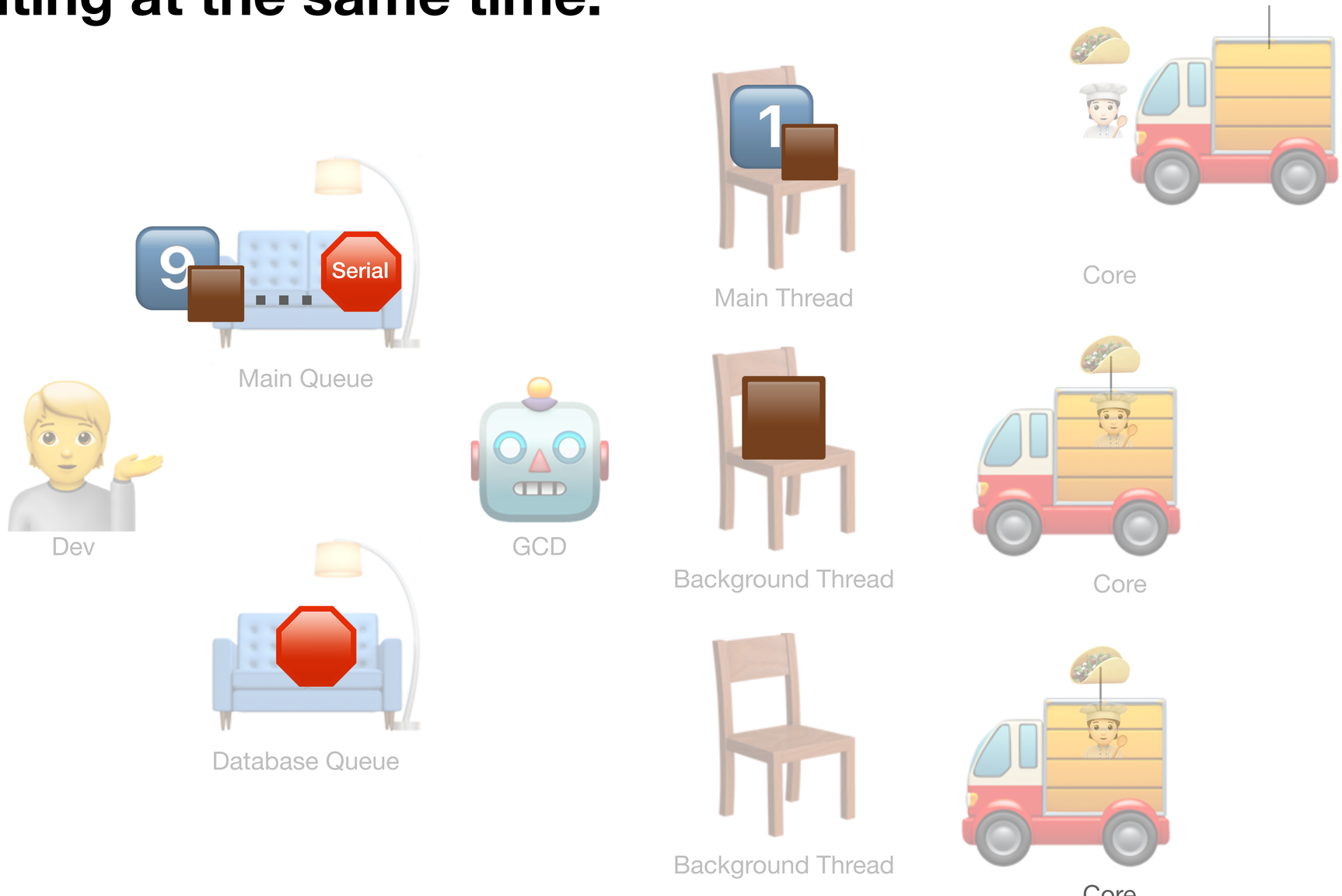


Parallel queues

Two queues executing at the same time.

Code

```
0  load entry index = 0
1  load entry index = 1
   ...
9  load entry index = 9
   update UI
```



Parallel queues

Two queues executing at the same time.

Code

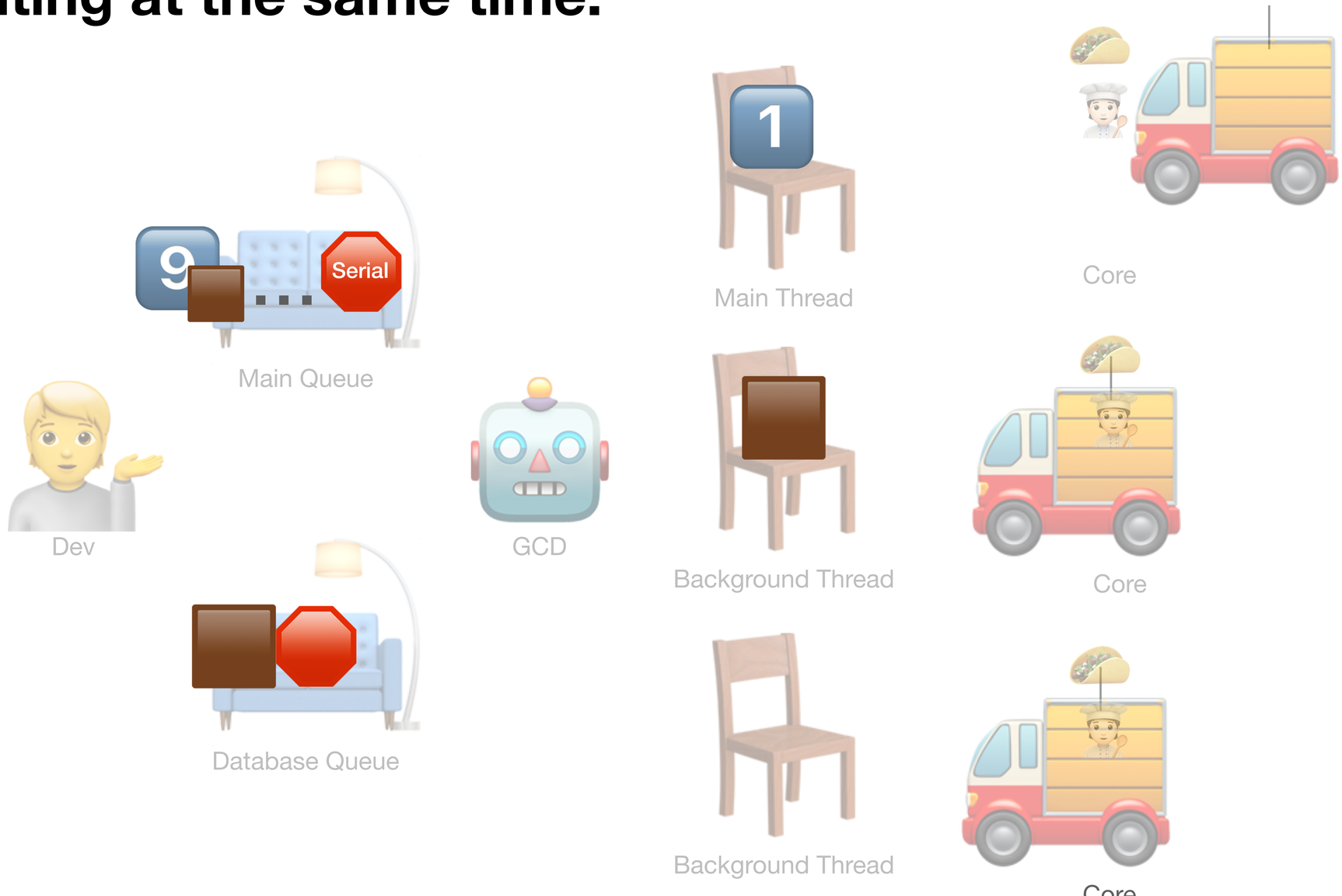
0 `load entry index = 0`

1 `load entry index = 1`

...

9 `load entry index = 9`

`update UI`



Parallel queues

Two queues executing at the same time.

Code

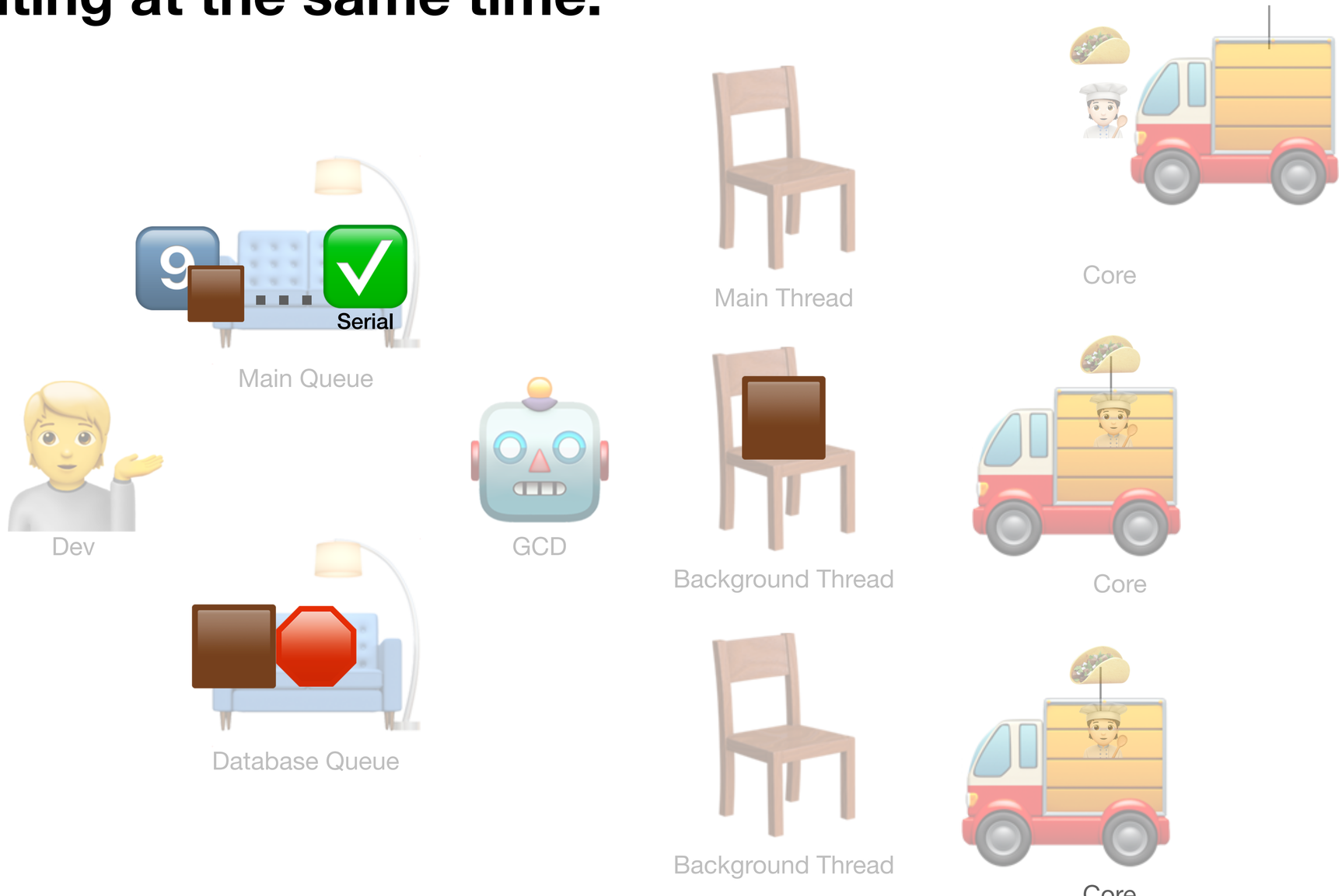
0 `load entry index = 0`

1 `load entry index = 1`

...

9 `load entry index = 9`

`update UI`



Parallel queues

Two queues executing at the same time.

Code

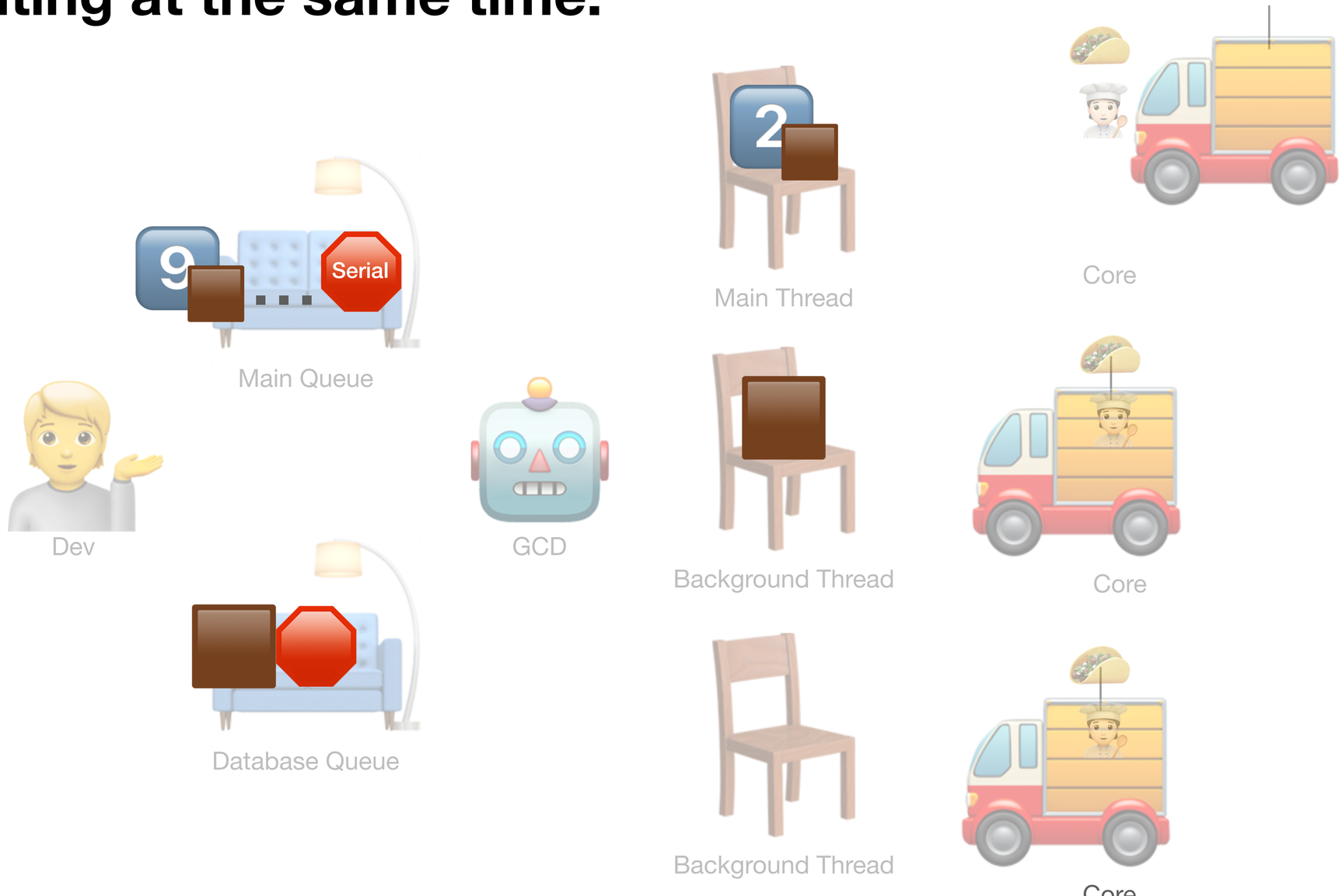
0 `load entry index = 0`

1 `load entry index = 1`

...

9 `load entry index = 9`

`update UI`



Parallel queues

Two queues executing at the same time.

Code

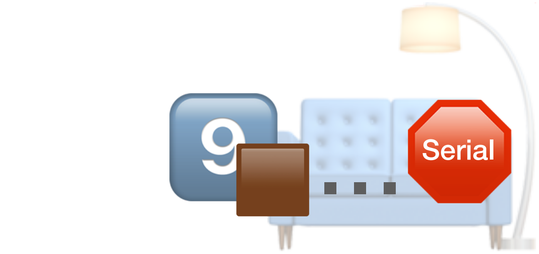
0 `load entry index = 0`

1 `load entry index = 1`

...

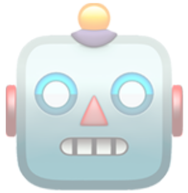
9 `load entry index = 9`

`update UI`

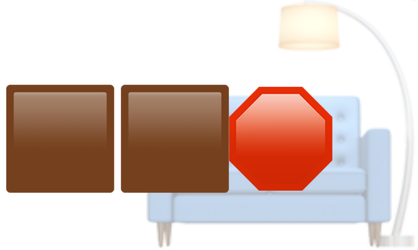


Main Queue

Dev

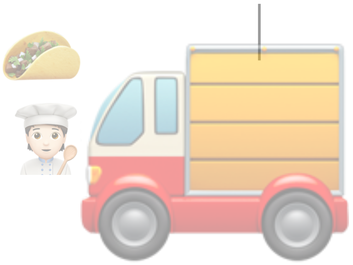


GCD



Database Queue

Main Thread



Core

Background Thread

Core

Background Thread

Core

Parallel queues

Two queues executing at the same time.

Code

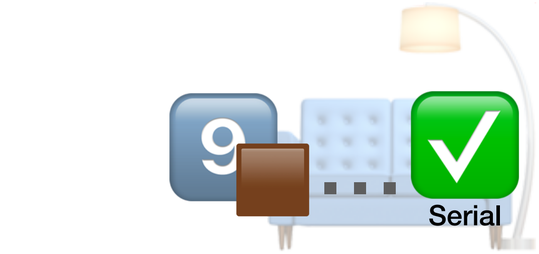
0 `load entry index = 0`

1 `load entry index = 1`

...

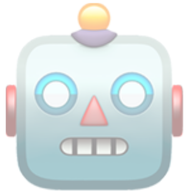
9 `load entry index = 9`

`update UI`

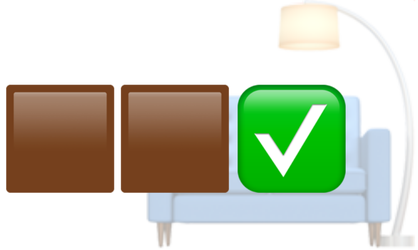


Main Queue

Dev

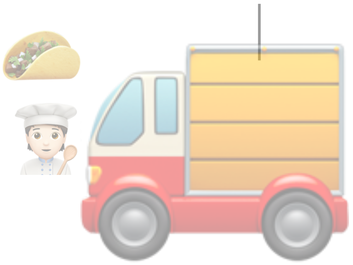


GCD



Database Queue

Main Thread



Core

Background Thread

Core

Background Thread

Core

Parallel queues

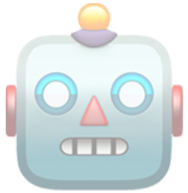
Two queues executing at the same time.

Code

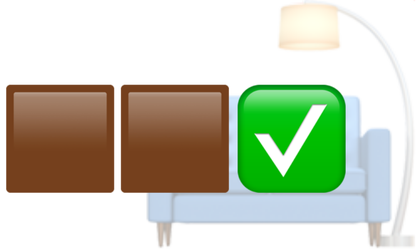
```
0  load entry index = 0
1  load entry index = 1
...
9  load entry index = 9
🖼  update UI
```

Main Queue

Dev

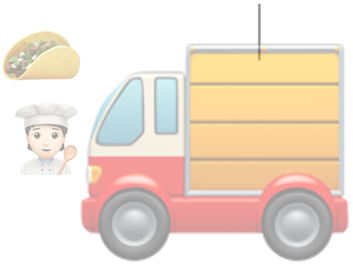


GCD



Database Queue

Main Thread



Core

Background Thread

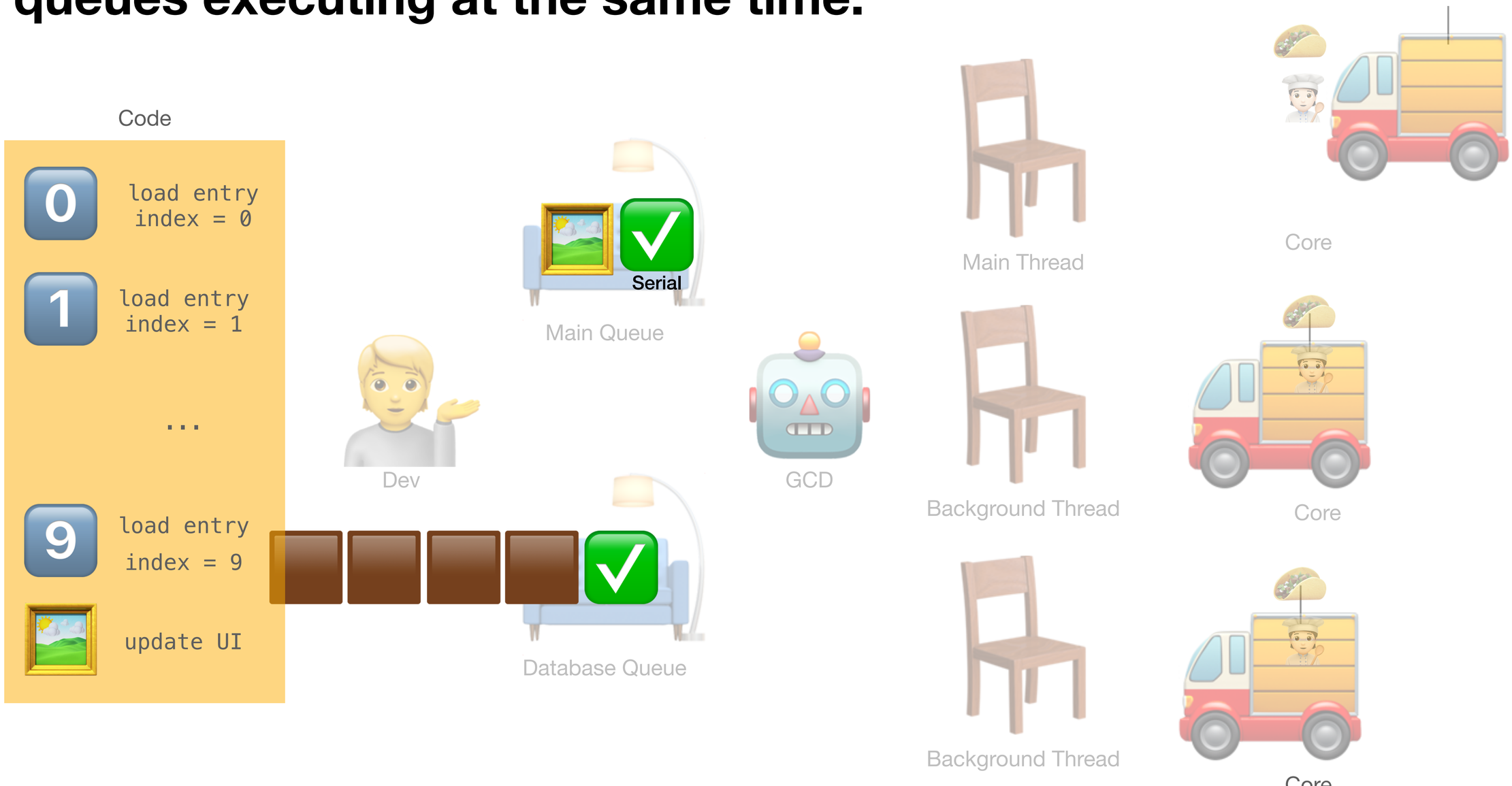
Core

Background Thread

Core

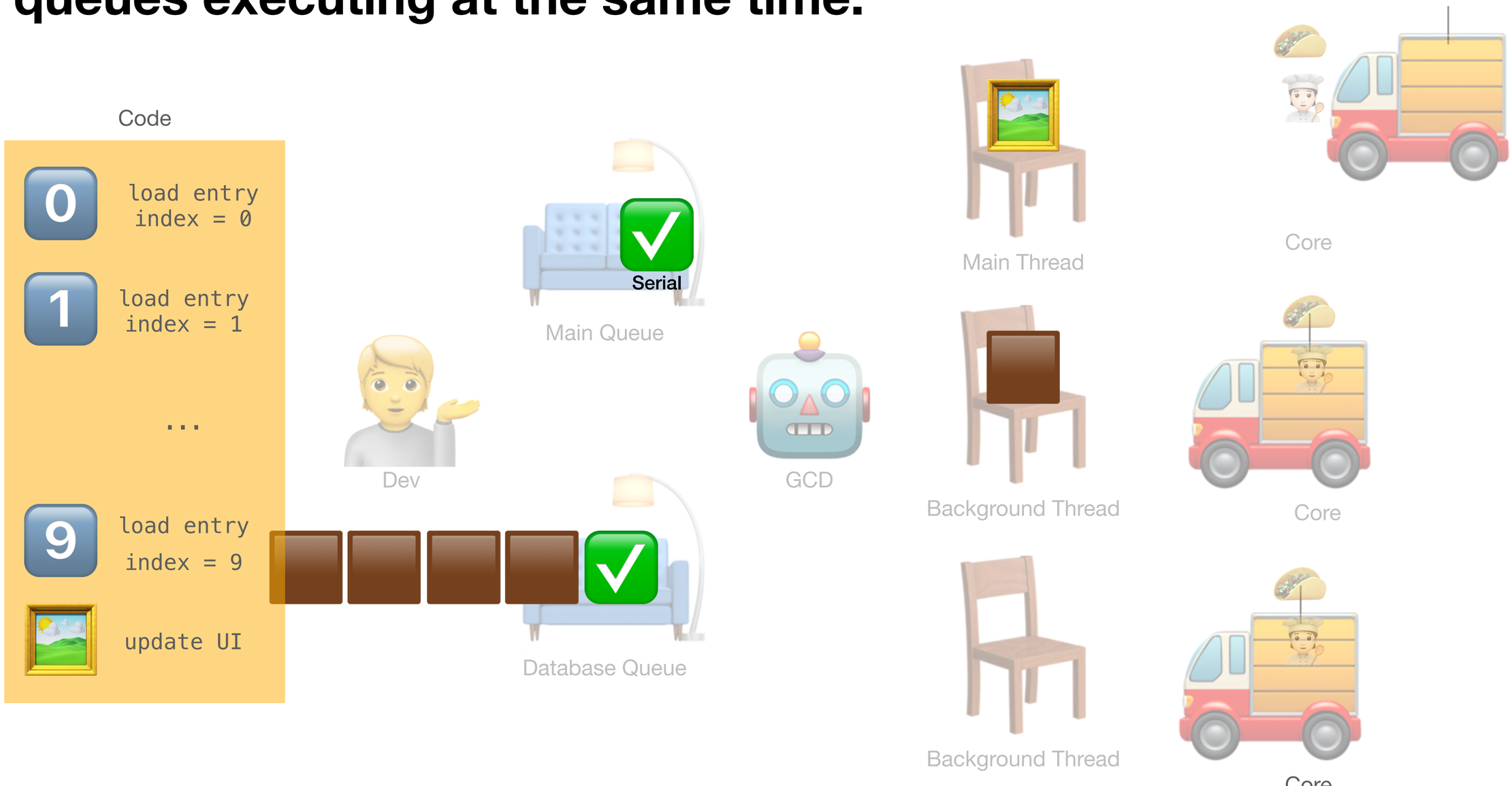
Parallel queues

Two queues executing at the same time.



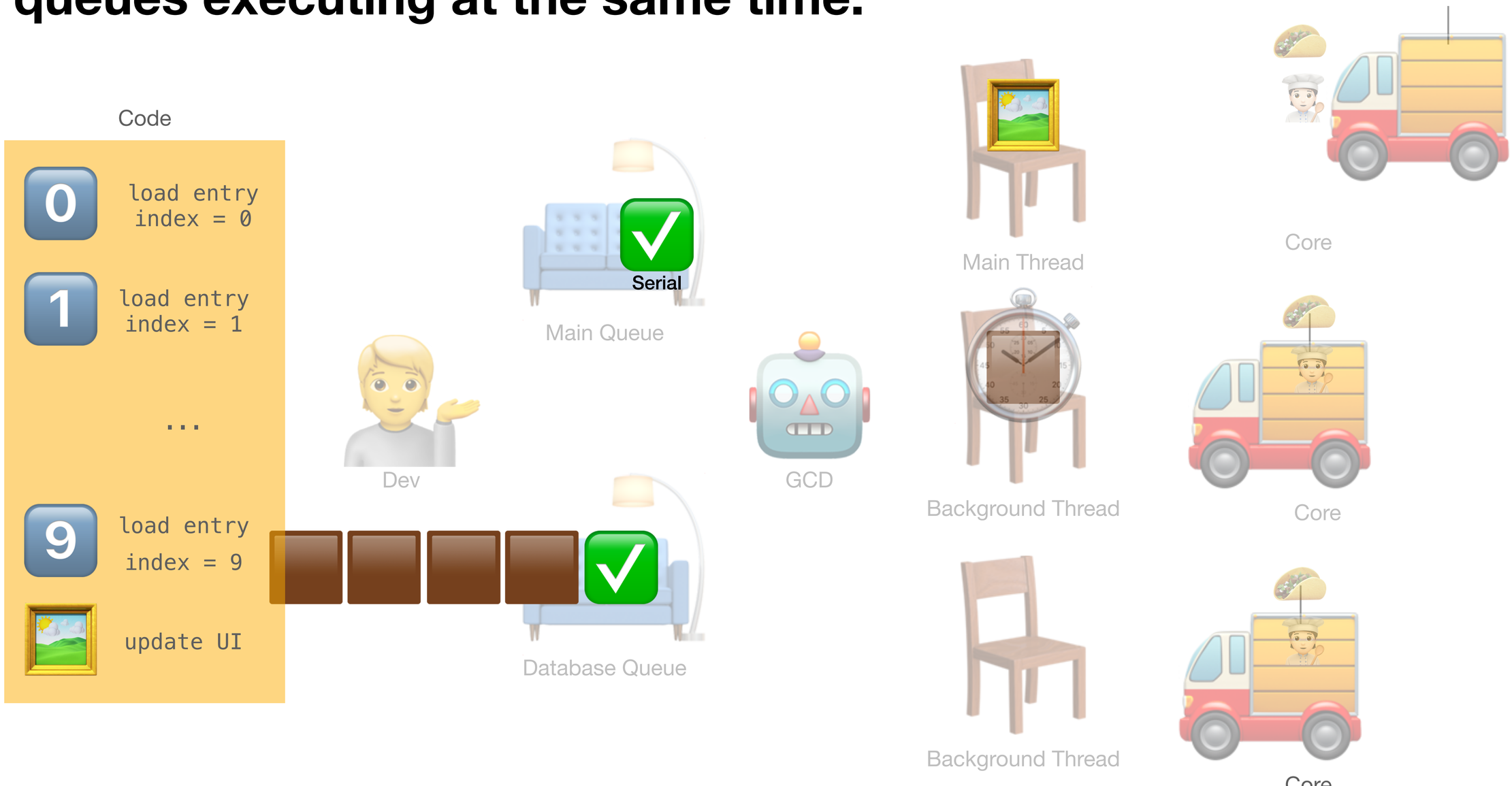
Parallel queues

Two queues executing at the same time.



Parallel queues

Two queues executing at the same time.



Quiz

Think of the code as it exists right now.

- Is the app executing tasks in parallel?
- Is the database queue serial?
- Is the algorithm for loading files concurrent?

Quiz

Think of the code as it exists right now.

- Is the app executing tasks in parallel?
  - ✅ Yes, the app is doing the following two tasks at the exact same time:
    - Updating the label
    - Loading files

Quiz

Think of the code as it exists right now.

- Is the database queue serial?
  - ✅ Yes, because the results are being completed in sequential order.
    - No matter how many times we load the files, they will always be loaded in increasing order based on their index (i.e. file 0 before file 1 before file 2...)

Quiz

Think of the code as it exists right now.

- Is the algorithm for loading files concurrent?
  - 🚫 No, because it cannot be completed in a different order.
    - If we are on a serial queue, we cannot be concurrent.

What is concurrency?

In theory, concurrency is having a smooth UI.

- Official definition: "concurrency is the ability of different parts or units of a program, algorithm, or problem to be executed out-of-order or in partial order, **without affecting the final outcome**"[1]
- Concurrency may be achieved using threads, which are "the smallest sequence of programmed instructions that can be managed independently by a scheduler".
- Darwin (core OS) is a threaded platform. Threading allows us to execute long-running tasks without blocking execution of other tasks.

[1] - https://en.wikipedia.org/wiki/Concurrency_(computer_science)

What is concurrency?

In practice, concurrency is introducing bugs without freezing the UI.

- Concurrency is very easy when you have tasks that are completely independent from each other (e.g. adding all numbers from 1-1000 in 2 threads).
- Concurrency cannot be achieved when tasks are dependent between each other (e.g. update one thread with info from another thread).
- Parallelism + Concurrency introduce new bugs:
  - Race conditions, deadlocking, livelocking, zombielocking, starvation, non-deterministic bugs.
- Parallelism should always be your last resort.

Concurrency and Parallelism

All queues are FIFO, but adding blocks depends on sync/async.

- Sync will stop the *current* queue (🛑 2️⃣3️⃣) from adding blocks to the target queue.
- Async will allow the *current* queue (✅ 2️⃣3️⃣) to continue after adding blocks to the target queue.

Concurrency and Parallelism

All queues are FIFO, but next block execution depends on serial/concurrent.

- Serial queues will stop the *current* queue (🛑 2️⃣3️⃣) execution until the active block finishes.
- Concurrent queues will stop the *current* queue (✅ 2️⃣3️⃣) execution until the active block begins.

What is GCD?

In practice, it's easi-*er* concurrency

- Understanding concurrency is still a prerequisite. GCD will only help avoid the most common scenarios.
- GCD provides practical APIs that help you handle queues, not threads. You are not guaranteed a specific thread for your closure, *except for the main thread*.
- The bugs are still there: deadlocking, livelocking, resource starvation...
- Stay as far away from GCD as possible. We've known it since 1974 (and maybe earlier): "Premature optimization is the root of all evil (or at least most of it) in programming." - Donald Knuth

GCD and more

Where to go from here?

- Will be uploaded to https://github.com/olivaresf/GCD/
- You can reach me @fromJrToSr
- Additional info:
  - https://theswiftdev.com/ultimate-grand-central-dispatch-tutorial-in-swift/
  - https://www.raywenderlich.com/5370-grand-central-dispatch-tutorial-for-swift-4-part-1-2

Support Fernando

Classes like these are expensive.

- **Practice Swift weekly with a 15-minute exercise.**
  https://mailchi.mp/hey/weekly-swift-exercise-signup

- **Donations are welcome and include the recorded session.**
  https://paypal.me/fromjuniortosenior

- **Mock Interview: iOS Networking is a 1-hour exercise** based on my experience as a team lead, as well as the many interviews I've aced throughout the years.
https://gum.co/wbyeU

Supporting me helps me keep classes free.

Next Class

Intro to Unit Testing

Q&A