Prerequisite

Asynchronous Code Primer

What is Asynchronous Code?

- Asynchronous (async) is a programming method that allows code execution to be paused
- While paused, other async code can run
    - Paused code can be resumed from where it left off and continue executing
    - Pausing is usually done when waiting on external resources
        - Network
        - Database

Async vs Threads

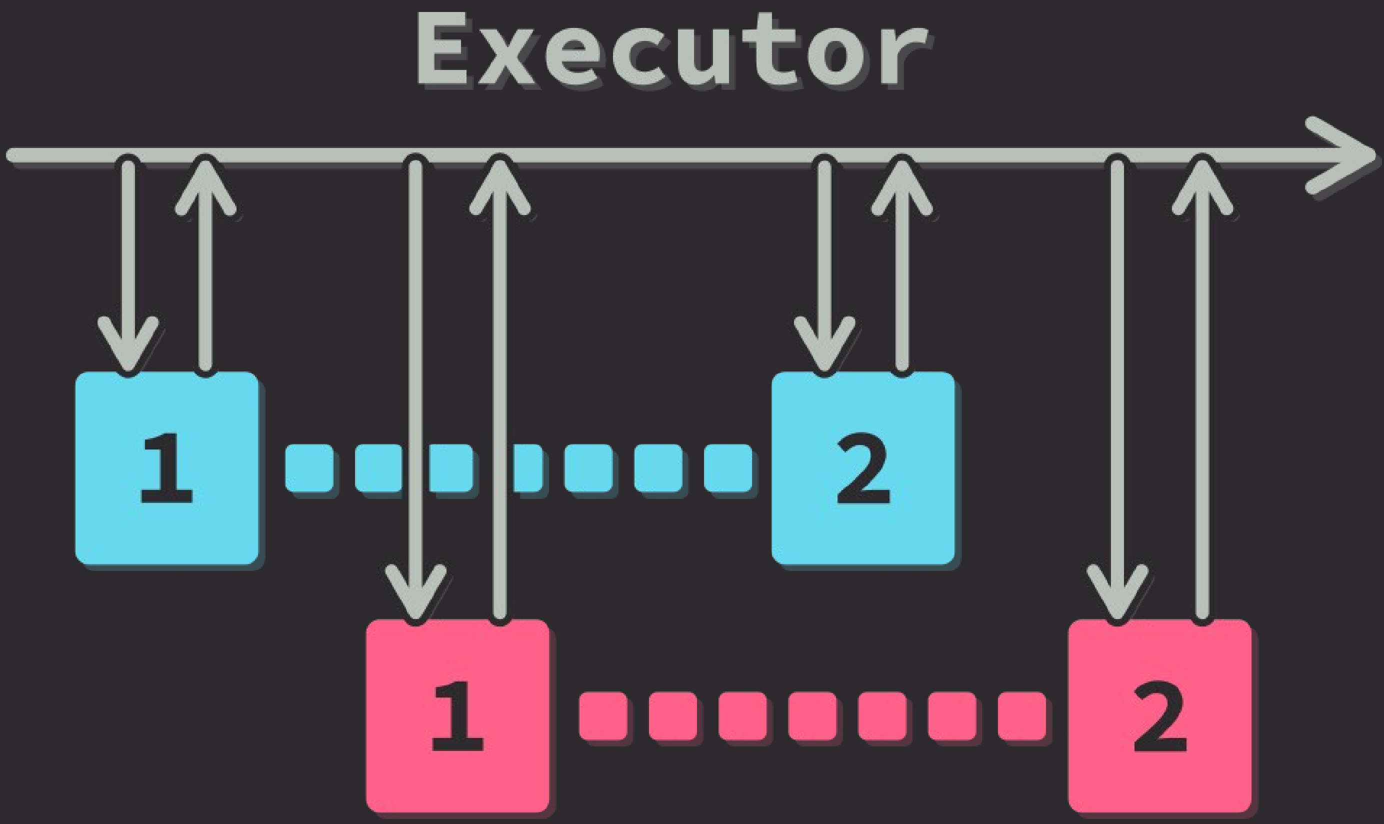
- Async code only ever executes serially
    - Threads can truly execute in parallel
- Async code can wait on a large amount of data sources
    - Threads can only wait on one data source

- Use async when there are many external data sources
- Use threads when there is heavy computation

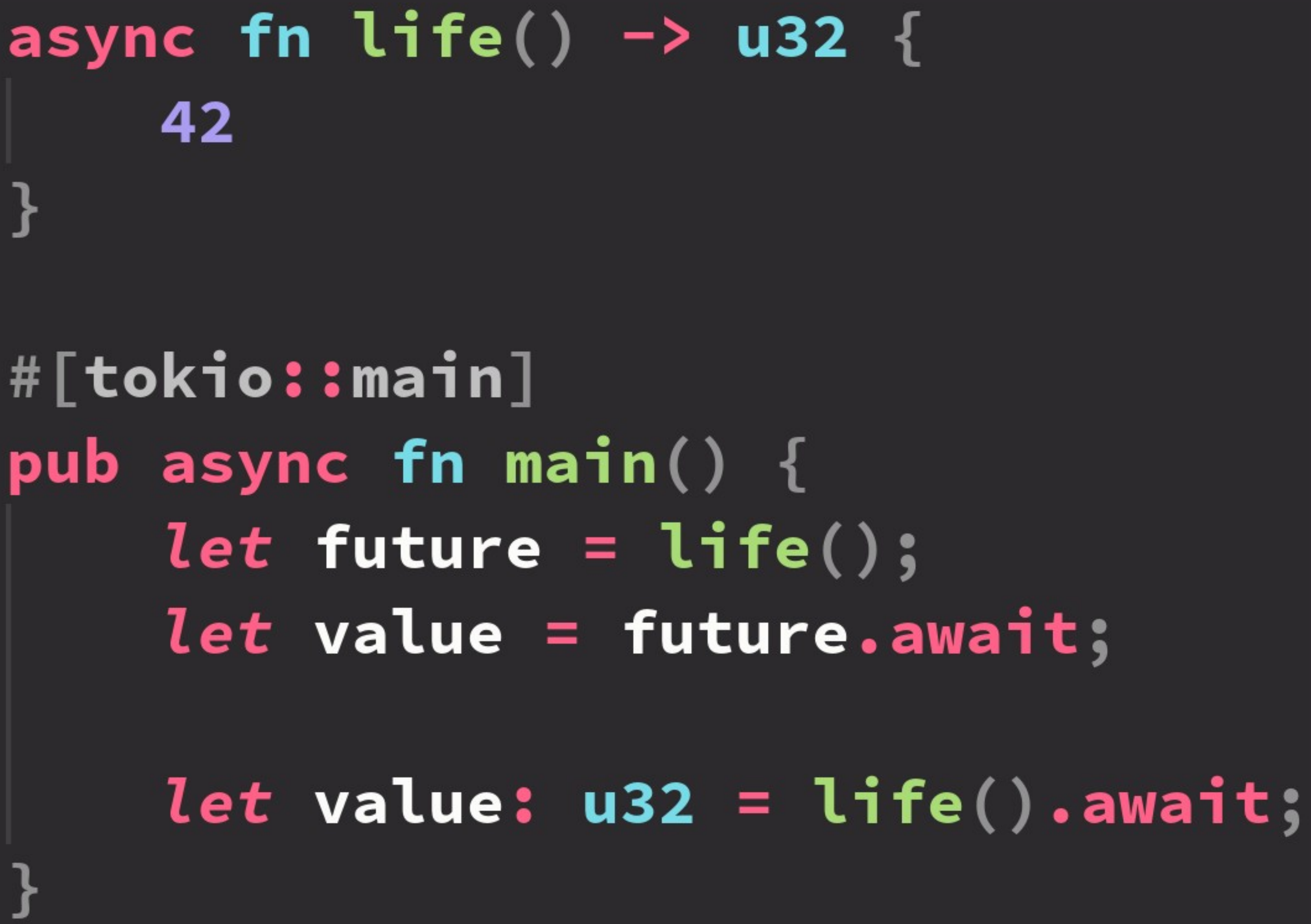
Futures

- Container type that encapsulates code to be executed at a future time
- No code within the ***Future*** is initially run
  - Lazy execution
- ***Futures*** are ran on an ***Executor***
  - Execution of a ***Future*** can be paused by using ***.await***
    - The ***Executor*** will then run other ***Futures*** until they complete or also ***.await***

Driving a Future



Async Functions



```rust
async fn life() -> u32 {
    42
}

#[tokio::main]
pub async fn main() {
    let future = life();
    let value = future.await;

    let value: u32 = life().await;
}
```

Example

```rust
async fn connect() -> Result<Connection, ConnectionError> {
    Ok(Connection)
}

async fn get_credentials(conn: &Connection)
    -> Result<Credentials, CredentialError>
{
    Ok(Credentials)
}

async fn generate_session(conn: &Connection, creds: &Credentials)
    -> Result<Session, SessionError>
{
    Ok(Session)
}
```

Example

```rust
#[tokio::main]
pub async fn main() -> Result<(), ApiError> {
    let conn = connect().await?;
    let creds = get_credentials(&conn).await?;
    let session = generate_session(&conn, &creds).await?;
    Ok(())
}
```

Recap

- Asynchronous code can be paused and resumed
  - Great when waiting on external data sources
- Async code is ran by an ***Executor***
  - Executors can be created using a macro on the ***main*** function
- The ***async*** keyword is used to create an asynchronous function
- Use ***.await*** to pause execution while waiting for external data