Swingset/Docs/Async.md

Asynchronous Considerations in SwingSet

When working with SwingSet, it's crucial to understand how asynchronous operations interact with the system's execution model. A **vat** (a unit of isolated execution in SwingSet) may restart or terminate before an asynchronous operation completes. Although JavaScript's `await` keyword allows you to write asynchronous code that resembles synchronous code, you must be mindful of events that could prevent a promise from settling:

- **Underlying condition never satisfied**: The condition required for the promise to settle may never occur.
- **Vat terminates, restarts, or upgrades before settlement**: The promise will be severed if its decider vat terminates, restarts, or upgrades before it settles.

Promise Temporality

When programming with promises in SwingSet, it's important to understand the **temporality** of promise settlement—that is, the timing and execution dependencies required for a promise to settle. Promises can be classified into three categories based on their temporality: **Immediate**, **Prompt**, and **Delayed**.

Immediate Promises

An **Immediate** promise is one that settles within the same Crank as its creation (see Kernel Cycles). We refer to this as "Immediate" because it occurs before any further message deliveries into the vat, similar to how all promise reaction callbacks happen before any JavaScript event callbacks (such as those queued by `setImmediate`).

Characteristics of Immediate Promises

- **Settlement Timing**: Settles within the same crank it was created.
- **Dependencies**: Requires no additional input to the "vat"; all information is locally available.
- **Safety**: Not affected by vat restarts or upgrades, which occur in their own crank.

Prompt Promises

A **Prompt** promise settles without requiring further input from outside SwingSet. All Immediate promises are also Prompt, but not all Prompt promises are Immediate.

The SwingSet kernel processes its run queue items until there is nothing left to run. Most hosts, like cosmic-swingset, do not inject new items on the SwingSet run queue until it's empty, and instead maintain their own queue of input events. In those cases the SwingSet kernel executes similarly to how a vat executes a crank: an external I/O event triggers some execution, and the resulting SwingSet run queue items are drained before returning to the host for the next I/O.

While any of these queue items can technically cause a vat to upgrade, most systems running on SwingSet will trigger a vat upgrade based on some I/O input. As such, Prompt promises are unaffected by vat upgrades, unless the vat getting upgraded somehow got involved in processing the I/O that triggered its own upgrade.

Characteristics of Prompt Promises

- **Settlement Timing**: Settles before any new I/O is processed.
- **Dependencies**: May involve multiple cranks but does not require external input.
- **Safety**: Safe from being severed by vat upgrades initiated by external I/O.

Factors That Can Poison Promptness

The following are examples of I/O that can prevent a promise from being Prompt (they "poison" promptness):

- **Waiting on a timer**
- **Waiting on an inter-chain network call**
- **Waiting on a bid being placed**
- **Waiting on a new oracle price**

Factors That Do Not Affect Promptness

- **Upgrade**: cosmic-swingset ensures quiescence between I/O and vat upgrades are triggered only when processing a new item from the chain's action queue (such as network input).
- **Bridge Calls**: Bridge calls are Prompt. While the caller may subsequently wait for an acknowledgment input (which is not Prompt), the bridge call itself does not affect promptness.

Delayed Promises

A **Delayed** promise is one that requires additional input or external events to settle. It does not settle within the same crank and depends on factors not available during the current run of SwingSet. Delayed promises are neither Immediate nor Prompt because they involve waiting for external dependencies or future events beyond the current execution cycles.

Characteristics of Delayed Promises

- **Settlement Timing**: Settles in future cranks after external events or inputs occur.
- **Dependencies**: Requires external input or events outside of SwingSet to settle.
- **Safety**: More susceptible to vat restarts, upgrades, or terminations affecting their settlement.

Common Scenarios for Delayed Promises

- **Waiting on a Timer**: Introduces a delay until a specified time, relying on external scheduling.
- **Waiting on External Data or Events**: Depends on input from outside the SwingSet environment, such as a new oracle price.
- **User Actions**: Requires actions from external users, like placing a bid.
- **Network Communications**: Involves waiting for responses from external systems or networks, such as inter-chain network calls.

Contrast with Immediate and Prompt Promises

- **Immediate Promises**:
  - **Settlement**: Within the same crank.
  - **Dependencies**: No external dependencies; all information is available within the "vat".
  - **Example**: A calculation based on existing in-vat data.
- **Prompt Promises**:
  - **Settlement**: Before any new I/O occurs, possibly over multiple cranks.
  - **Dependencies**: No external input required, but may involve inter-vat asynchronous operations.
  - **Example**: Internal message passing between vats that settles before any new external events.
- **Delayed Promises**:
  - **Settlement**: After external events or inputs are received, over multiple cranks.
  - **Dependencies**: Requires external input, events, or time to pass.
  - **Example**: Waiting for a user to place a bid or for an oracle to provide new data.

Summary

Understanding the temporality of promises in SwingSet helps you write robust asynchronous code that behaves predictably, even in the face of vat restarts, upgrades, or terminations. By being aware of the differences between Immediate, Prompt, and Delayed promises, you can design your smart contracts and applications to be more resilient and efficient.

- **Immediate Promises** are settled within the same crank and are safe from vat restarts or upgrades during that crank.
- **Prompt Promises** settle without external input before any new I/O, making them safe from being severed by vat upgrades initiated by external events.
- **Delayed Promises** depend on external input or events and settle over future cranks, making them more vulnerable to disruptions like vat restarts or terminations.

By classifying your promises appropriately, you can better manage asynchronous operations and ensure the reliability of your SwingSet applications.