CAB202 Asynchronous Programming with Async Await

A survey on its usage in c# and javascript and the paradigms related

Anhad Ahuja

June 7, 2022

1 Introduction

Given no context for the underlying physcical system behind a dataset makes allows a analsys for raw data especially for deriving correlations instead of causations. However, if given some context there is sure to be false causality drawn from statements presented in this report. This report will outline some very detailed statistics about the data given and attempt to produce consumable information for potential analysis.

2 How async works under the hodo in c#

2.1 State Machine implementation

2.2 SynchronizationContext

The context that a piece of code gets run on is

```
public HomeViewModel()
  {
      SelectedIndex = 0;
      Games = new ObservableCollection<Game>(Db.games);
      DispatcherTimer timer = new DispatcherTimer();
  }
```

3 Previous paradigm used and history of introduction

4 Using lazy evaluation of enumerables: `IAsyncEnumarable`

5 UI based example and how to avoid system locking

Locks up Ui, conventionaly heavy computationaly or IO based operation that locks up UI thread is undesirable. It could be possible to create a new thread using system API and delegate tasks to that, but that has a lot of computational and programmer overhead. Generally not a trivial matter especially to sync up content and data.

6 `Task.Run` based example

7 Additional quirks and useful features of Task based await

7.1 Cancellation tokens

7.2 Progress

8 Comparison to javascript

9 Javascript similiar example using `Promise`

References