Web Programming

Week 11

"vita brevis, ars longa."

Hippocrates

Agenda

Coordination of asynchronous actions

Scheduler and DataFlow abstraction

Coordination schemata

similar to concurrency

1) No coordination needed
2) Sequence (of side effects)
3) Dependency on former results

No Coordination

=> nothing to do

Execution model: confined

All actions run independently.

Sequence

Actor *Flux Architecture, Redux, ViewX etc.*

In a sequence of actions, each action can only start if the preceding one has finished.

How to achieve this?

Delegated Coordination => Scheduler

Result Dependency

Actions B and C need the result of action A. A must be executed **exactly once** before B and C.

How to do this?

Implicit Coordination => DataFlowVariable

Reminder

Promise

```
const processEven = i => new Promise( (resolve, reject) => {
    if (i % 2 === 0) {
        resolve(i);
    } else {
        reject(i);
    }
  }
);
processEven(4).then( num => console.log(num));
```

success/failure callbacks

Scheduler Idea

Queue (FIFO) of functions
that are started with a lock.

Callback unlocks.

DataFlowVariable

Function, that sets a value if it is not already set. Returns the value.

Lazy: access to variables that will become available later.

Trick: do not set the value, but a function that returns the value.

Let's code 1

Asynchronous Todo Fortune Service

Double-Click Protection,

Lazy Loading, Sequence guarantee

[Let's code]

Excel with

DataFlowVariable

Home / Code Kitchen

Try to re-do the excel solution.