

Web Programming

Week 9

"Frameworks and APIs change fast. Software design principles are evergreen. Learn principles that translate across language barriers."

Eric Elliot

Retrospective

Quiz

Homework

Retro: Test "FW"

"Framework" = 2 Abstraktionen: *public API*

test(name, callback)

Assert

More will come later...

Callback - HOF

Higher-order Function

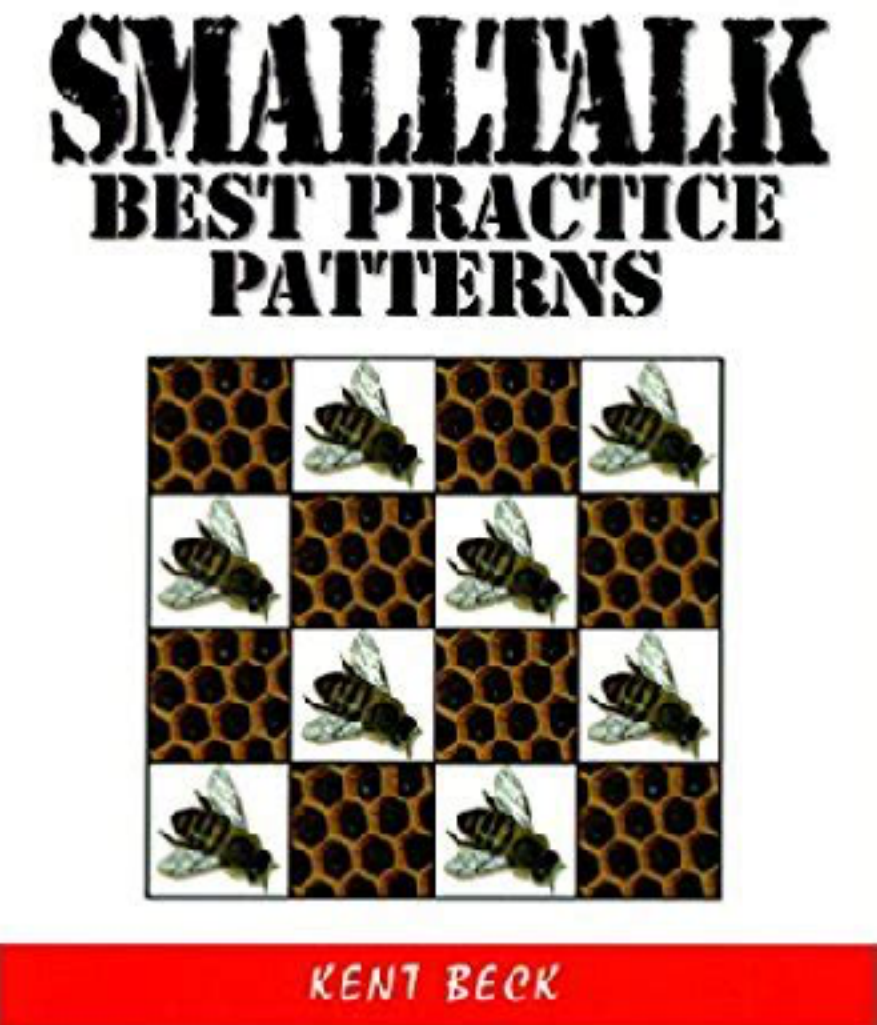
```
function test(name, callback) {  
  const assert = Assert();  
  callback(assert);  
  report(name, assert.getOk());  
}
```

prework

callback

postwork

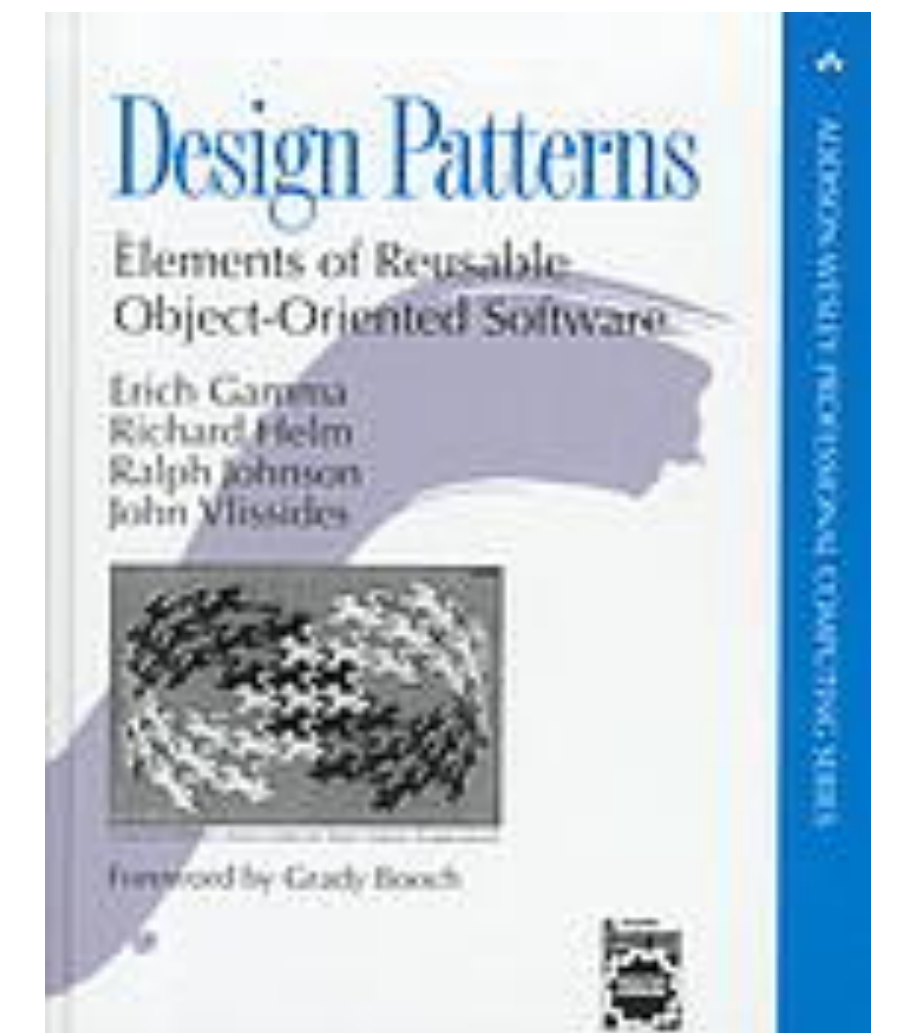
Patterns



Execute Around Method

<http://web.cecs.pdx.edu/~black/OOP/slides/Best%20Practices.pdf>

Template Method, Method Object



Ressource Handling

Extracted - Abstracted

Protocol: open, use, close.

Files, DB Queries, Transactions,
load URLs, call REST services, error
containment,
CPU-time, Threads (Pool), UI-Access, ...

Today: UI "Framework"

FW usage requires FW insight:
what it does,
how it does it,
why it does it this way.

" I could build this myself! "

"Hello, World!" of UI FWs

Todo List



× UI FW verstehen



× verbessern



× anwenden



Tasks: 3

Open: 2

Start "explore"

Move 1: start at the end

Tests are red

The simplest solution that could
possibly workTM

Milestone 0

It runs

Reasonable amount of tests

All tests ok

Red-Green-Refactor

Move: Reorganisation

Separate the un-essentials

Improve clarity (= abstraction)

Improve clarity

Responsibilities: "What" vs "How"

Dependencies: Who knows whom?

Sequence: What happens when?

Clear and simple rules.

Observable (v.1.0)

```
const Observable = value => {  
  const listeners = [];  
  return {  
    onChange: callback => listeners.push(callback),  
    getValue: () => value,  
    setValue: val => {  
      if (value === val) return;  
      value = val;  
      listeners.forEach(notify => notify(val));  
    }  
  }  
};
```

many

protection

ordering

Observable Topics++

Observer modifies Observable

"Bindstorm"

Less obvious: $A \rightarrow B \rightarrow C \rightarrow A$

One or more Observers?

Callback at registration? Remove?

Milestone 2

Separate Todo operations
from Todo display

All tests ok

Improve Clarity

A) single Todo changes

B) the List of Todos changes

=> observable list

Milestone 3

View, Controller, and Model
are separated

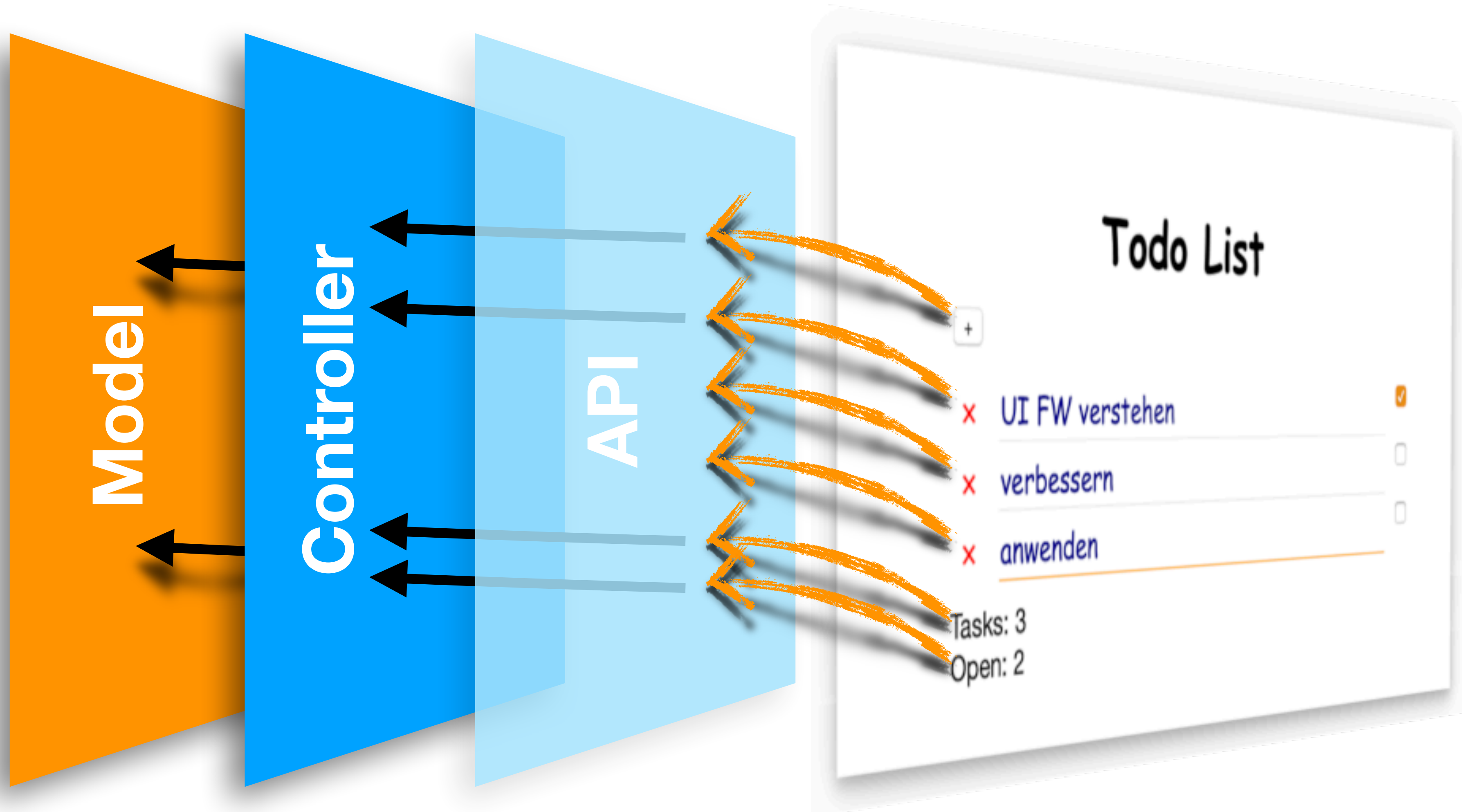
All tests ok

Milestone 4

All View
separated

All tests ok

MVC, classic version



Possible Extensions

- 👍 Observer Pattern: Observable Values
 - 👍 Observable Collections (Array, List, Set)
- Binding, Validation, Conversion
- Asynchronous Data Flow (e.g. REST)

Work at Home

Can you spot the memory-leaks in the Observable List?

Can you test it?

Can you resolve it?