Web Programming

Week 10

"Don't call us. We call you."

Hollywood

Retrospective

Quiz

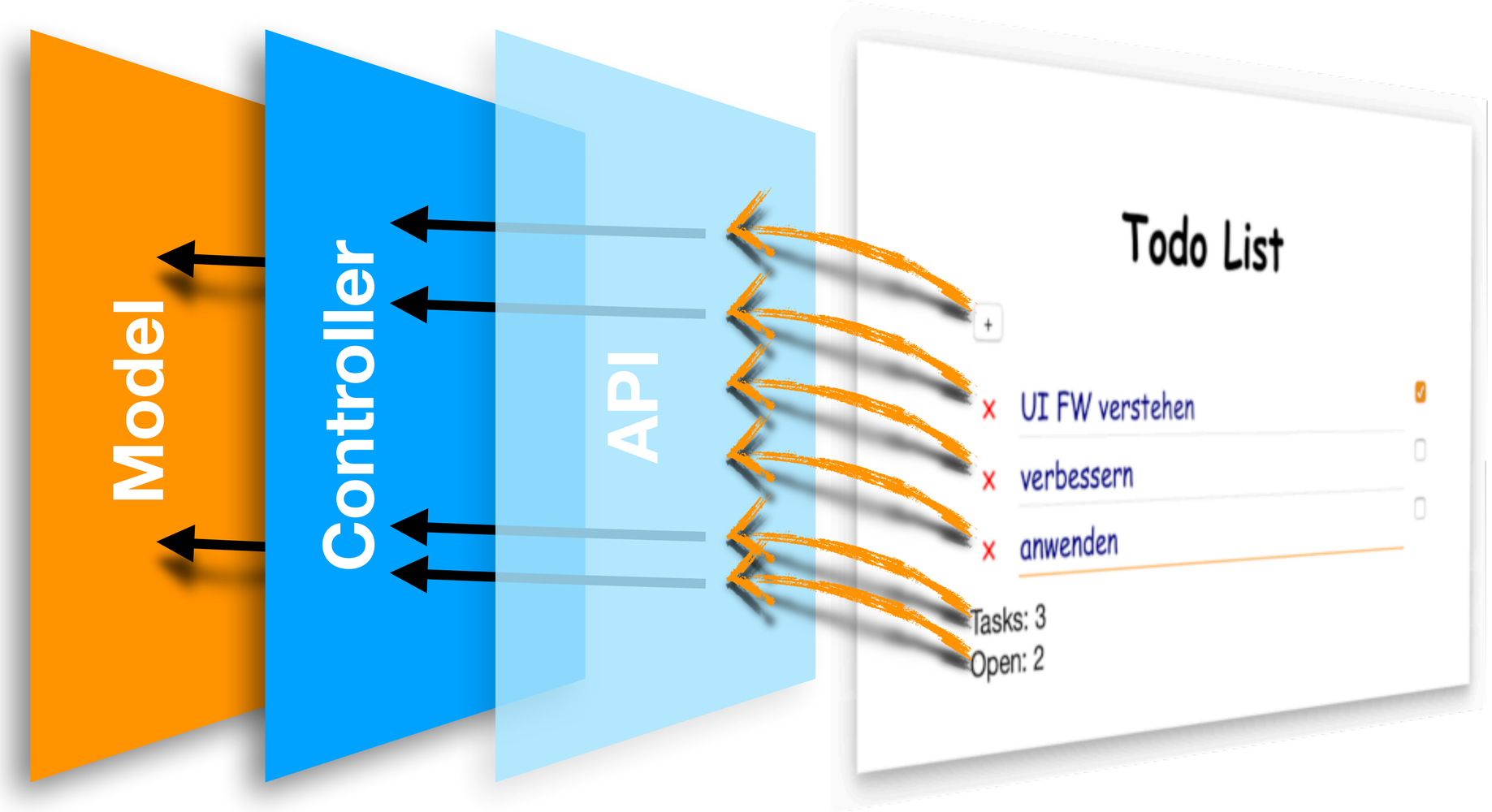
Homework

Agenda

Homework: memory leak
find, test, resolve

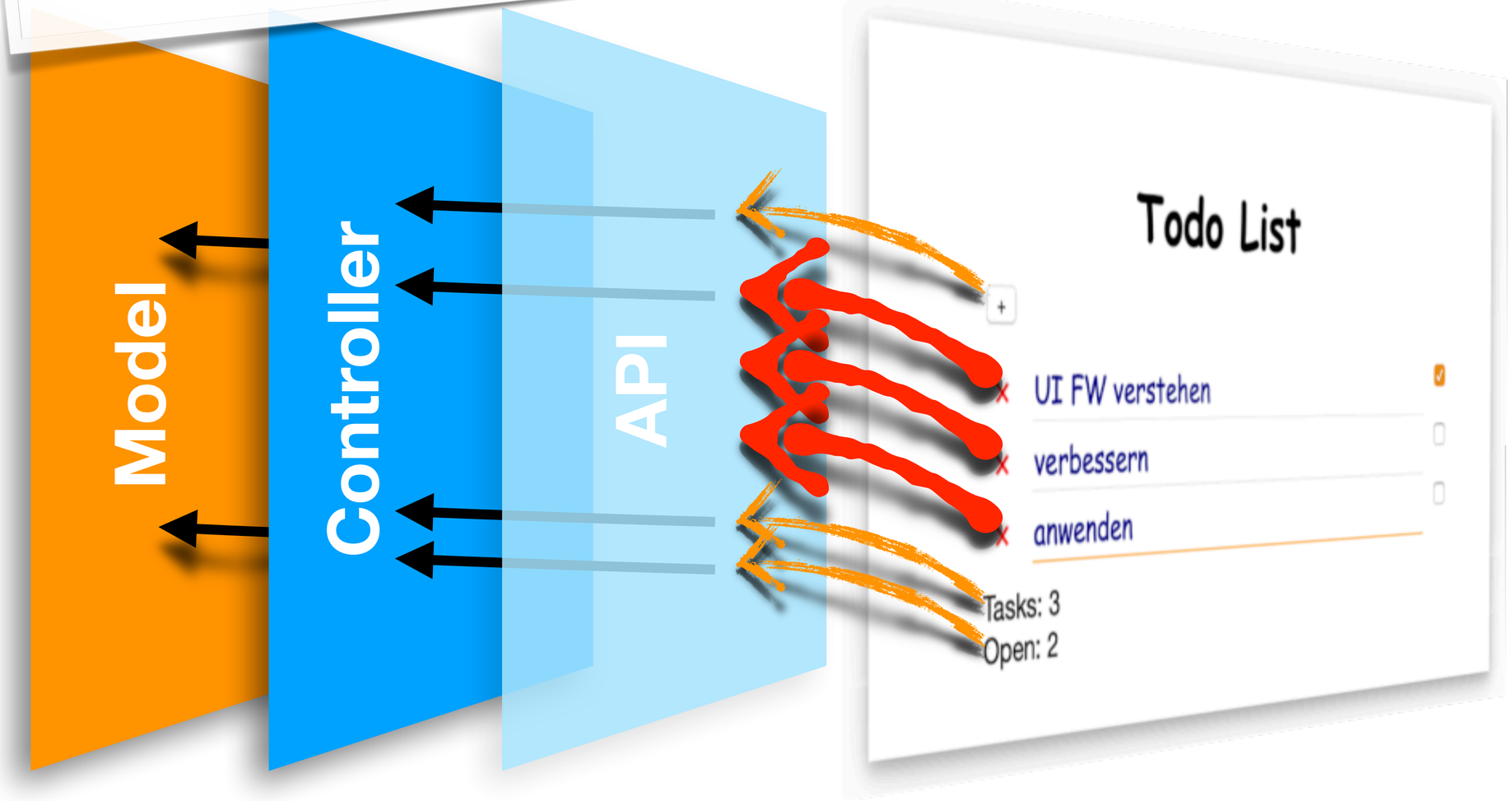
Introduction asynchronous calls

MVC, classic version



MVC, classic version

Remove Listeners are never removed



Testing

```
test("todo-memory-leak", assert => {
    const todoController = TodoController();

    todoController.onTodoAdd(todo => {
       todoController.onTodoRemove( todo => {

       });
    });

    for (let i=0; i<10000; i++){
        const todo = todoController.addTodo();
        todoController.removeTodo(todo);
    }
});
```

10'000 : 2s
20'000 : 8s
100'000 : ?s

Testing

```
test("todo-memory-leak", assert => {
    const todoController = TodoController();

    todoController.onTodoAdd(todo => {
        todoController.onTodoRemove( (todo, removeMe) => {
            removeMe();
        });
    });

    for (let i=0; i<10000; i++){
        const todo = todoController.addTodo();
        todoController.removeTodo(todo);
    }
});
```

idea: self remove

New Topic: Async

Eric Elliott @_ericelliott · 21h

Every JS app developer needs to know:
-FP basics
-objects: composition vs inheritance
-async patterns (callbacks, promises, events, streams)

14 181 674

Callback, Events

Snake, week 3

soon: DataFlowVars

```
function start() {
    //...
    window.onkeydown = evt => {
        // doSomething();
    };

    setInterval(() => {
        // doSomething();
    }, 1000 / 5);
}
```

Promise

Callback Hell

```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

Promise

"Thenable"

most prominent use

```
fetch ('http://fhnw.ch/json/students/list')
 .then(response => response.json())
 .then(students => console.log(students.length))
 .catch (err    => console.log(err)
```

"Functor"

"Monoid"

"Monad"

Promise

example definition

```
const processEven = i => new Promise( (resolve, reject) => {
    if (i % 2 === 0) {
        resolve(i);
    } else {
        reject(i);
    }
  }
);
```

success/failure callbacks

Promise

example use

promise

auto promotion

```
processEven(4)
    .then ( it => {console.log(it); return it} )
    .then ( it => processEven(it+1))
    .catch( err => console.log( "Error: " + err))
```

"synchron"

console.log(1)

console.log(2)

console.log(3)

1

2

3

"asynchron"

```
console.log(1)

new Promise( resolve =>
    ...; resolve(2)
).then( n => console.log(n) )

console.log(3)
```

1

2

3

"asynchron"

```
console.log(1)

new Promise( resolve =>
    ...; resolve(2)
).then( n => console.log(n) )

console.log(3)
```

1

2

3

"asynchron"

```
console.log(1)

new Promise( resolve =>
    ...; resolve(2)
).then( n => console.log(n) )

console.log(3)
```

```
1

3
```

"asynchron"

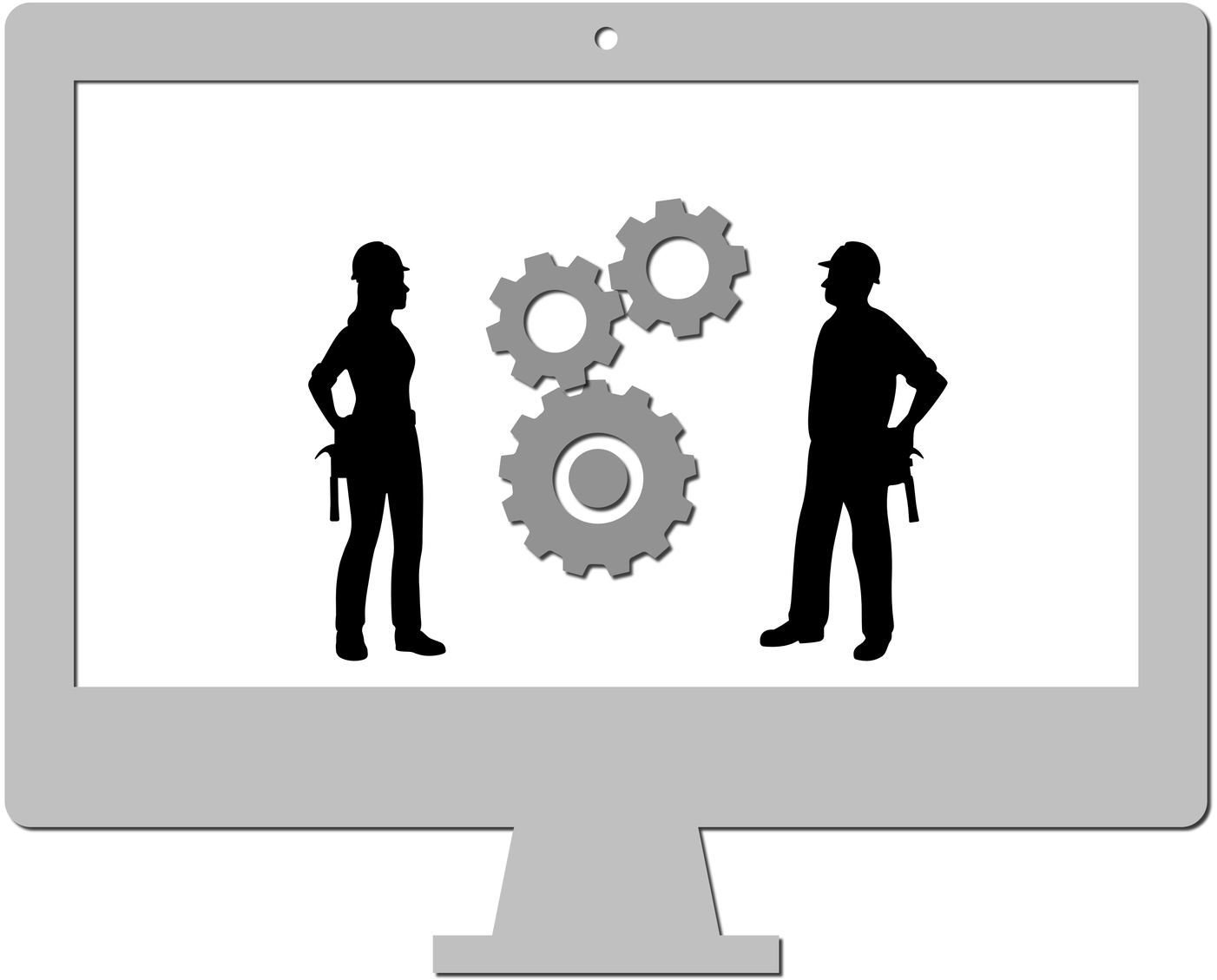
```
console.log(1)

new Promise( resolve =>
    ...; resolve(2)
).then( n => console.log(n) )

console.log(3)
```

1

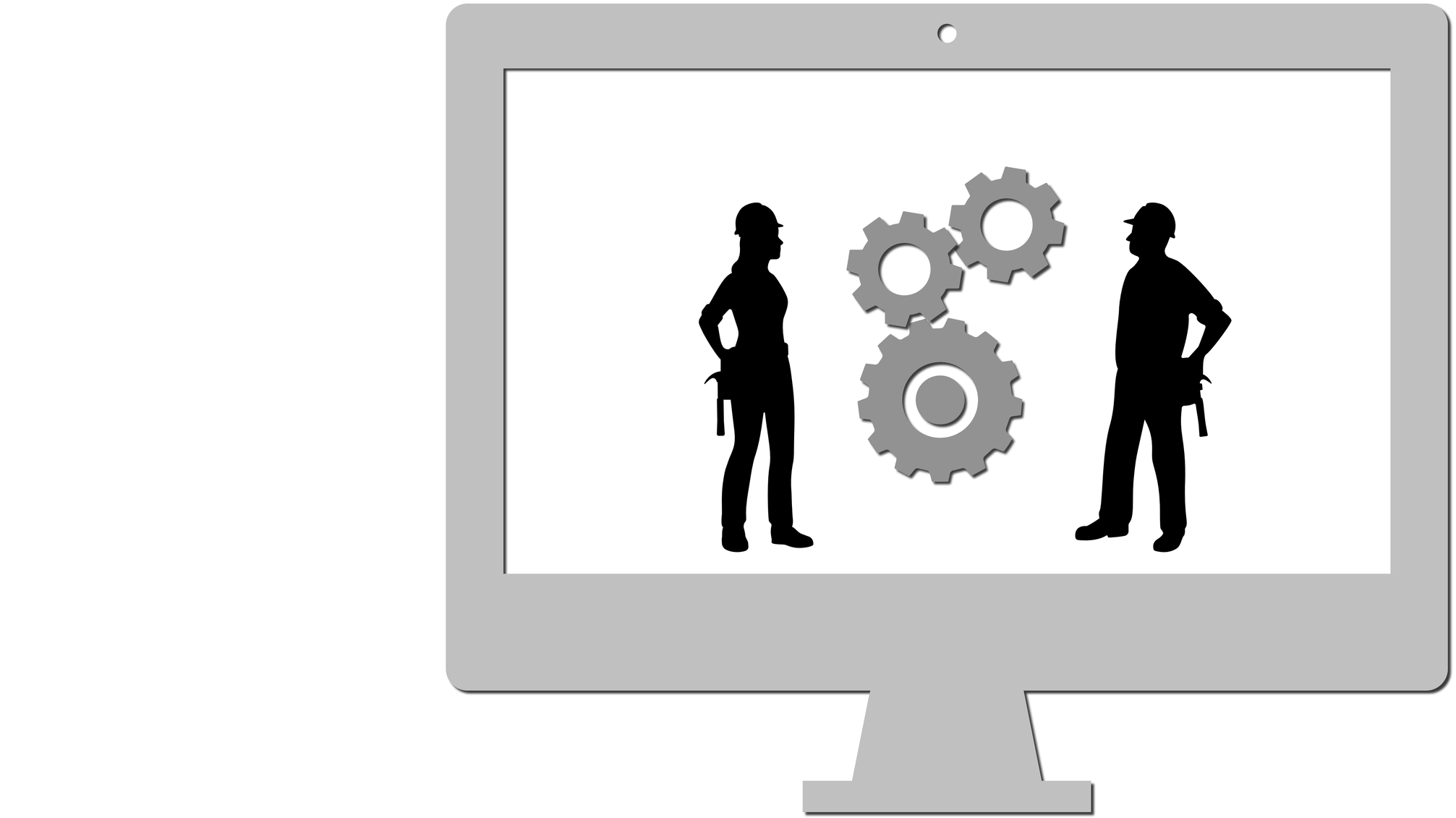
3

2

1, prom(2), 3

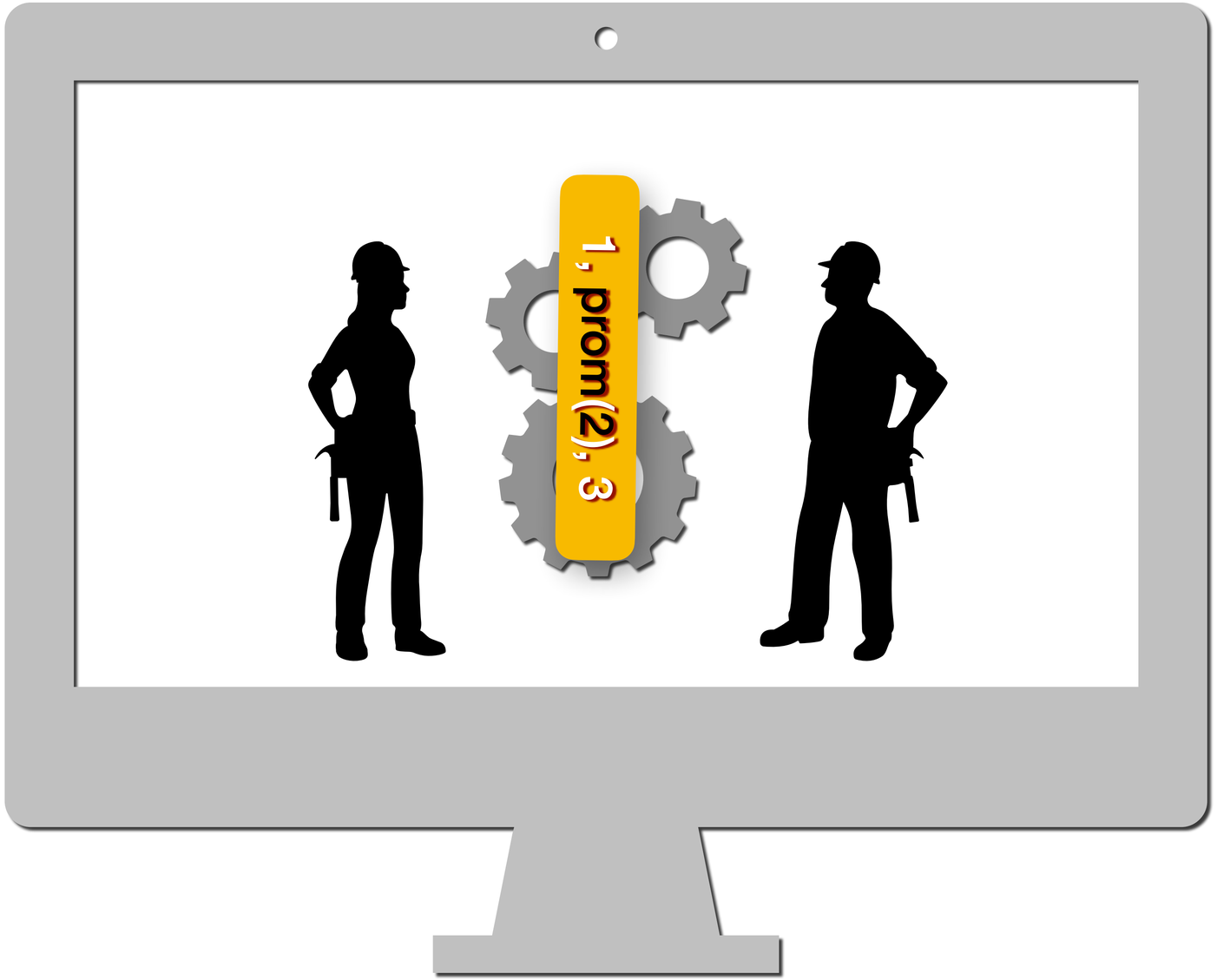


1, prom(2), 3

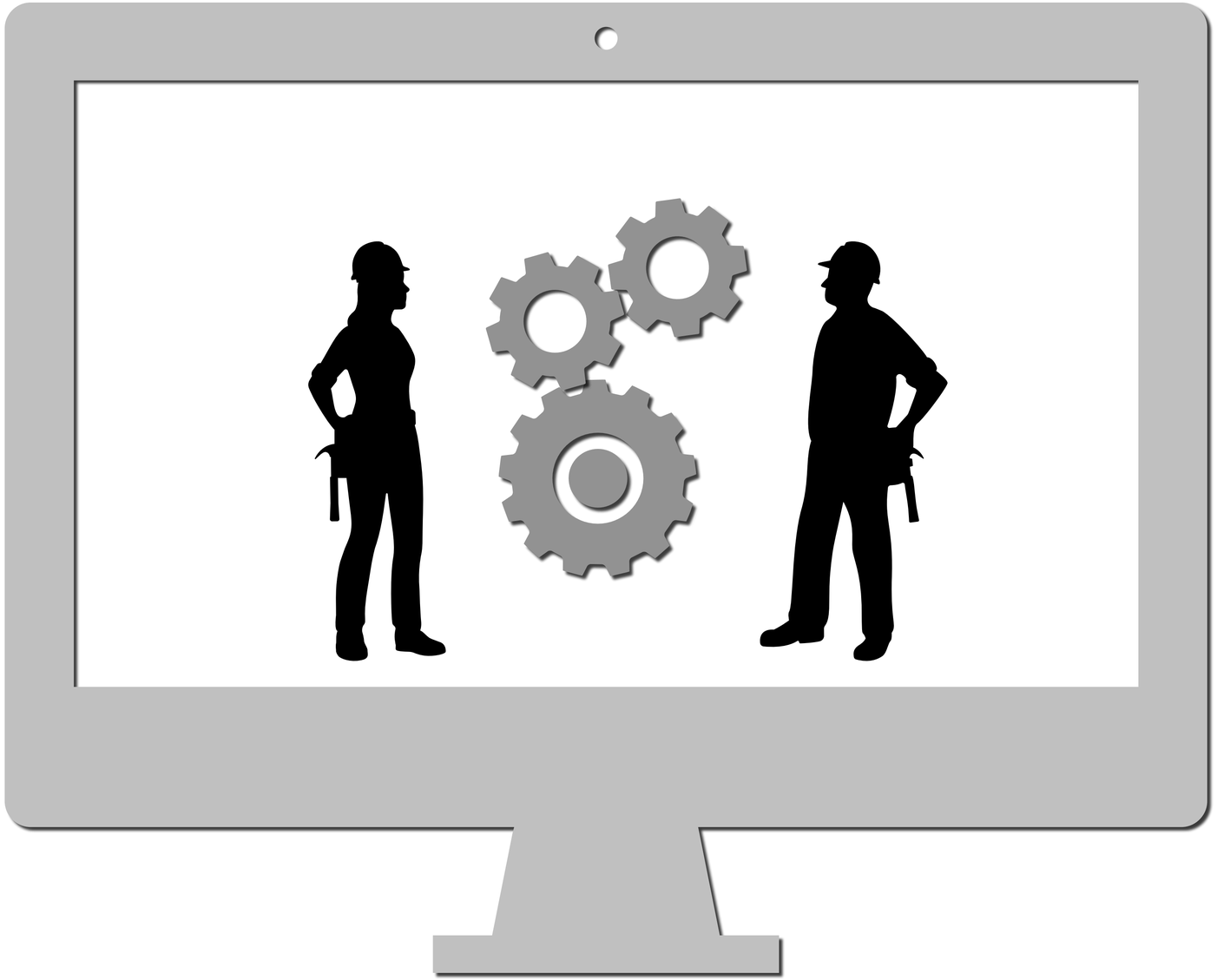


1, prom(2), 3

resolve(2)



resolve(2)



resolve(2)

async / await

```
const foo = async i => {
    const x = await processEven(i).catch( err => err);
    console.log("foo: " + x);
};
foo(4);
```

Alternative syntax for Promises
that reads more like imperative code

async / await variants

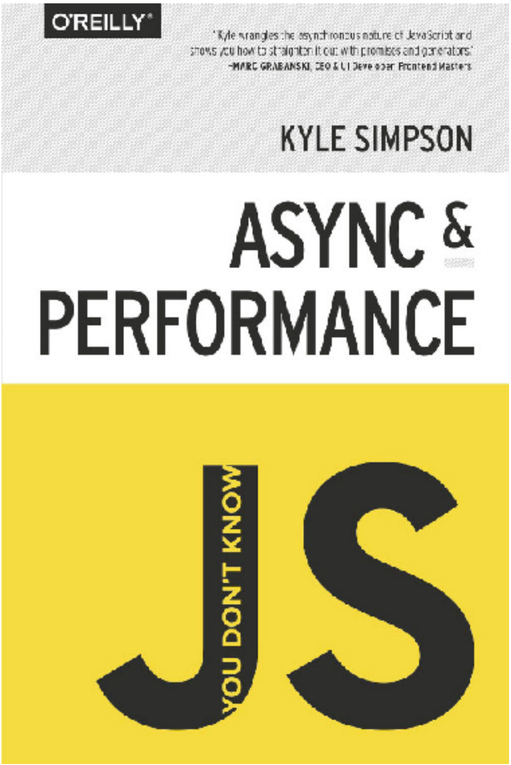
```
async function foo(i) {
    try {
      const x = await processEven(i);
      console.log("foo: " + x);
    }
    catch(err) { console.log(err); }
};
foo(4);
```

Not in Scope

cooperative concurrency with

*generator functions / yield

Work at Home



Chapters 1, 2, 3

Also good: Promise and Async/Await description in Mozilla Dev Network