Michael Meadows

Event-Driven UX in the Real World with Angular and Socket.io

Event-Driven UX in the Real World with Angular and Socket.io

Event driven UI programming using WebSockets is great! The problem is that every demo app is a chat app or a game. This session focuses on real-world examples using socket.io and Angular 2 to demonstrate how event-driven user interfaces can dramatically improve a user experience. Attendees will leave able to design practical event-driven applications that provide a richer level of interaction with a lower level of programming effort.

But First, some education…

Education...

Why Angular 2… no wait, 4.0

- Data Binding
- Event-Driven HTML
- Focus on Components
- Fast
- Active Community/Active Development

Why Event-Driven Server Code

- Already Doing It (AJAX)
- AJAX sucks (req/resp not suited)
- Better Data Security (only transmit what’s displayed)
- More Responsive
- Eventual Consistency
- CQRS!

Why socket.io

- Simple
- Stable
- Small
- Still Maintained

UX Patterns/Anti-Patterns

Anti-Pattern: Request/Response

REPORTS

- Unpredictable response time (depends on parameters)
- Most users don't need/want to wait
- "Save for Later" is More valuable than "Use Immediately"

WEB ACTIONS

- 2 seconds is *sooooo long*
- Chatty isn't always bad
- Only able to respond to events on browser… Server events are opaque.

Anti-Pattern: Polling

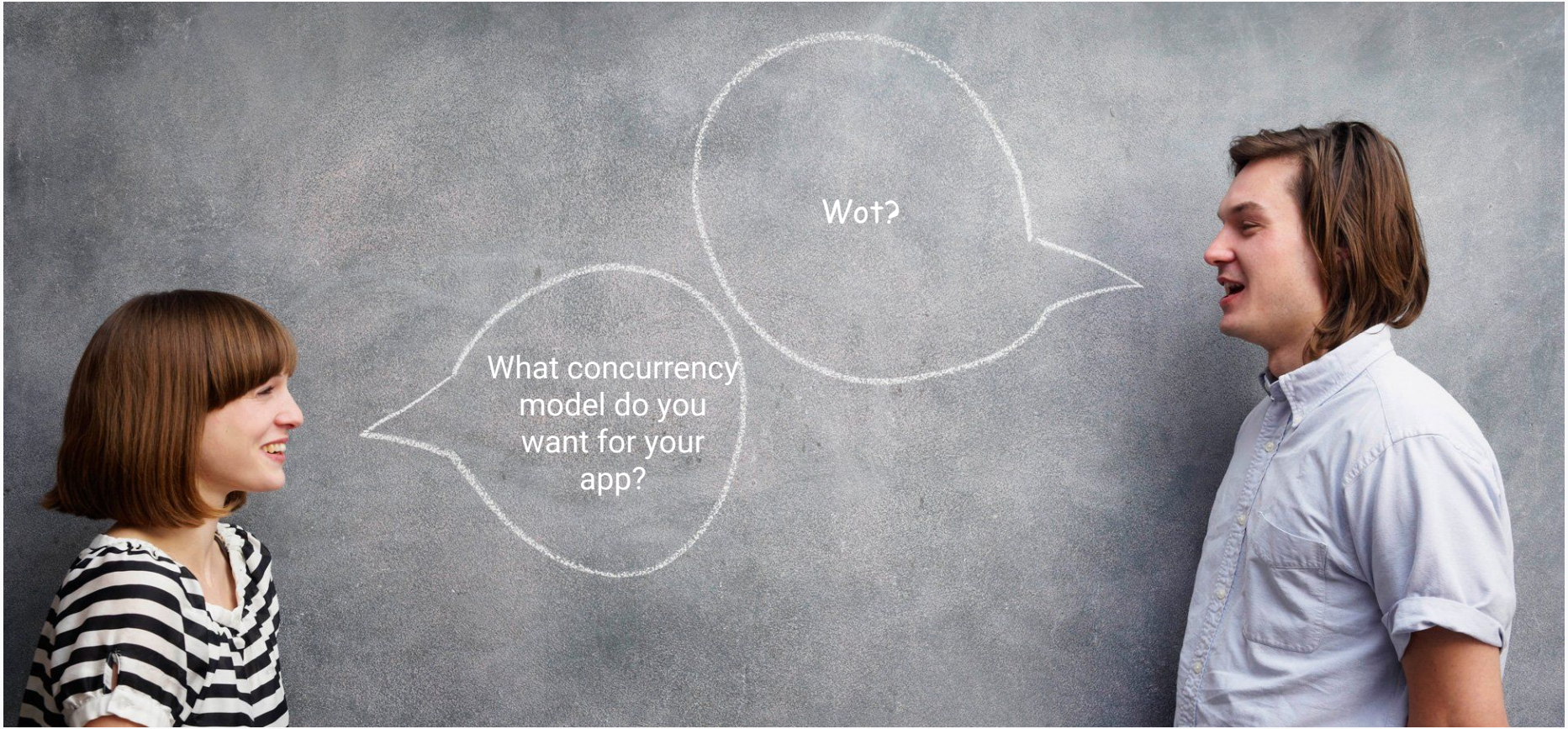
Srsly?

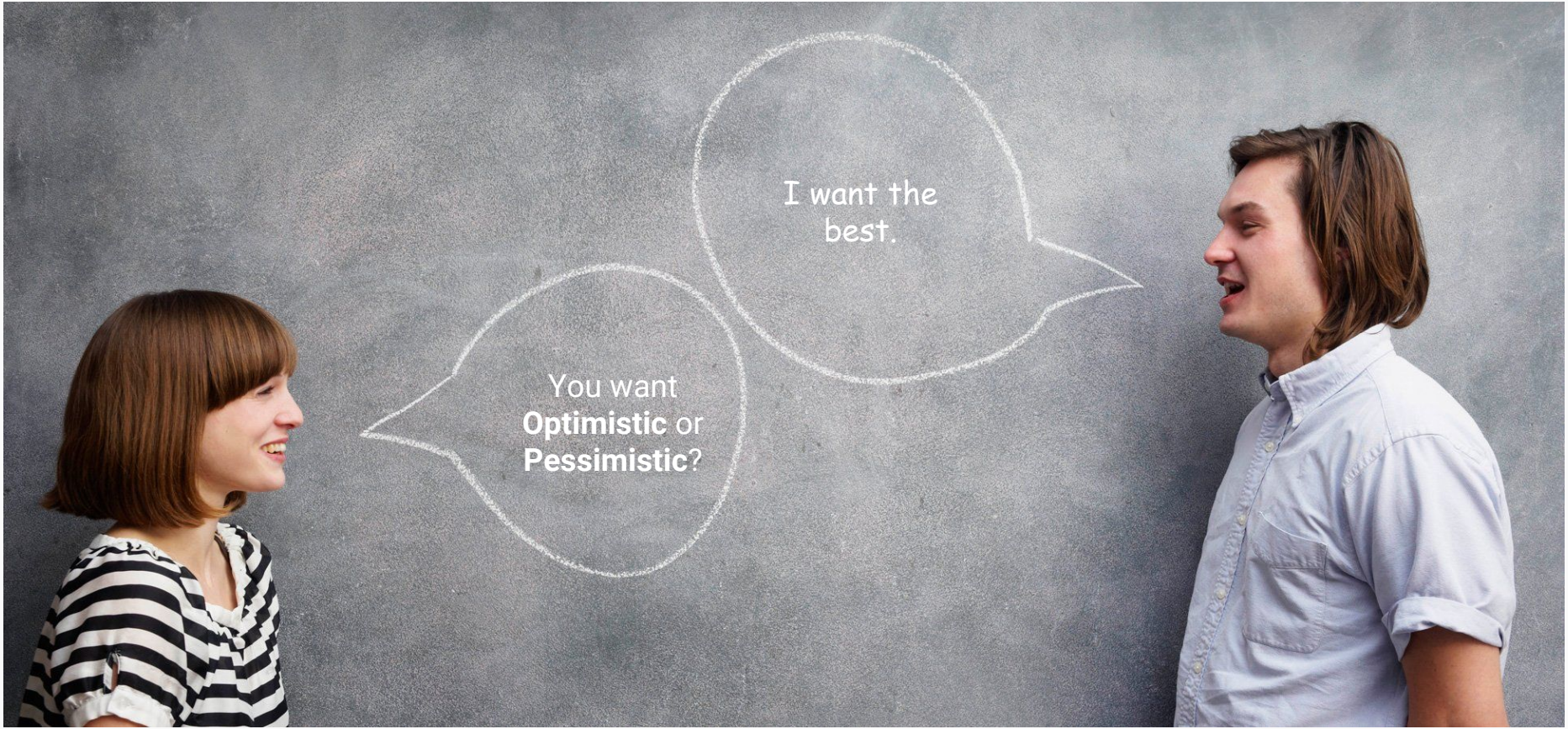
Anti-Pattern: The Save Button

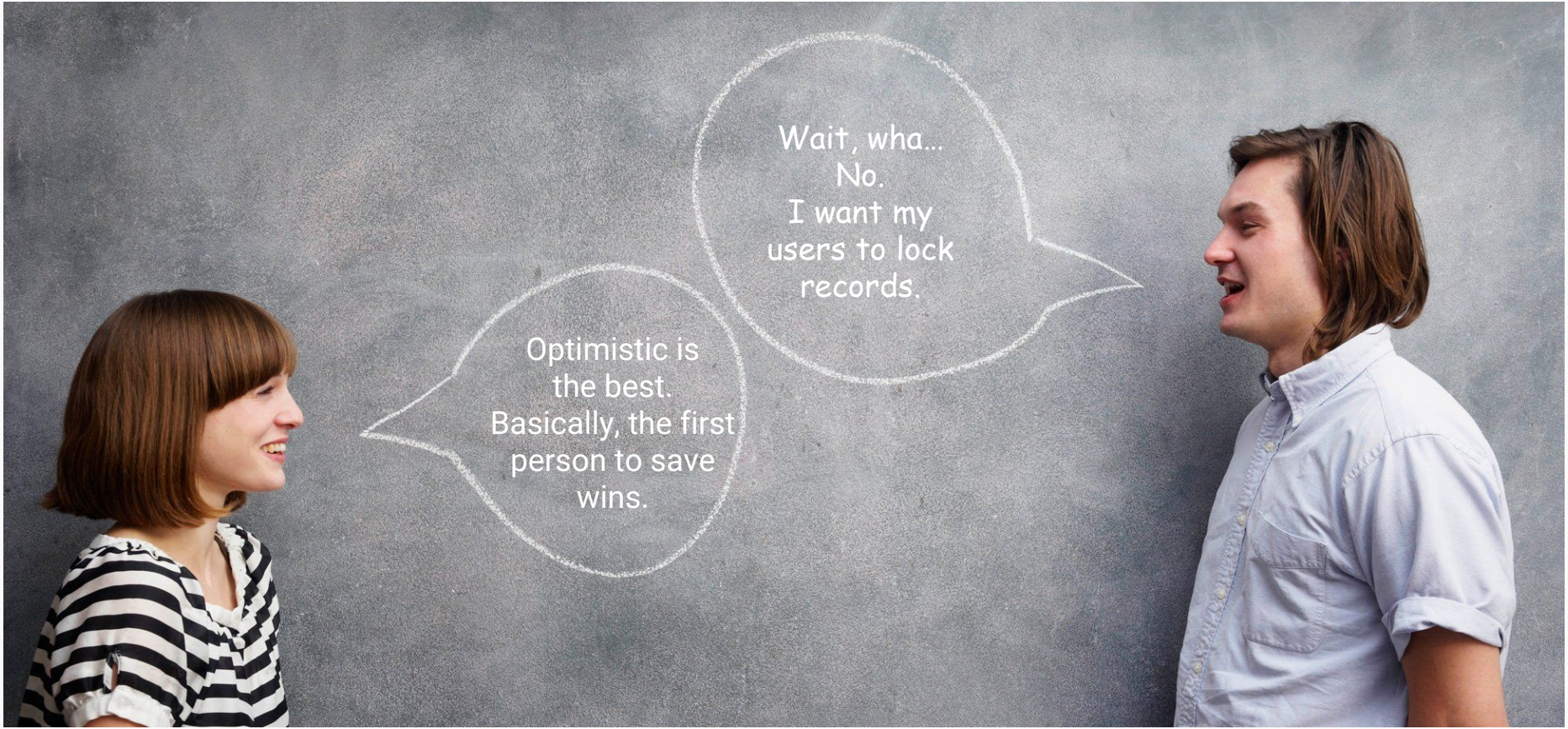
- Creates artificial transactional barrier
- Introduces greater chance for concurrency issues
- Is a poor replacement for checkpoints

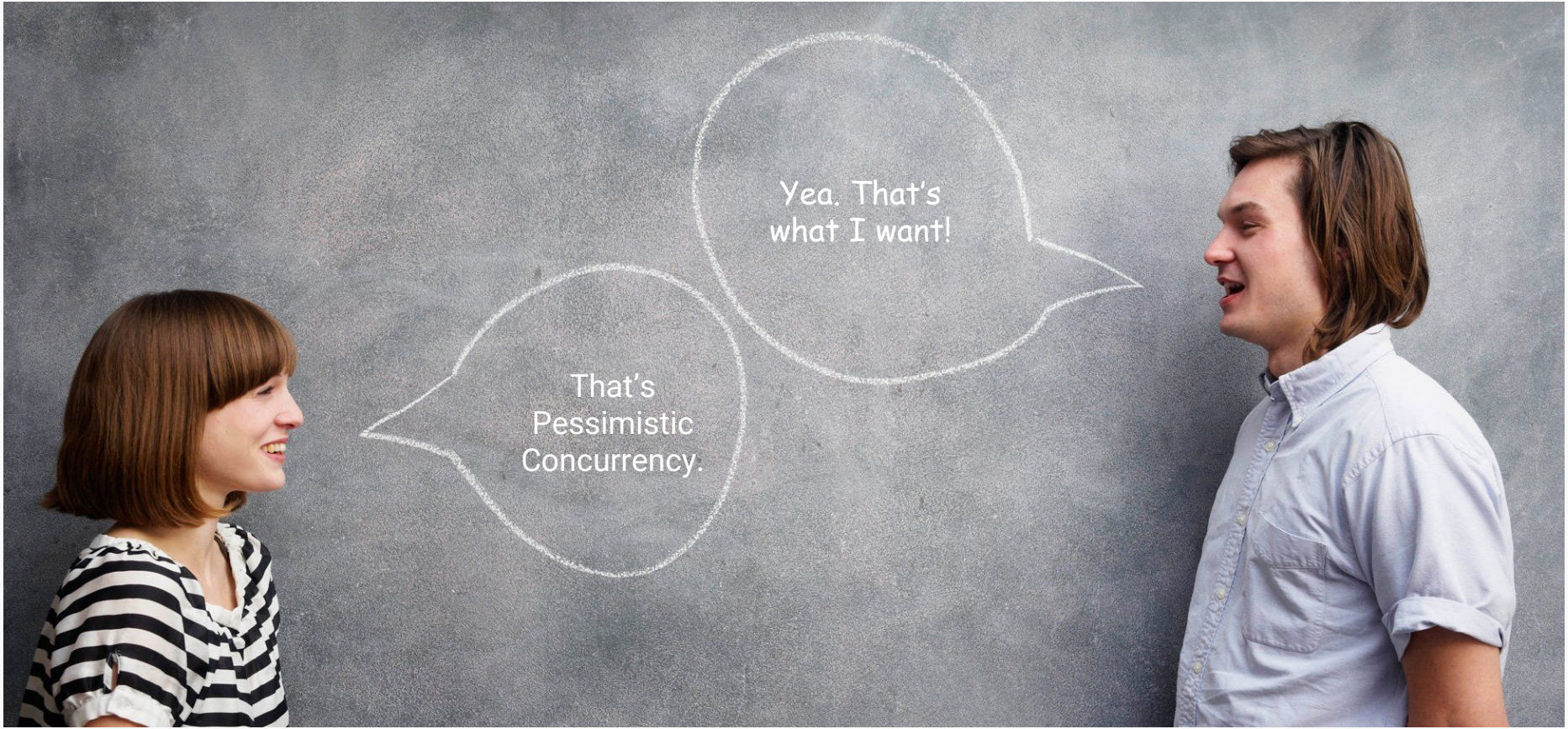
Anti-Pattern: Optimistic Concurrency

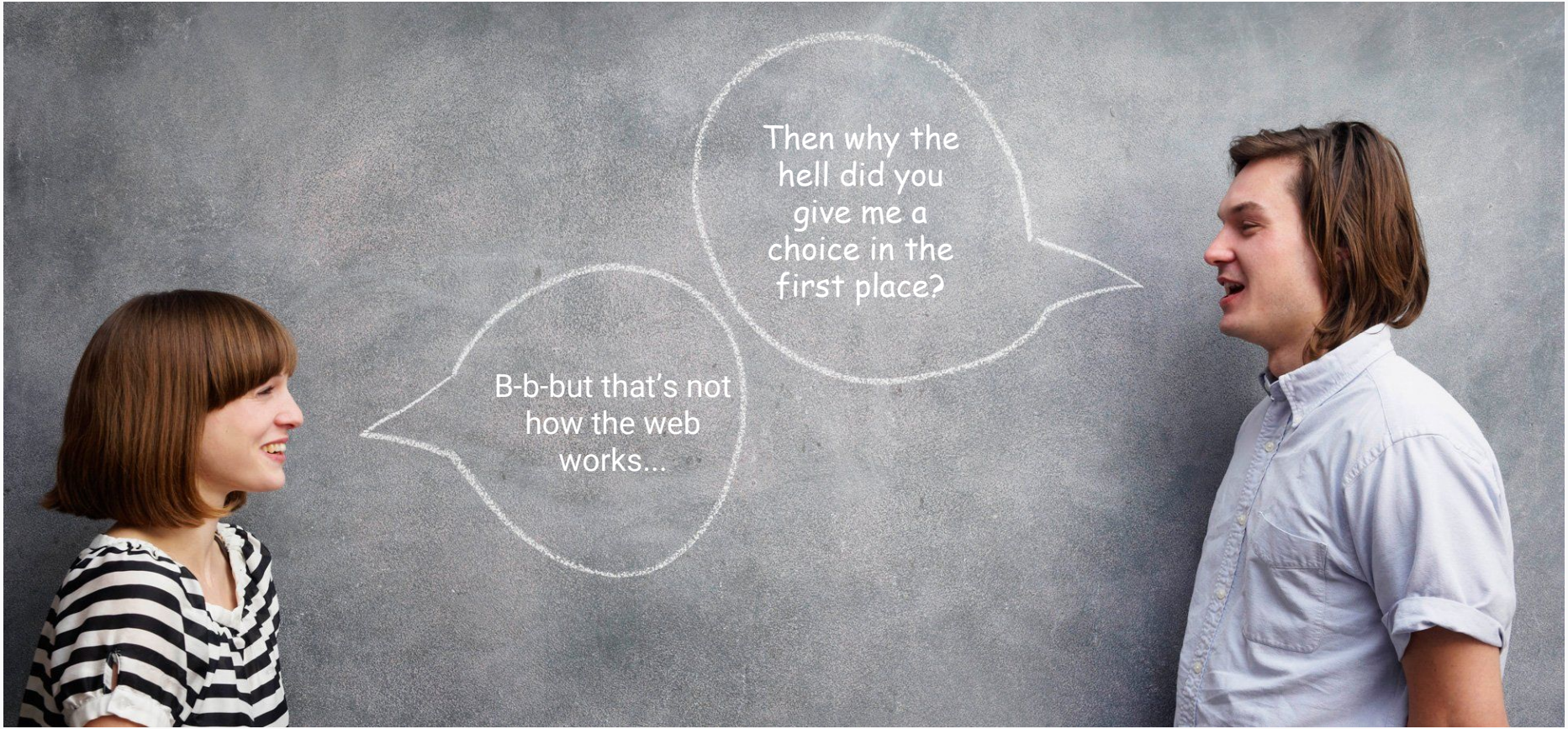
Don’t Even!


Wot?
What concurrency model do you want for your app?


I want the best.
You want **Optimistic** or **Pessimistic**?


Optimistic is the best. Basically, the first person to save wins.
Wait, wha... No. I want my users to lock records.


Yea. That's what I want!
That's Pessimistic Concurrency.


Then why the hell did you give me a choice in the first place?
B-b-but that's not how the web works...

Pattern: Notifications

- Asynchronous transactions
- “Fire-and-Forget”
- Event-Driven
- Supports Workflow

Demo

Code: [rxjs] Subject and Observable

```
private userLogout = new Subject<string>();
private userLogin = new Subject<string>();

constructor() { }

get userLogingOutEvent(): Observable<string> {
  return this.userLogout;
}
get userLoginEvents(): Observable<string> {
  return this.userLogin;
}

public onLoggingOut(userName: string): void {
  this.userLogout.next(userName);
}

public onLogin(userName: string): void {
  this.userLogin.next(userName);
}
```

Code: [rxjs] Observable .subscribe and .forEach

```
ngOnInit() {
  this.route.params.subscribe((params: Params) => {
    const id: number = +params['id'];
    this.supportRequestService.getSupportRequest(id).then(model => {
      this.supportRequestService.startViewing(model);
      this.supportRequest = model;
    });
  });
  this.router.events.forEach(event => {
    if (!this.supportRequest) {
      return;
    }
    if (event instanceof NavigationStart && event.url.substring(0, 9) === '/support/') {
      this.supportRequestService.unlockRecord(this.supportRequest);
      this.supportRequestService.stopViewing(this.supportRequest.id, this.userService.getUserName());
      this.supportRequest = null;
    }
  });
```

Code: [socket.io] account.js

```
exports.notify = function (userName, message, logins) {
    if (!logins.has(userName)) {
        console.log(`User ${userName} was not found`);
        return;
    }
    var socket = logins.get(userName).socket;
    socket.emit('notify', {
        message: message
    });
}
```

```
exports.notifyAll = function (io, message, logins) {
    io.sockets.emit('notify', {
        message: message
    });
}
```

```
exports.notifyOthers = function (io, userName, message, logins) {
    if (!logins.has(userName)) {
        io.sockets.emit('notify', {
            message: message
        });
        return;
    }
    var socket = logins.get(userName).socket;
    socket.broadcast.emit('notify', {
        message: message
    });
}
```

Code: [ng] user.service.ts

```
private subscribeEvents(socket: SocketIOClient.Socket): void {
  socket
    .on('connect', () => {
      if (this.isLoggedIn()) {
        const userName = this.getUserName();
        this.socket.emit('login', {
          userName: userName,
          token: localStorage.getItem(UserService.TOKEN)
        });
      }
    })
    .on('token acquired', token => {
      localStorage.setItem(UserService.TOKEN, token);
    })
    .on('notify', (data: NotificationDto) => {
      this.notification.next(data);
    });
}
```

Code: [ng] app.component.ts

```
ngOnInit(): void {
  this.notificationService.notificationCount.subscribe(count => this.pendingNotifications = count);
  this.userService.notification.subscribe(data => {
    this.notificationService.push(data);
  });
  this.eventAggregator.userLoginEvents.forEach(userName => this.onLoggedIn(userName));
  this.eventAggregator.userLogingOutEvent.forEach(userName => this.onLoggedOut());
  this.loggedIn = this.userService.isLoggedIn();
  if (this.loggedIn) { this.onLoggedIn(this.userService.getUserName()); } else { this.onLoggedOut(); }
}
```

Code: [ng] notification.service.ts

```
public push(notification: Notification): void {
  this.notifications.push(notification);
  this.notificationBarService.create({ message: notification.message, type: NotificationType.Info });
  this.notificationCount.next(this.notifications.length);
  this.moreNotifications.next(notification);
}
```

Pattern: Data Streaming

- Subscribe to data, rather than pull it
- Live data, rather than dead
- Just-in-time, rather than pre-cached

Demo


HANDLEp*** Address 8016a950 has base at 80100000
.6.2 irql:1f SYSVER 0xf0000565
Name
ntoskrnl.exe
atapi.sys
Disk.sys
Ntfs.sys
NTice.sys
Cdrom.SYS
KSecDD.SYS
win32k.sys
Cdfs.SYS
nbf.sys
netbt.sys
afd.sys
Parport.SYS
Dll Base DateStmp - Name
80010000
80007000
801db000
80237000
f1f48000
f228c000
f2290000
fe0c2000
fdca2000
fdc35000
f1f68000
f2008000
fdc14000
f1dd0000

Code: [socket.io] support.js + [ng] support-request.service.ts

```
socket.on('find', function (data, fn) {
    var id = data.id;
    var found = items.find((item, index) => {
        return item.id === id;
    })
    var dto = getDto(found);
    fn(dto);
});
```

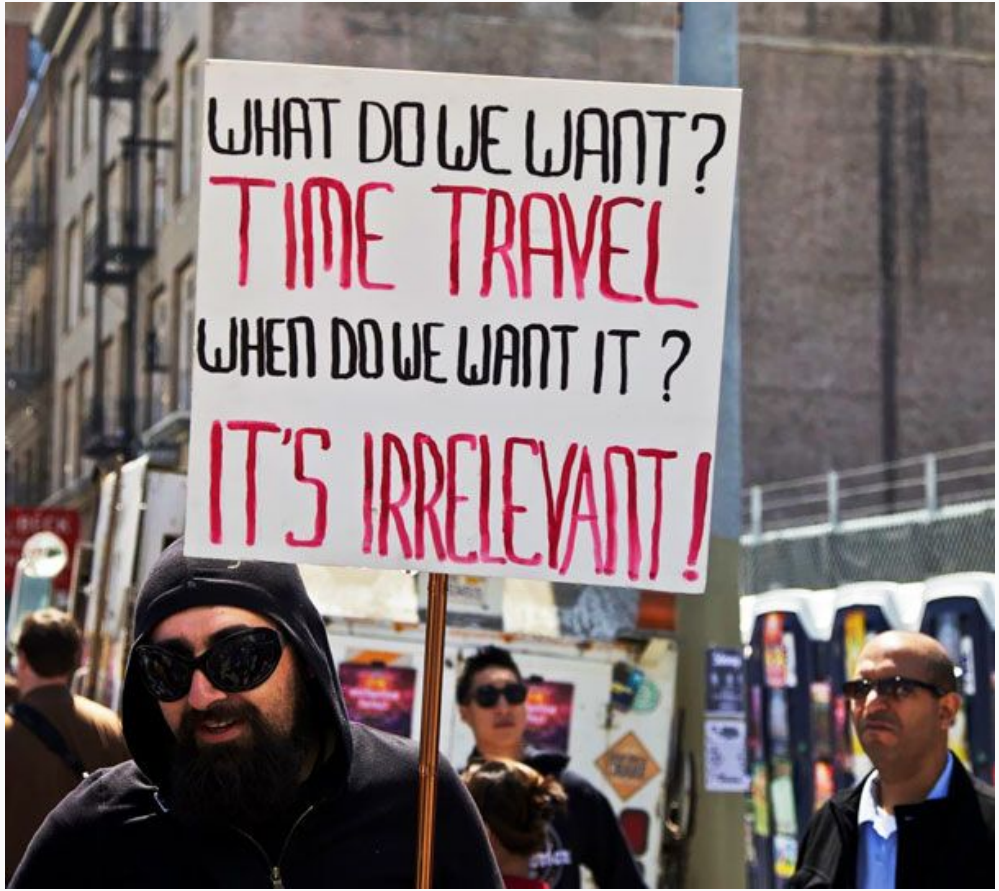
```
public getSupportRequest(id: number): Promise<SupportRequestModel> {
  return new Promise((resolve, reject) => {
    const result = this.models.find(other => other.id === id);
    if (!result) {
      this.socket.emit('find', { id: id }, (data: SupportRequestDto) => {
        const model = this.addModel(data);
        resolve(model);
      });
    } else {
      resolve(result);
    }
  });
}
```

Code: [ng] support-request.service.ts & support-request-item.component.ts

```
public getSupportRequest(id: number): Promise<SupportRequestModel> {
  return new Promise((resolve, reject) => {
    const result = this.models.find(other => other.id === id);
    if (!result) {
      this.socket.emit('find', { id: id }, (data: SupportRequestDto) => {
        const model = this.addModel(data);
        resolve(model);
      });
    } else {
      resolve(result);
    }
  });
}
```

```
this.supportRequestService.getSupportRequest(id).then(model => {
  this.supportRequestService.startViewing(model);
  this.supportRequest = model;
});
```

Demo



Code: [ng] + [socket.io]

```
public requestUpdate() {
  this.models = new Array<SupportRequestModel>();
  this.socket.emit('get', { id: null });
}
```

support-request.service.ts [ng]

```
socket.on('get', function () {
    for (var i = 0; i < items.length; i++) {
        var dto = getDto(items[i]);
        socket.emit('nextItem', dto);
    }
});
```

```
this.socket.on('nextItem', (data: SupportRequestDto) => {
  this.addModel(data);
});
```

```
this.supportRequestSubject.next(model);
```

```
this.supportRequestService
   .supportRequestModels
   .subscribe(m => this.supportRequests.push(m));
```

```
<li *ngFor="let supportRequest of supportRequests"
```

Support.js
[socket.io]

support-request-list.component.ts [ng]

support-request-list.component.html [ng]

Pattern: Record Locks

- Prevents lost work
- Informs others of what's being done
- Models real-life workflows

Demo

Code: [ng] support-request.service.ts & [socket.io] support.js

1

```
this.socket.emit('lock', { id: model.id, userName: this.userService.getUserName() });
```

2

```
socket.on('lock', function (data) {
    var id = data.id;
    var userName = data.userName;

    var entity = items.find((item) => {
        return item.id === id;
    });
    entity.locked = true;
    entity.lockedBy = userName;

    socket.nsp.emit('locked', {
        id: id,
        userName: userName
    });
});
```

3

```
this.socket.on('locked', data => {
  const id = data.id;
  const userName = data.userName;
  const found = this.models
    .find(model => model.id === id);
  if (!found) {
    return;
  }
  found.locked = true;
  found.lockedBy = userName;
});
```

Code: [ng] support-request.service.ts

```
public lockRecord(model: SupportRequestModel): void {
  this.socket.emit('lock', { id: model.id, userName: this.userService.getUserName() });
}

public unlockRecord(model: SupportRequestModel): void {
  this.socket.emit('unlock', { id: model.id });
}
```

Pattern: Soft Locks

- Lock single fields or groups of fields
- Explicit or JIT locking
- Batch or debounce unlocking

Demo

Code: [ng] field-wrapper.ts

```
set value(toSet: T) {
    if (!this.editing) {
        this.socket.emit('editing', {
            id: this.id,
            userName: this.userName,
            fieldName: this.name
        });
        this.editing = true;
    }
    this.utilityService.debounce(this, 5000).forEach(() => {
        this.socket.emit('release', {
            id: this.id,
            userName: this.userName,
            fieldName: this.name,
            value: this.value
        });
        this.editing = false;
    });
    this.mutator(toSet);
}
```

```
get locked() {
    return (this.fieldLocked || this.entityIsLocked(this.userName));
}
```

Code: support.js

```
socket.on('editing', function (data) {
    var id = data.id;
    var userName = data.userName;
    var fieldName = data.fieldName;
    socket.broadcast.emit(`editing-${id}-${fieldName}`, {
        userName: userName
    })
});
```

```
socket.on('release', function (data) {
    var id = data.id;
    var userName = data.userName;
    var fieldName = data.fieldName;
    var newValue = data.value;

    var entity = items.find((item) => {
        return item.id === id;
    });
    if (!entity) {
        return;
    }
    entity[fieldName] = newValue;
    socket.broadcast.emit(`released-${id}-${fieldName}`, {
        userName: userName,
        value: newValue
    })
})
```

Code: [ng] field-wrapper.ts & any ng markup

```
socket.on(`editing-${id}-${this.name}`, data => {
    this.fieldLocked = true;
    this.lockedBy = data.userName;
});
socket.on(`released-${id}-${this.name}`, data => {
    this.fieldLocked = false;
    this.lockedBy = null;
    this.mutator(data.value);
});
```

```
[disabled]="supportRequest.description.locked" [(ngModel)]="supportRequest.description.value"
```

Pattern: CQRS

Command Query Response Segregation

- Eventually consistent
- Treat transactions and views separately
- Use right technology for specific problems

Bringing it All Together

#Winning @
http://m.eado.ws/st2017
http://m.eado.ws