Modern webbutveckling

av Robert Welin-Berger
robertwb@kth.se

Modern webbutveckling

1. Projektstorlek och Arkitektur
2. Callbacks
3. Event driven arkitektur
4. MEAN stack
5. ODM/ORM

1. **Projektstorlek och Arkitektur**

- Flexibilitet mot struktur
- Frivillig struktur
- Stöttande struktur
- Påtvingad struktur
- Vad ska man välja

Frivillig struktur

- Ultimat flexibilitet
- De rena språken
- PHP, Node.js och Java
- Språkskillnader
- Bibliotek
- Små hack och tester

Stöttande struktur

- Ramverk som kan partiellt integreras
- Angular.js, Express och Backbone.js
- Integrationen kan utökas med projektet
- Enormt stor bredd på flexibilitet

Påtvingad struktur

- Java EE
- Wordpress (CMS)
- Spring
- Både för enkla och stora saker

2. Callbacks

- Definition
- Syncronous example
- Asynconous example
- Closure
- Usage

Definition

- “A callback is a piece of executable code that is passed as an argument to other code, which is expected to call back (execute) the argument at some convenient time”
- “The invocation may be immediate as in a synchronous callback”
- “or it might happen at later time as in an asynchronous callback”

Callback

```
function someAction(x, y, someCallback) {
    return someCallback(x, y);
}

function calcProduct(x, y) {
    return x * y;
}

alert(someAction(5, 15, calcProduct));
```

Callback

```
<button onclick="myFunction()">


<script>
function myFunction() {
    console.log("Hello World");
}
</script>
```

Closure

- Att kunna nå variabler i ett yttre scope

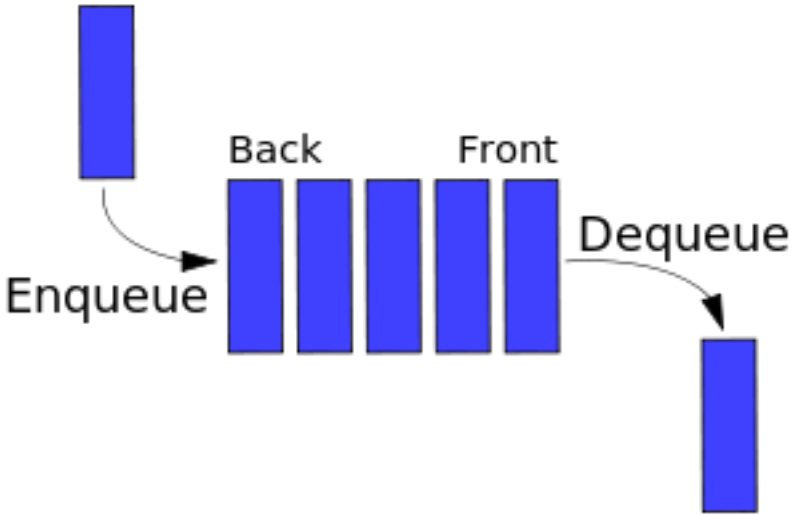
```
function bestSellingBooks(threshold) {
  var x = 5
  return bookList.filter(
      function (book) {
          console.log(x);
          return book.sales >= threshold;
  } );
}
```

3. Event driven arkitektur

- Callbacks kön
- Callbacks i stället för trådar
- Singletrådat
- Lätt backend
- Bokningssystem -exempel

Callback Kön

- Kollas en i taget
- Körs eller återsätts
- Går snabbt, en bit kollas



Callbacks i stället för trådar

- En processor kör processer en i taget
- Kan simplifieras till Round-robin
- Återimplementation?
- Storleksordningar snabbare att byta

Singletrådat

- Race conditions, vad är det?
- Färre saker att tänka på
- Tydliga event
- När det växer - Distribuering
- Slipper “internal states”
- Callback hell

Lätt backend

- Små anrop
- Interaktiva webbsidor
- API liknande strukturer
- Tyngre front-end, Angular osv

Bokningssystem

- Ber om tider
- Små requests
- Ber om data ofta
- Många requests sammtidigt
- 500/s => 1,000,000 per dag

MEAN stack

- Stack
- Mongodb
- Node
- Express
- Angular
- Queue App - example

Stack

- Frontend
- Backend
- Databas
- “Staplade på varandra”
- Integrerade med varandra

Mongodb

- No-SQL
- Bra på många sätt
- Inte SQL = bra
- Data är ofta relationell
- Överanvänds ofta

Node

- Javascript på backenden
- Google Chrome V8
- IO.js/Node.js
- Omfattande bibliotek för webben
- Tillgång till alla javascript bibliotek

Express

- Ramverk för routing och mycket annat
- Minimalistiskt
- Middleware
- Väl intergrerat med Databaser

Express - Example

```
// A middleware with no mount path.
// It gets executed for every request to the app


app.use(function (req, res, next) {
  console.log('Time:', Date.now());
  next();
});


// Generate paths for all files in public folder
app.use(expressio.static(__dirname + '/public'));
```

Angular

- MVW
- Det är en helt egen miljö
- Flexibel integration
- Single page application
- Väl integrerat med andra populära ramverk
- Two way data binding

Queue App - Basic setup

```
var expressio = require('express.io');

var app = expressio();
app.http().io();

app.use(expressio.static(__dirname + '/public'));

app.io.on('connection', function(socket){
  console.log('a user connected');
});
```

Queue App - Websocket Routing

```
app.io.route('listen', function(req) {
  console.log('a user added to ' + req.data);
  req.io.join(req.data);
})

app.io.route('join', function(req) {
  console.log('a user joined to ' + req.data.queue);
  app.io.room(req.data.queue).broadcast('join', req.data.user);
})
```

Queue App

```
<table>
    <tbody><tr ng-repeat="user in users">
        <td>{{ user.name }}</td>
        <td>{{ user.place }}</td>
        <td>{{ user.comment }}</td>
    </tr></tbody>
</table>
```

5. ORM/ODM

- ORM = SQL, ODM = mongodb
- Mappar ner objekt direkt mot databasen
- Säkrar datan på servern vid krash
- Enkelt återuppta körning
- Enkelt sätt att skriva lätta dataskopplingar
- Långsammare
- Skalar inte alltid som det ska.

ODM - Exempel

- Påminner mycket om vanliga klasser
- Lägger till typer

```
var courseSchema = new Schema({
    name: String,
    open: { type: Boolean, default: true },
    active: { type: Boolean, default: true },
    queue: [userSchema],
    admin: {type:[adminSchema], default: []}
});
```