Node och Javascript

av Robert Welin-Berger
robertwb@kth.se

Modern webbutveckling

1. Javascript origins
2. Javascript essentials
3. Callbacks
4. Promises
5. Node
6. Express

Javascript Origins

- Namnet var ett rent “marketing move”
- Utvecklades på Netscape
- Ett hastigt bygge som inte han putsas
- Ganska missförstått
- Användare och syfte påverkar åsikter

Javascript Origins

- Är numera ett samarbete.
- Brokig utvecklig och stöd
- Olika buggar i olika webläsare
- Allt ändras snabbt, sätter spår i attityden
- Nya reformer som Typescript
- Nya försök som webasm

1. Javascript essentials

- == vs ===
- NaN
- self refererar till objektet i scope
- Inget scope för blocks, bara funktioner
- (function(){//stuff})();
- Man undviker att fylla global namespace
- var self = this;

1. Javascript essentials - continued

- Brackets kommer alltid på samma rad!
- Kommatecken är “optional”
- if (typeof obj === 'object' && obj !== null) {
- [1,2,3] == [1,2,3]; //false
- undefined = "I'm not undefined!";//skrivs över

Prototypes

- Ser ut som klasser
- Är inte klasser
- Ingen djup förståelse krävs för kursen
- Beror på version av JS
- ES5 - zack = Object.create(person)
- ES6 - zack.__proto__ = person
- ES6 there are “classes”, actually wrappers

Prototypes - continued

- var person = { fName: "Paul", lName: "Irish" }
- var mark = new Person("Mark", "Miller");
- En map String -> x
- zack['lname'] = "rilegious";
- Funktioner är “first class objects”
- Allt kan ändras dynamiskt
- Det spelar roll HUR man lägger till funktioner

3. Callbacks

- Definition
- Syncronous example
- Asynconous example
- Closure

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”

Synchronous example

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

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

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

Asynchronous example

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


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

Closure

- Att kunna nå variabler i ett yttre scope

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

Promises

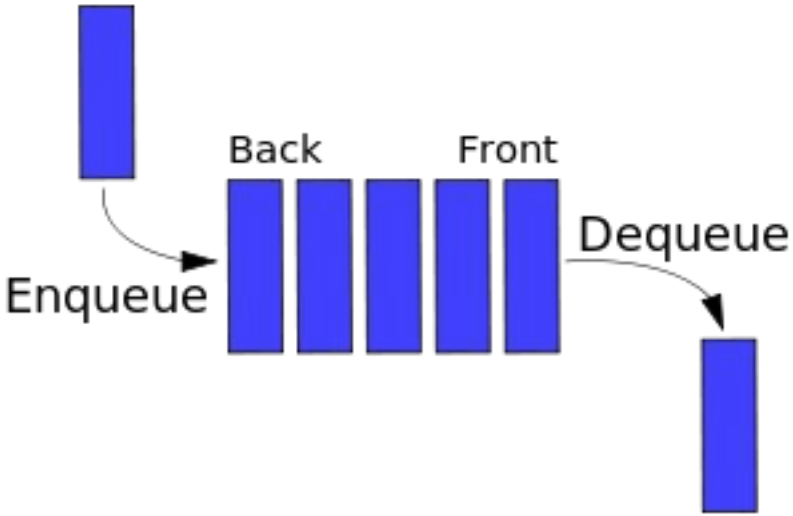
- Callbacks fungerar inte alltid bra
- Tre olika states
- Pending
- Fulfilled
- Rejected
- Inte standardiserat i praktiken .then .done
- Bättre för linjära sekvenser av händelser

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 samtidigt
- 500/s => 1,000,000 per dag

Node

- Javascript på backenden
- Google Chrome V8/Microsoft Chakra
- Node.js (IO.js fanns ett tag)
- Omfattande bibliotek för webben
- Tillgång till alla javascript bibliotek
- Microsoft IoT

Express

- Ramverk för routing och mycket annat
- Minimalistiskt
- Middleware
- Standarden, det mesta använder Express

Express - Example

```
// 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(express.static(__dirname + '/public'));
```

Queue App - Basic setup

```
var express = require('express);
var app = express();
var httpServer = http.Server(app);
var io = require('socket.io').listen(httpServer);

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

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

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);
})
```