REALTIME WEB APPS

Fingoweb PSD organizer

SLAWOMIR.WILUSZ@FINGOWEB.COM - FINGOWEB.COM - PSDORGANIZER.COM - @2017

POLLING

- <meta http-equiv="refresh" content="5">
- window.location + setTimeout
- AJAX
 - Short polling
 - Long polling

PUSH

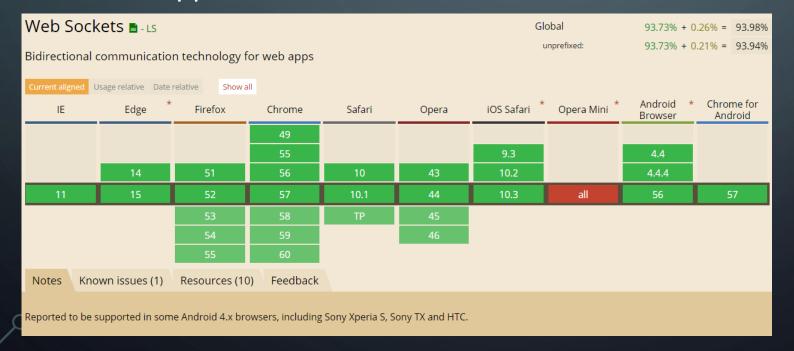
• server-sent event



WEBSOCKETS

Advantages of Websockets over SSE:

- Real time, two directional communication.
- Native support in more browsers



SERVER-SIDE – IN MEMORY

• socket.io

Socket.IO is a JavaScript library for realtime web applications. It enables realtime, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have a nearly identical API.

SERVER-SIDE – DB CHANGE FEED

- Oplog / Binlog
 - MongoDB
 - MySql
- Triggers
 - PostgreSQL LISTEN/NOTIFY
- Dedicated solution
 - RethinkDB "changes()"
 - CouchDB "_changes"

PUTTING THE PIECES TOGETHER

- React
- NodeJS
- FeathersJS +RxJS + Socket.io
- RethinkDB

DEMO – SERVER SIDE

```
package.json × ii index.html ×
                             📭 app.js ×
                                         client.js ×
 const feathers = require('feathers');
 const socketio = require('feathers-socketio');
 const handler = require('feathers-errors/handler');
 const r = require('rethinkdbdash')({
   db: 'realtimeapp'
⊕});
 const rethinkdb = require('feathers-rethinkdb');
 const app = feathers();
 app.configure(socketio());
 app.use('/comments', rethinkdb({
   Model: r,
   name: 'comments'
<u>}}));</u>
 app.use('/', feathers.static(_dirname));
 app.use(handler());
 app.listen(3001);
```

DEMO – CLIENT SIDE

```
📠 package,json × 🖹 index.html × 📴 app.js × 🛅 client.js ×
   import ic from 'socket.io-client';
   import socketio from 'feathers-socketio/client';
   import React from 'react';
   import ReactDOM from 'react-dom';
   import rx from 'feathers-reactive';
   import RxJS from 'rxjs';
   const socket = io();
   const app = feathers()
    .configure (socketio (socket))
    .configure (rx(RxJS));
   const commentService = app.service('comments');
   class CommentsList extends React.Component {
    componentDidMount() {
      commentService.find({ query: {
          $sort: {
         .subscribe(<u>comments</u> => this.setState({<u>comments</u>}));
      commentService.remove(item.id);
       return false;
      if (this.state && this.state.comments) {
           {this.state.comments.map(comment =>
               <Created: {new Date(<u>comment</u>.created).toISOString()} <a href="#" onClick={this.remove.bind(this, <u>comment</u>)}>x</a><br/>cyrpe>
           Rendered: {new Date().toISOString()}
```

DEMO – CLIENT SIDE 2

```
package.json × 🕩 index.html × 📭 app.js × 📭 client.js ×
   class CommentForm extends React.Component {
     constructor (props) {
       super (props);
       this.state = {
         text: '',
     updateProperty(prop, ev) {
      this.setState({[prop]: ev.target.value});
      ev.preventDefault();
       commentService.create({
         text: this.state.text,
       )).then(() => this.setState({
        text: ''.
         created: null
       return <form onSubmit={this.submit.bind(this)}>
           <input type="text" onChange={this.updateProperty.bind(this, 'text')} value={this.state.text}/>
          <button type="submit">Add</button>
         </div>
     <CommentForm />
   </div>, document.getElementById('app'));
```

DEMO – PSDORGANIZER.COM



THE END

THANK YOU

Fingoweb

