This work is licensed under CC BY 4.0

Realtime Web Applications



Content

- ✓ What is real-time web?
  - Definition
  - Examples of real-time web applications
- ✓ Technologies for real-time web applications
  - History, problems, today
- ✓ Real time for the Node.js platform and Express
  - Using Socket.IO
  - Using webhooks

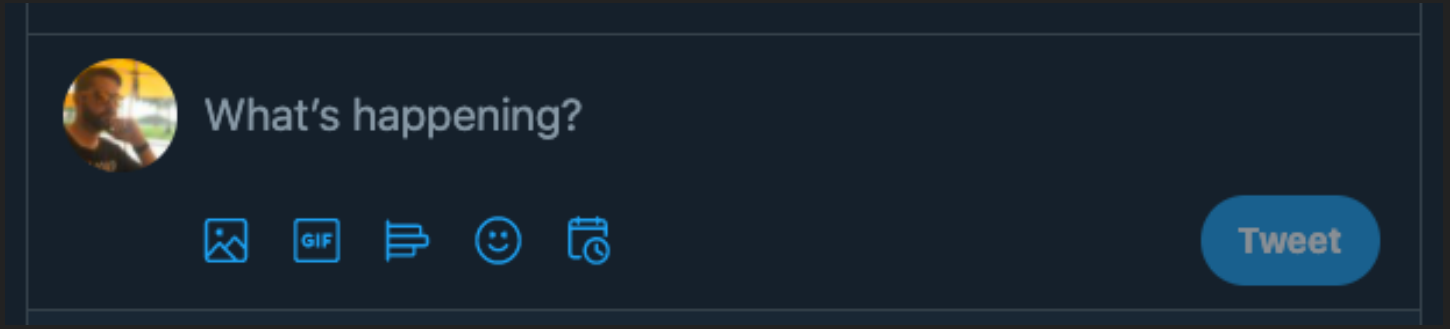
Definition

> Real-time web applications is clients receives data updates in a short time interval after the data is changed/created and without need for additional requests.

- ✓ What do we mean by short time? (definitions from Wikipedia)
  - Hard – missing a deadline is a total system failure.
  - Firm – infrequent deadline misses are tolarable, but may degrade the system's quality of service. The usefulness of a result is zero after its deadline.
  - Soft – the usfulness of a result degrades after its deadline, thereby degrading the system's quality of service.
- ✓ Broadcast to many clients
- ✓ Lots of different techniques (and hacks) for web applications
  - Not the way the web was intended to work!

What kind of applications?

- ✓ Web-based games
- ✓ Social communication
  - Chat, Live comments, feeds…
  - Notifications
- ✓ Collaboration tools



What kind of applications?

- ✓ 2nd Screen Experience
  - Activity streams (live comments)
  - Statistics
    - Live sport
    - Video game extras
  - Voting, Peer-instructions



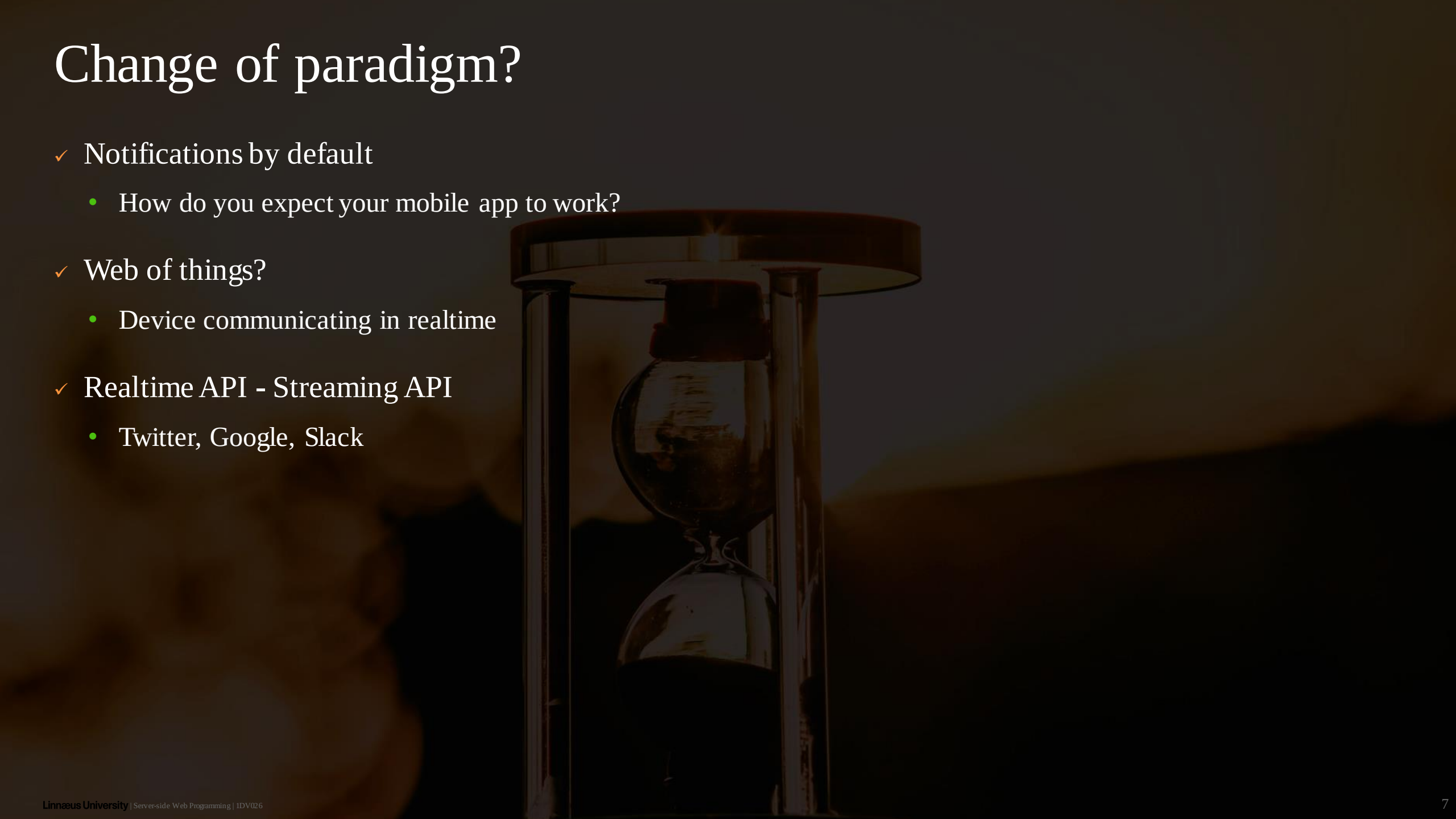
What kind of applications?

- ✓ Decision making
  - Trading
  - Automatic price adjusting
  - Monitoring infrastructure and application
    - Users, servers…



Change of paradigm?

- ✓ Notifications by default
  - How do you expect your mobile app to work?
- ✓ Web of things?
  - Device communicating in realtime
- ✓ Realtime API - Streaming API
  - Twitter, Google, Slack

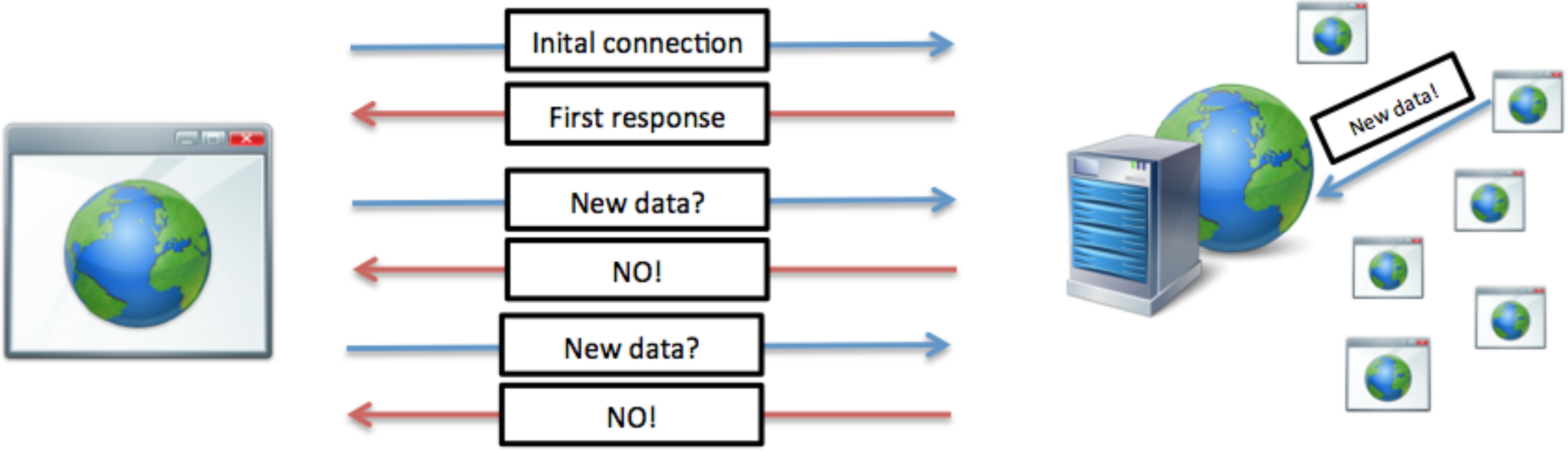




Technologies for real-time web

The problem with http…

…in real-time scenario



History

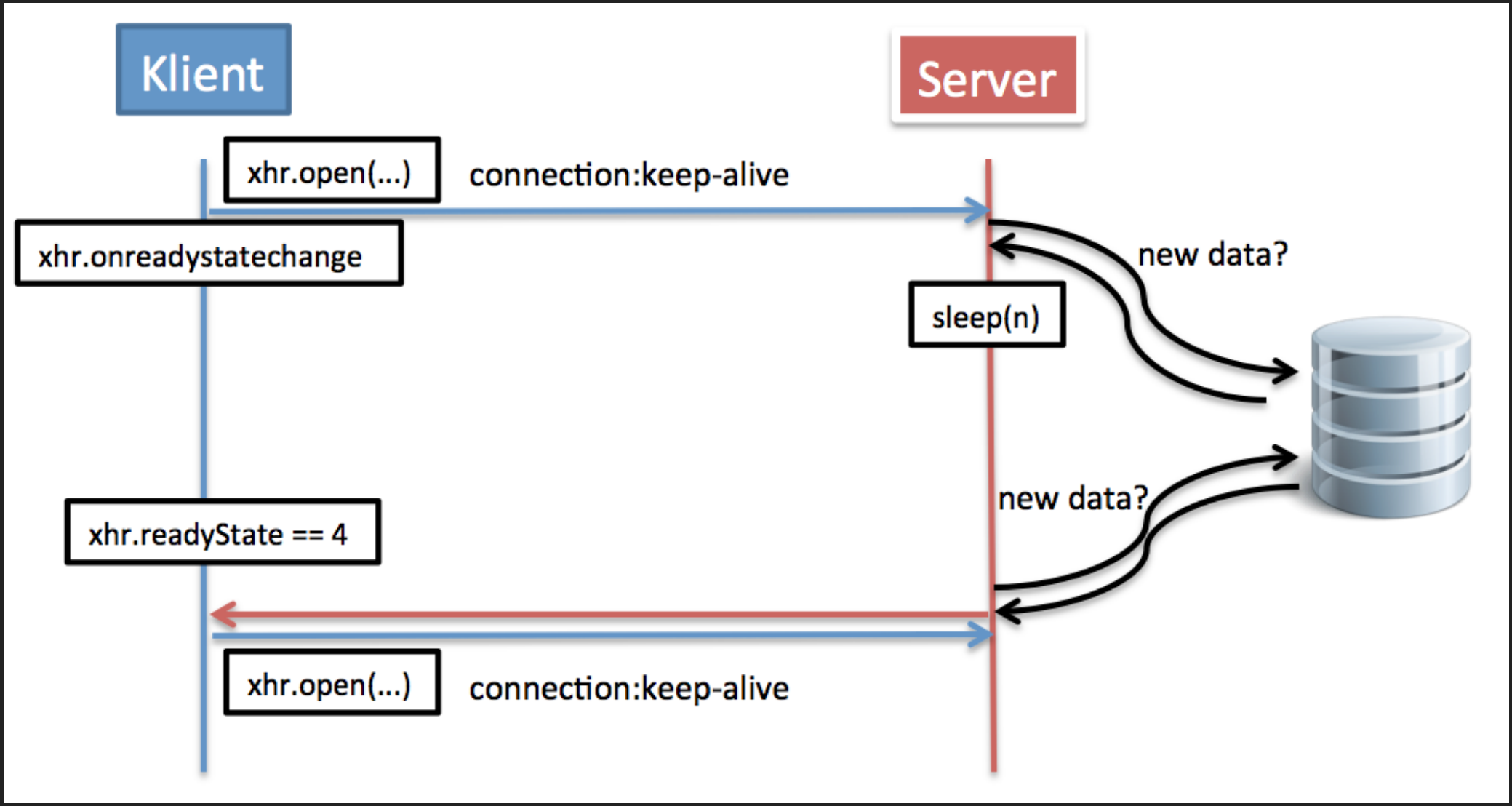
- ✓ JAVA Applets (Netscape 2.0 – 1996)
  - Java virtual machine
  - Raw TCP-socket
- ✓ Flash platform
  - RTMP-servers, Shared Objects
  - Flash client
    - ✝ 2021
- ✓ Silverlight (.NET)
  - Well…. (no more developmenbt since 2013)



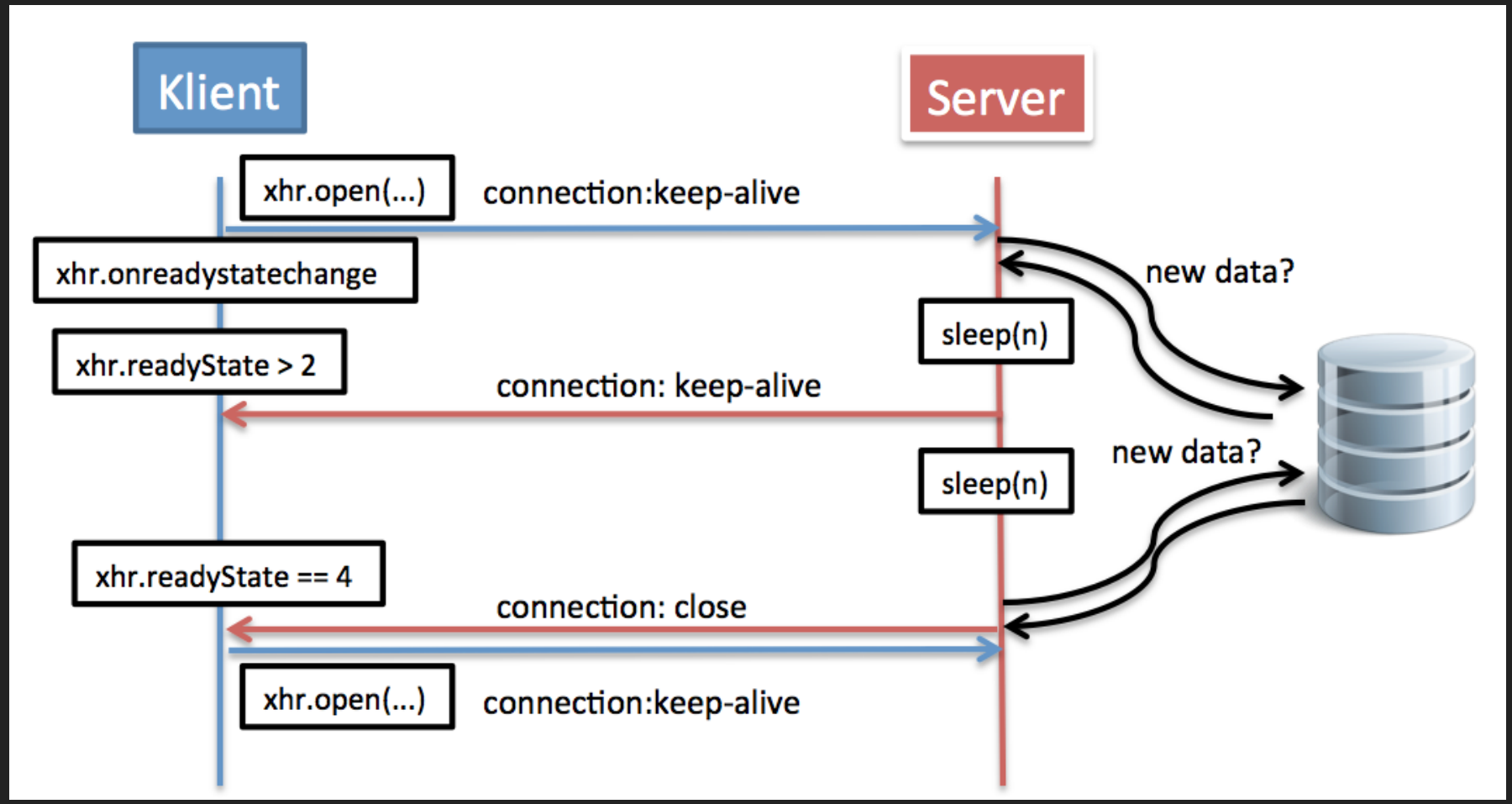
HTML and JavaScript

- ✓ Comet (umbrella term)
  - Techniques that goes around the HTTP 1.1 spec. and browser implementations
    - Persistent HTTP connections, Domain sharding…
  - Long-lived hidden iframe
  - XHR long polling
  - XHR streaming
- ✓ Server-sent events
  - Just from server, no bi-directional or full duplex (combine with XHR)
- ✓ WebSocket

XHR Long Polling



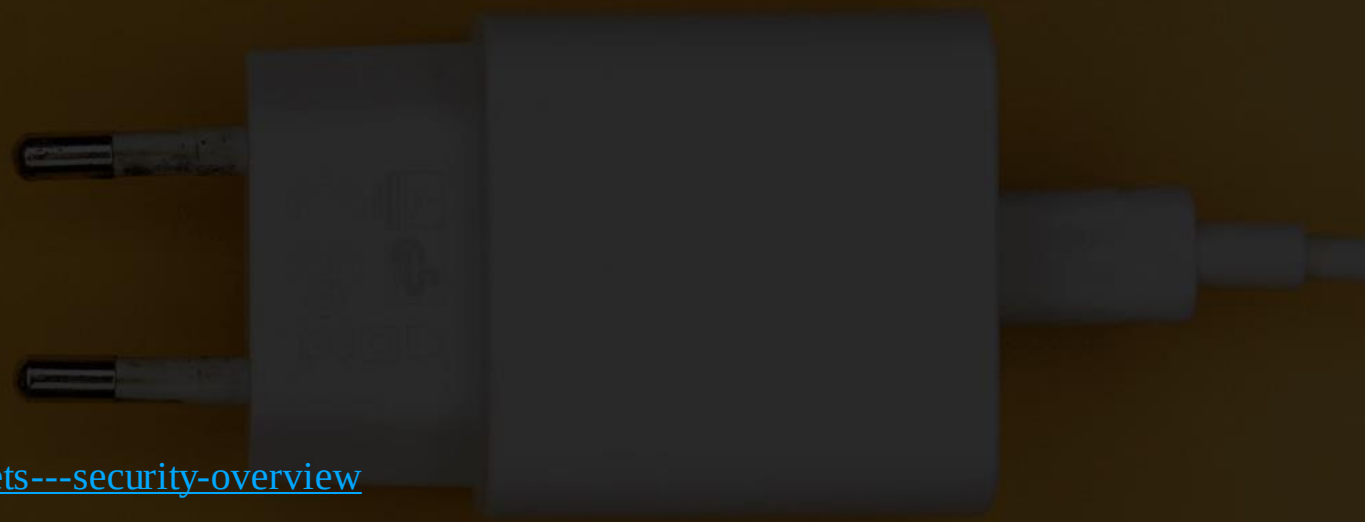
XHR Streaming



- ✓ "Transfer Encoding: chunked"
  - Do not need to open new connection when data is received

WebSocket

- ✓ Socket?
- ✓ Connect on HTTP(S), Upgrade to ws:// or wss://
  - Firewalls, proxies? Default port 80/443
- ✓ Two-way communication, full-duplex
  - Text or/and binary data
- ✓ IETF Standard, RFC 6455, 2011
  - WebSocket API for browsers
- ✓ Security
  - Use Content Security Policy
    - https://gist.github.com/subudeepak/9897212#websockets---security-overview
  - Encryption with wss://



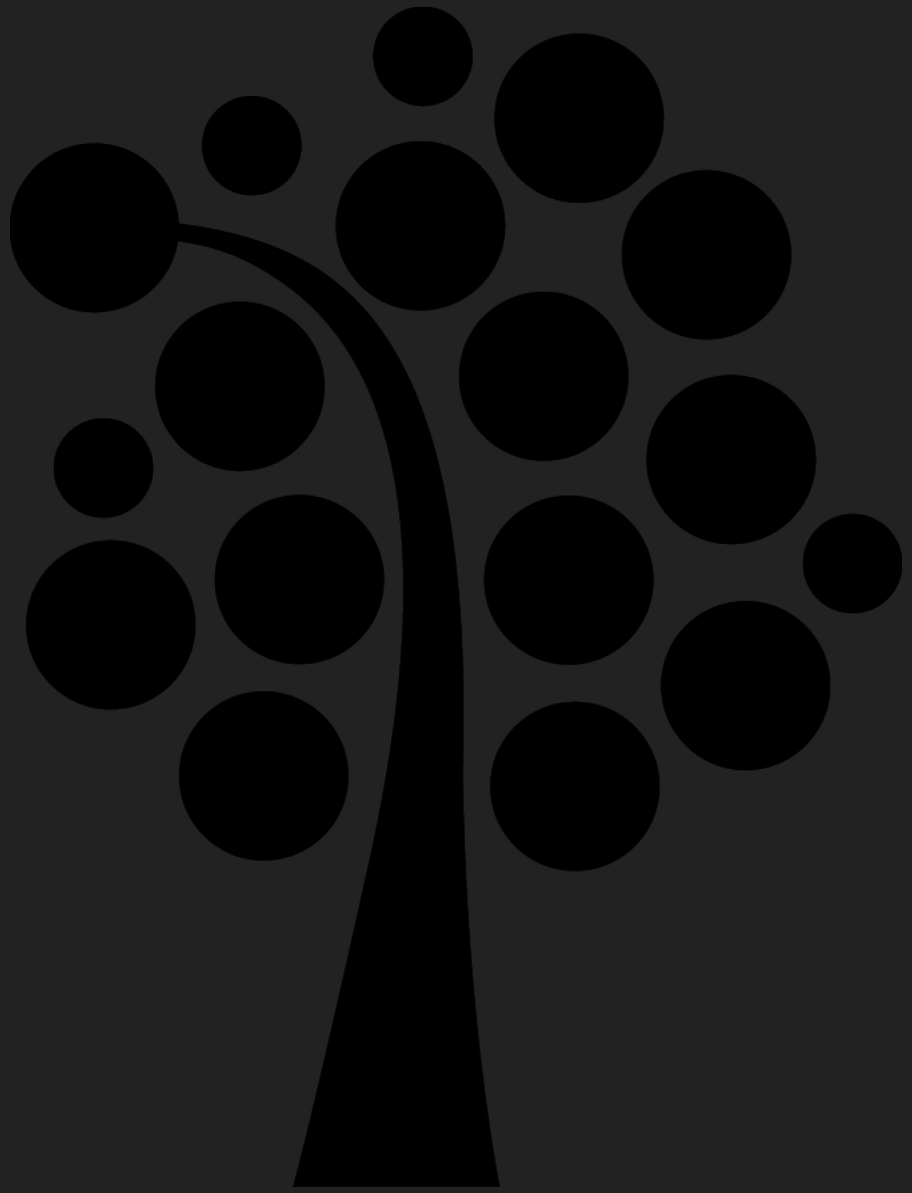
Challenges with WebSockets

- ✓ Browser support
  - Fallback strategies
- ✓ Concurrency
  - Synchronization
  - Fault-tolerant?
- ✓ Offline, Queue-strategies



Other Architectures

- ✓ Client-to-Server
  - What we just said….
- ✓ Client-to-Client
  - P2P (peer-to-peer)
  - WebRTC
- ✓ Server-to-Server
  - Webhooks



This work is licensed under CC BY 4.0

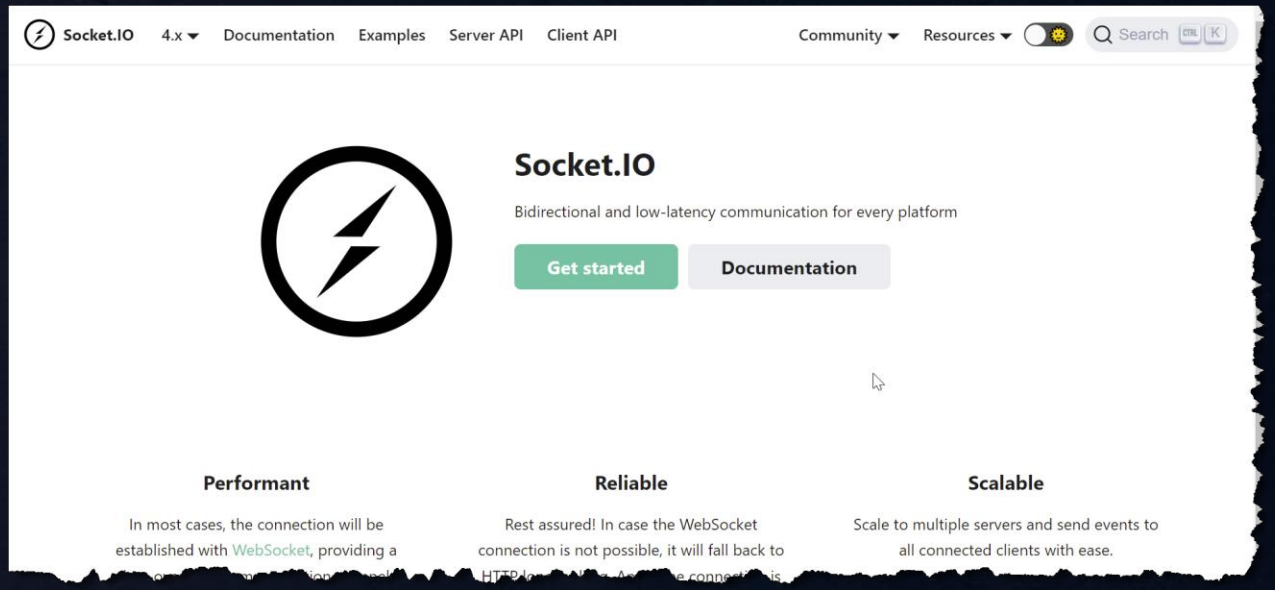
Example

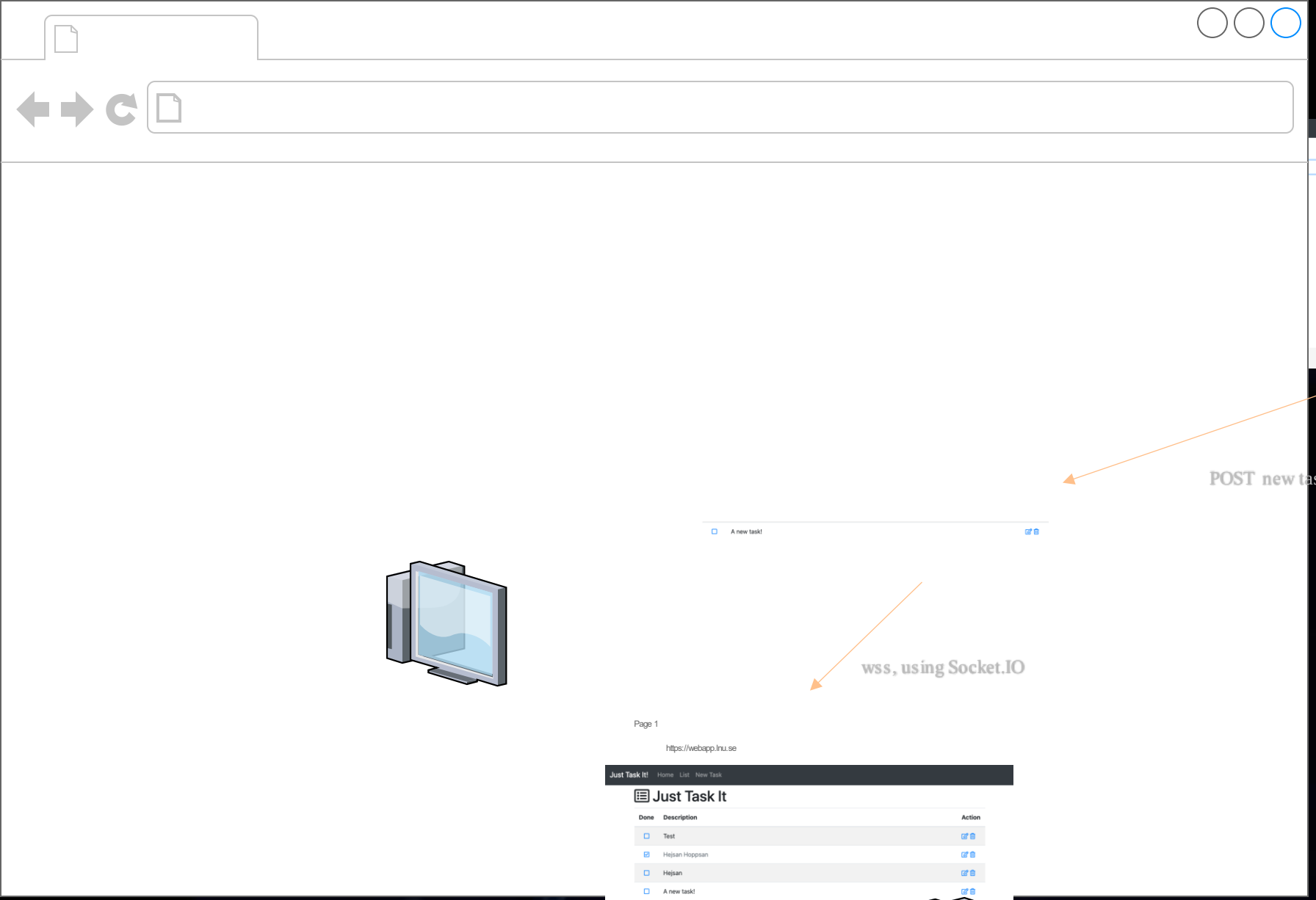
Just Task It featuring WebSockets

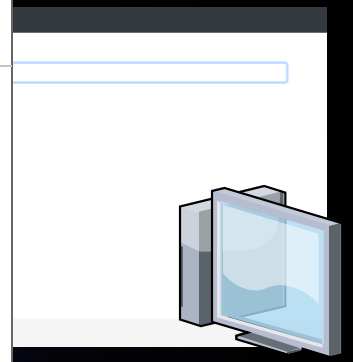


WebSocket Server on the Node.js Platform

- ✓ Socket.IO (https://socket.io)
- ✓ ws (https://www.npmjs.com/package/ws)
- ✓ …







POST new task

wss, using Socket.IO

Client

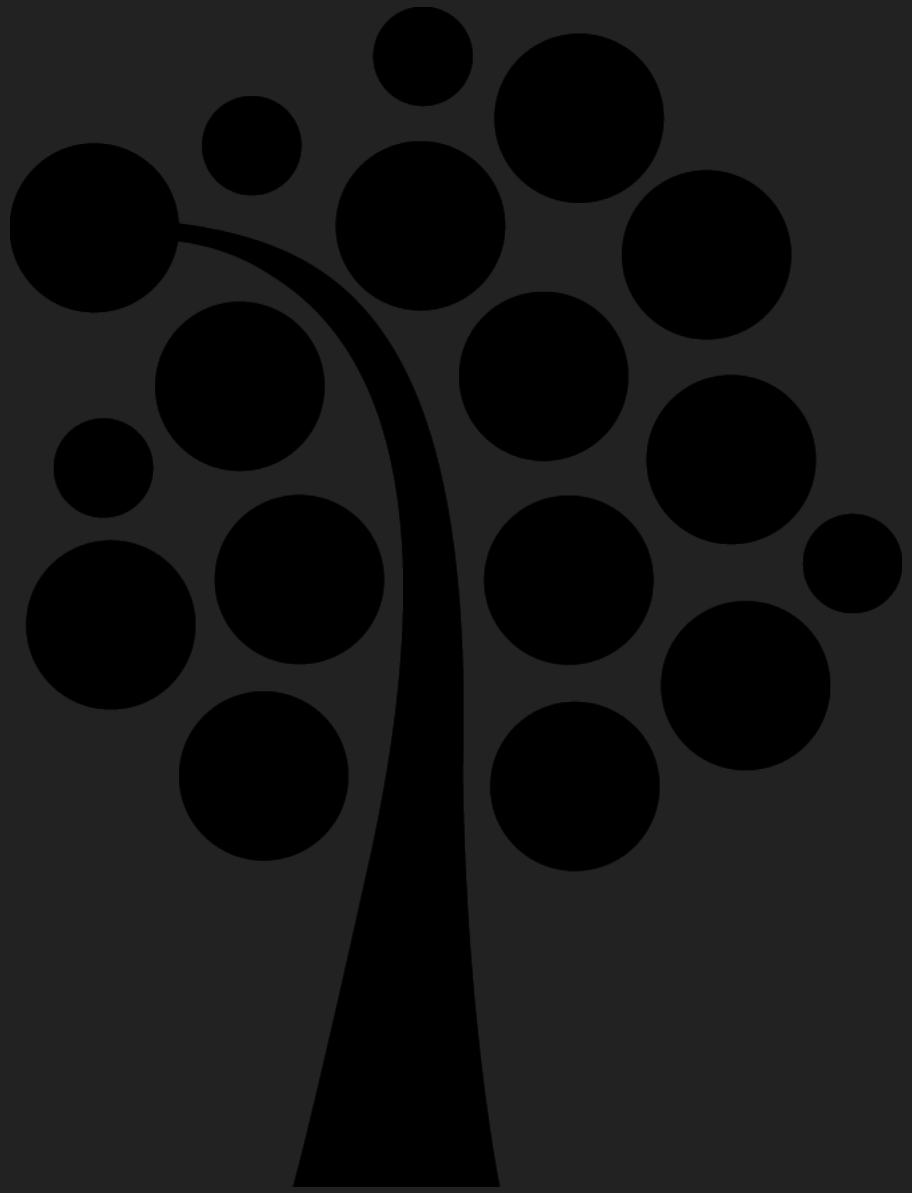
Example, Express – Socket.IO

```
$ npm install socket.io
```

```
...
// WebSocket support with Socket.IO.
import { createServer } from 'node:http'
import { Server } from 'socket.io'

try {
  ...
  // Socket.IO: Create an HTTP server and pass it to Socket.IO. (Make sure to call .listen on the server, not the app.)
  const httpServer = createServer(app)
  const io = new Server(httpServer)
  ...
  app.use((req, res, next) => {
    ...
    // Socket.IO: Add the io object to the response object to make it available in controllers.
    res.io = io
    ...
  })
  ...
  // Starts the HTTP server listening for connections.
  // Socket.IO: Call .listen on the server, not the app.
  httpServer.listen(process.env.PORT, () => {
    ...
  })
} catch (err) {
  ...
}
```

https://gitlab.lnu.se/1dv026/content/examples/example-just-task-it-featuring-websocket



This work is licensed under CC BY 4.0

Example

Just Task It featuring webhook and WebSockets



Webhooks – Server to Server



Request with payload

The destinations application's endpoint (webhook URL).

Request URL: POST https://www.example.com/webhooks

Request headers:
Content-Type: application/json

Request body:

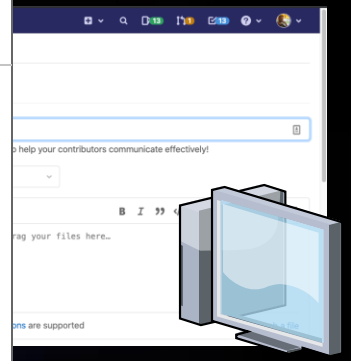
```
{
  "id": "evt_3CfW3G316gk19AbrWp",
  "type": "event",
  "data": {
    …
```

200 Success

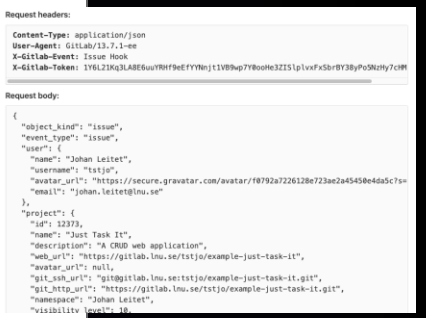
Publisher -
event triggering
(GitLab)

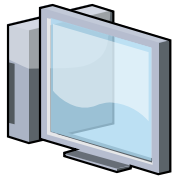
Subscriber -
event processing
(your web application)

Create new issue on GitLab



POST /webhooks



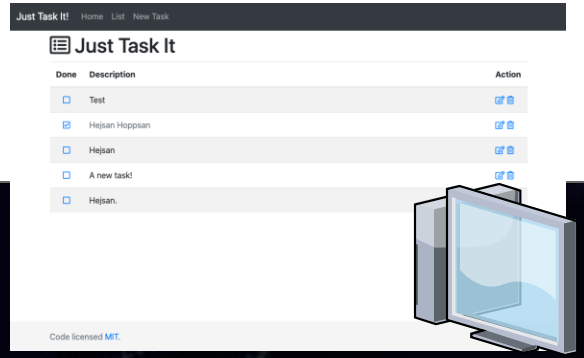




wss, using Socket.IO

Page 1

https://webapp.lnu.se



Client

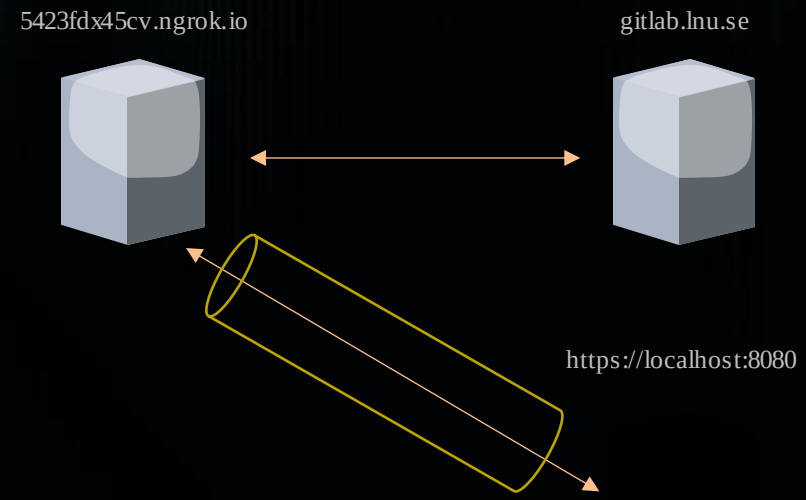
Tunnel a webhook to http://localhost

- ✓ When we develop on localhost, we almost never have a public IP.
- ✓ To receive the hook we need a tool that give us a public IP and tunnels the trafic to localhost
- ✓ **NGROK**, Localtunnel, Serveo, Teleconsole, Pagekite, …
  - `npx ngrok http -bind-tls=true localhost:8080`

https://ngrok.com/

```
Session Status                online
Account                       Ellen Nu (Plan: Free)
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://a9a4-81-225-224-80.ngrok.io -> http://localhost:8080
Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00

HTTP Requests
-------------
POST /webhooks                200 OK
POST /webhooks                200 OK
POST /webhooks                401 Unauthorized
POST /webhooks                200 OK
```



Example, Express – Webhook

1. ```
   // server.js: Enable body parsing of json
   app.use(express.json())
   ```

2. ```
   // route.js: Create a route for the hook(s)
   router.use('/webhooks', webhooksRouter)
   ```

3. Authenticate the webhook payload to ensure that it is legit.

4. Handle the request (add a task to the database).

5. Answer the request!

