FS18

WEB ENGINEERING & DESIGN 3

SPA/PWA

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Admin / Single Page Application (SPA) - In the News

- **Mini Project start 8.3.2019**
  - Introduction, group assignment, etc.
- **23.02.2019: Alternatives to JSX**
  - Templating in React
  - JSX is a domain-specific language
    - Need transformation to JavaScript, babel
  - react-hyperscript
    - Easy of use
  - HTM
    - Similar syntax, but no transformation beforehand (slower)
    - Runtime transformation
  - ijk
    - LISP like, less verbose

```
class A extends React.Component {
    render() {
        return (
            <div className={"class"} onclick={some}>
                {"something"}
                <div>something else</div>
            </div>
        )
    }
}
```

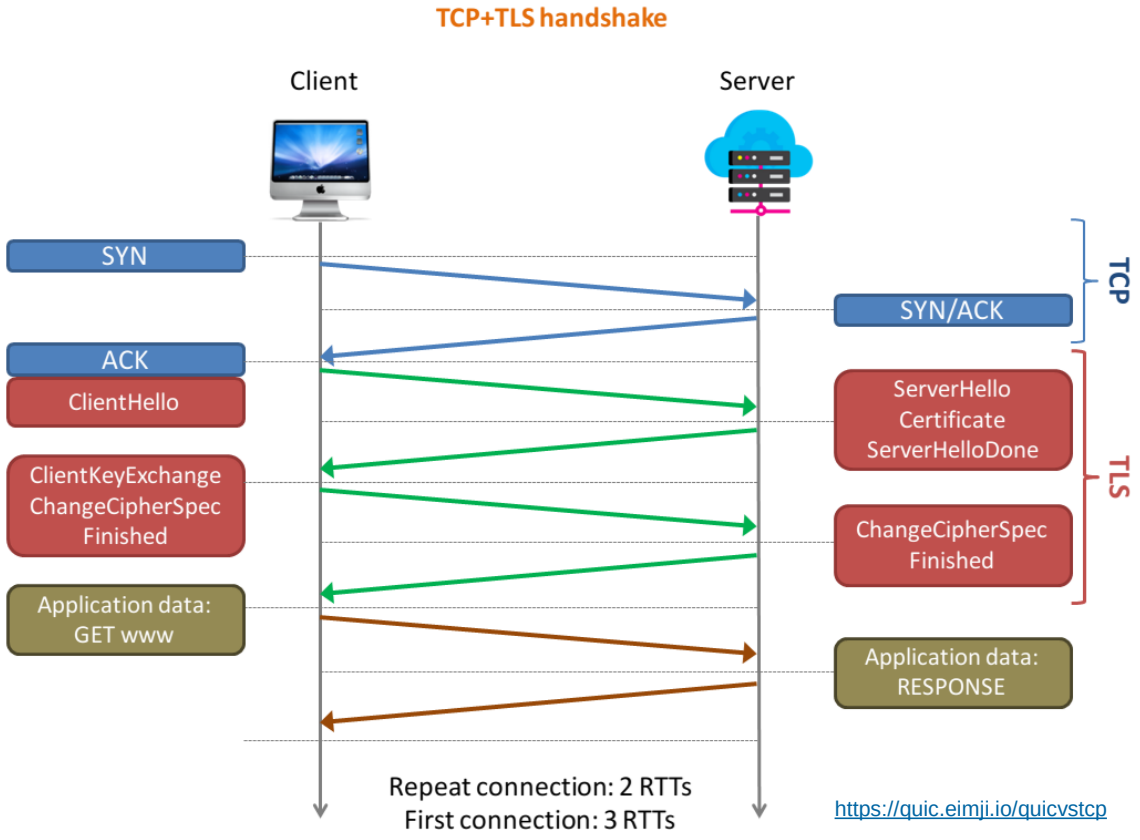
```
class A extends React.Component {
  render() {
    return React.createElement("div", {
      className: "class",
      onclick: some
    }, "something", React.createElement("div", null, "something else"));
  }

}
```

In the News

- **25.02.2019: Go 1.12 is released**
  - Backwards compatible
  - TLS 1.3, will be enabled by default in 1.13
    - 1 RTT instead of 2
    - 0 RTT possible, for previous connections, loosing perfect forward secrecy
  - 67% of browsers used already support it
- **ETS, (eTLS), disable forward secrecy**
  - ETA instead DPI

https://www.thesslstore.com/blog/tls-1-3-handshake-tls-1-2/

- **From DSA, HS18**



https://quic.eimji.io/quicvstcp

In the News

- **27.02.2019: #SBHACK19**
  - Biggest blockchain hackathon in 2019
  - 21. - 23. JUNE 2019
  - Areas
    - Agriculture & Food
    - Finance
    - Intelligent Parcel
    - Mobility
    - Public Service
    - Sports
    - Supply Chain
  - HSR as academic partner

Learning Objectives

The learning objective of this lecture is to acquire knowledge and expertise in single page applications (SPA) and to be able to understand its implementation.

The students are able to...

- **… differentiate between HTTP and WebSocket and its alternatives**
- **… know what server side rendering is**
- **… tell the difference between JavaScript and WebAssembly**
- **… to explain what PWAs are**

Revisiting Exercises

- **Exercise 1 – WebSockets?**
- **WebSocket**
  - Full duplex communication with TCP
  - Text or binary
  - Ports 80/443, compatible with HTTP, uses upgrade header to change from HTTP to WebSocket
  - HTTP request/reply
    - WebSocket does not need request
  - wss://localhost…
- **Connection Establishment**
  - Sec-WebSocket-Key: base64 rnd
  - Sec-WebSocket-Accept: base64 hash(rnd) – against caching

```
let connection = new WebSocket('wss://localhost:8080/api');
connection.binaryType = "arraybuffer";
```

```
GET ws://localhost:8080/ws HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: tvdBI26pvLeIgFiDC3H5Sg==
Sec-WebSocket-Version: 13
Origin: http://localhost:8080
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: Myz76wyg5WStKRkpSCKhTt/tWAM=
```

WebSockets

- **Reverse Proxy?**
  - Nginx / Apache
  - Proxy need to set header again, otherwise lost
  - Connection timeout
- **Go implementation**
  - Basic: x/net
  - Advanced: github.com/gorilla/websocket
- **Example:**

```
func ws(w http.ResponseWriter, r *http.Request, _
httprouter.Params) {
          conn, _ := upgrader.Upgrade(w, r, nil)
          messageType, _, _ := conn.NextReader()
          for {
                    res, _ :=
http.Get("https://api.coinmarketcap.com/v1/ticker/?limit=10")
                    w, _ := conn.NextWriter(messageType)
                    io.Copy(w, res.Body)
                    time.Sleep(2 * time.Second)
          }
}
```

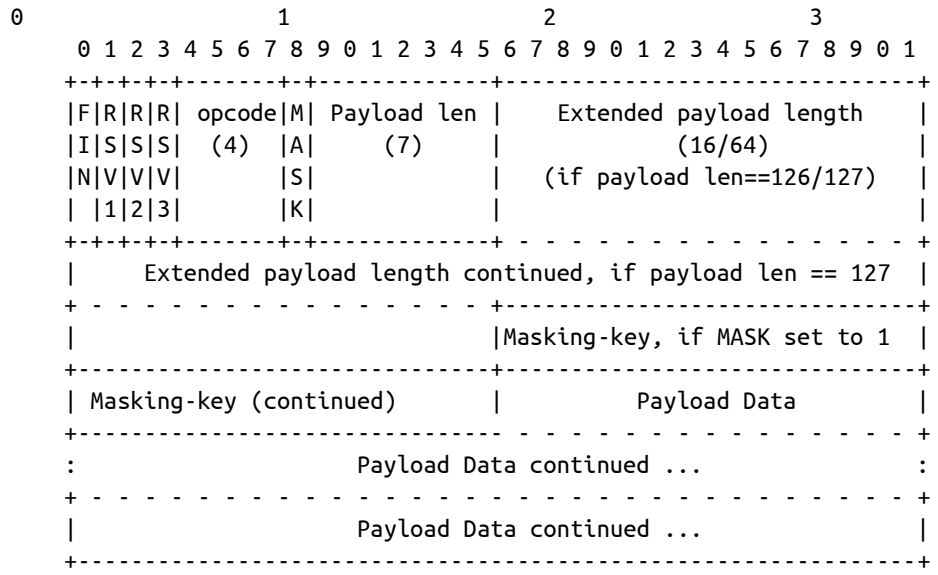
```
location /chat/ {
    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_read_timeout 60s;
}
```

```
mod_proxy_wstunnel:

ProxyPass "/ws2/"  "ws://echo.websocket.org/"
ProxyPass "/wss2/" "wss://echo.websocket.org/"
ProxyWebsocketIdleTimeout 60
```

```
var connection = new WebSocket('ws://localhost:8080/ws');
    connection.onopen = function () {
        connection.send('Ping');
    };
    let counter = 0;
    connection.onmessage = function (e) {
        console.log('update: ' + counter++);
        let string = 'Markets:<br>';
        for(let s of JSON.parse(e.data)) {
            string += s.name + ', price=' + s.price_usd + '<br>';
        }
        let element = document.getElementById('text');
        element.innerHTML = string;
    };
```

WebSocket Protocol

- **Protocol**
  - Client Handshake Request
    - Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
    - Sec-WebSocket-Version: 13, since 2011
  - Server Handshake Response
    - Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
  - If mask set, XOR “encryption”
    - “By having the browser generate a random mask for each frame, the hostile client code cannot choose the byte patterns that appear on the wire and use that to attack vulnerable network infrastructure. (source)”
    - “treating WebSockets data as a cacheable HTTP request. (source)”
    - prevent proxy cache poisoning attacks
  - wss://
    - “but until certificates are free, this would make any web standard requiring SSL/TLS a "rich man's" standard rather than an internet wide solution.”



- **FIN – last message**
- **Opcode**
  - 0: continue, 1: text, 2: binary

WebSockets

- **There is always a blog with the title: “Do you really need xy?”**
- **29.3.2018: Do you really need WebSockets?**
  - WebSockets vs. Server-Sent events / EventSource
  - Advantages of SSE over Websockets:
    - Transported over simple HTTP instead of a custom protocol
    - Can be poly-filled with javascript to "backport" SSE to browsers that do not support it yet.
    - Built in support for re-connection

	WebSockets	Server Sent Ev.	Long Polling
Parallel conn.	1024	~6	~6
Support	92%	88% (no Edge)	100%

- Advantages of Websockets over SSE:
  - Real time, two directional communication.
  - Native support in more browsers
- Ideal use cases of SSE:
  - Stock ticker streaming
  - Twitter feed updating
  - Notifications to browser
- SSE gotchas:
  - No binary support
  - Maximum open connections limit
- SSE: 88%, WebSocket: 92%

Long Polling

- No data is available, server waits with its response, client waits

URL: b.socrative.com/student/

Room: HSR

Server Side Request

- **Exercise 1 – How does server side rendering work?**
  - Prerendering, create static HTML
  - SSR, create HTML on the fly
  - Example with vue.js
  - Need a second server that understands JS
    - node.js
- **Events do not work anymore in our example**
  - Button press needs to go to the server again
- **Google, AJAX…**
  - Official website: “Today, as long as you're not blocking Googlebot from crawling your JavaScript or CSS files, we are generally able to render and understand your web pages like modern browsers. ”

```
const Vue = require('vue')
const server = require('express')()
const renderer = require('vue-server-
renderer').createRenderer()

server.get('*', (req, res) => {
  const app = new Vue({
    data: {
      url: req.url
    },
    template: `<div>The visited URL is: {{ url }}</div>`
  })

  renderer.renderToString(app, (err, html) => {
    if (err) {
      res.status(500).end('Internal Server Error')
      return
    }
    res.end(`
      <!DOCTYPE html>
      <html lang="en">
        <head><title>Hello</title></head>
        <body>${html}</body>
      </html>
    `)
  })
})

server.listen(8080)
```

URL: b.socrative.com/student/

Room: HSR

WebAssembly

- **Exercise 1 – WebAssembly replaces JavaScript?**
  - No (not all of it)
- **Efficient binary instruction format for a stack-based virtual machine**
  - C/C++/Rust on the web client
  - Execute at native speed, C++ example:

```
int squarer(int num) {return num * num;}
```

- **Rust**
  - Install: curl https://sh.rustup.rs -sSf | sh
  - Add $HOME/.cargo/bin to you path
  - cargo install wasm-pack
  - wasm-pack build
  - Now we have our wasm, execute it in browser

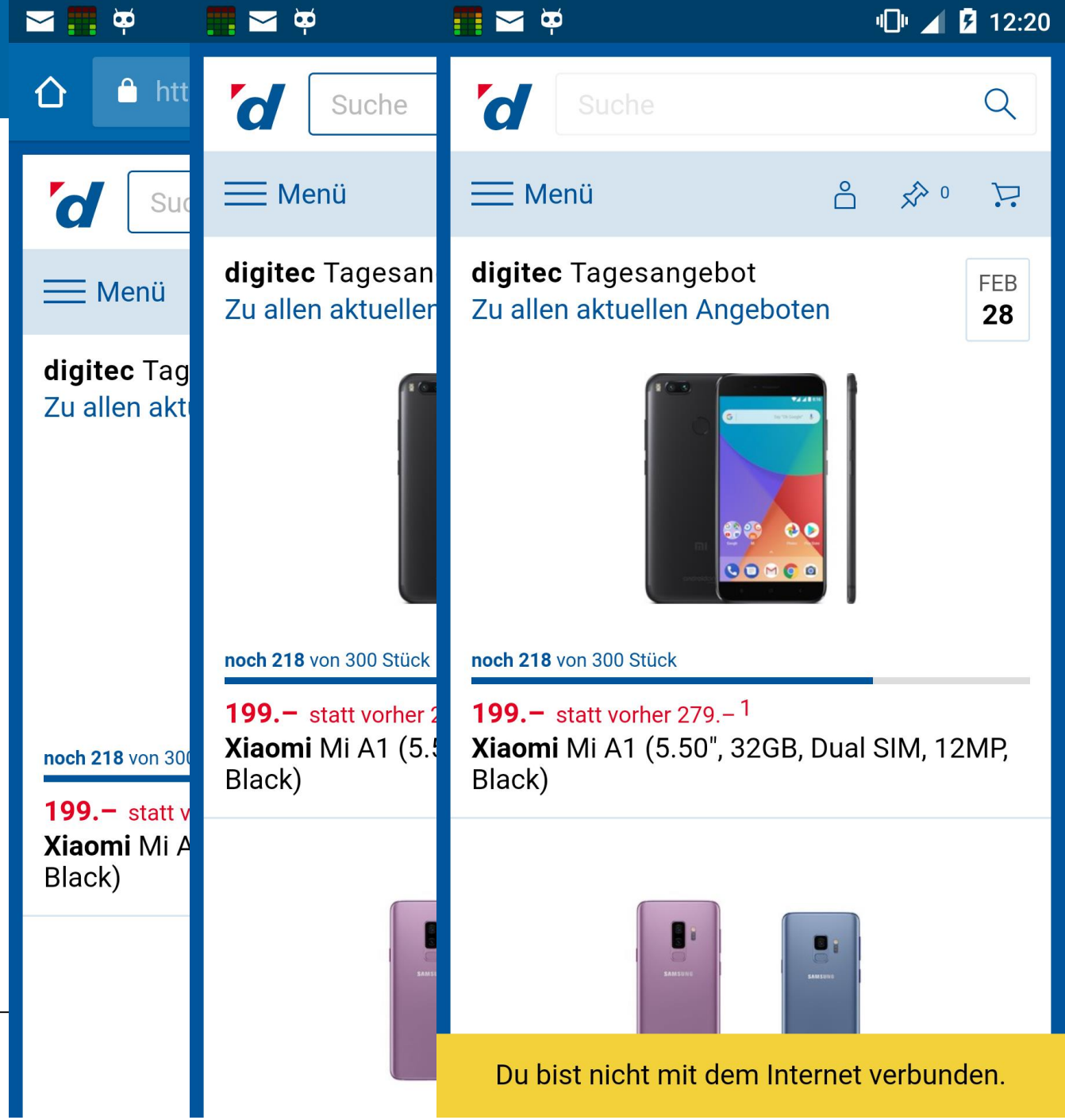
```
[package]
name = "rust-wasm"
version = "0.0.1"
authors = ["Thomas Bocek"]

[lib]
crate-type = ["cdylib"]

[dependencies]
wasm-bindgen = "0.2"
```

```
//src/lib.rs
#[wasm_bindgen]
pub fn mul(x: i32, y: i32) -> i32 {
    x * y
}
```

```
WebAssembly.instantiateStreaming(fetch("wasm"))
        .then(wasmModule => {
          const result = wasmModule.instance.exports.mul(3, 4);
          const text = document.createTextNode(result);
          document.body.appendChild(text);
        });
```

PWA – Progressive Web Apps

- **“Progressive web apps (PWAs) are web applications that are regular web pages or websites, but can appear to the user like traditional applications or native mobile applications…” – Wikipedia**
- **To be a Progressive Web App, a site must:**
  - Originate from a Secure Origin, TLS.
  - Reference a Web App Manifest with at least the following properties:
    - name
    - short_name
    - start_url
    - display with a value of standalone or fullscreen
    - An icon at least 144×144 large in png format.



PWA – Progressive Web Apps

- **To be a Progressive Web App, a site must:**
  - Load while offline (even if only a custom offline page). This means that Progressive Web Apps require Service Workers. (Mozilla)
    - Service workers: scriptable network proxy in the web browser to manage the web/HTTP requests programmatically
    - Runs in worker context: has no DOM access, no localstorage, used for Assets
- **Advantages**
  - Better User Experience
  - One codebase + language for mobile+desktop
- **Disadvantages**
  - Yet another standard ☺
  - Hardware access depends on standards

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('service-worker.js');
}
```

```
self.addEventListener('fetch', function (event) {
    if (event.request.method === 'GET') {
        console.log('Handling fetch event for',
event.request.url);
        event.respondWith(
            fetch(event.request).catch(error => {
                console.log('Fetch failed; returning
offline page instead.', error);
                return caches.match('offline.html');
            })
        );
    }
});
```

URL: b.socrative.com/student/

Room: HSR

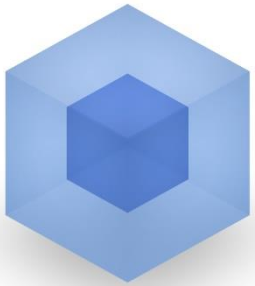
Bundler / Webpack

- **Bundler**
  - “A JavaScript bundler is a tool that puts your code and all its dependencies together in one JavaScript file”
  - Dependencies (package.json) Install yarn (or npm)
    - yarn install
    - Dev dependencies: e.g., css-loader, vue-loader, vue-template-compiler, webpack,webpack-cl, webpack-dev-server
  - Webpack (webpack.config.js)
    - Entry: which files to start the bundling process
    - Output: where (and how) the generated bundle are stored, using defaults
    - Modules, rules, loaders: packages that transform files matching a pattern (plugins)

```
'use strict'

const { VueLoaderPlugin } = require('vue-loader')

module.exports = {
  mode: 'development',
  entry: [
    './src/app.js'
  ],
  module: {
    rules: [
      {
        test: /\.vue$/,
        use: 'vue-loader'
      }
    ]
  },
  plugins: [
    new VueLoaderPlugin()
  ]
}
```

Bundler / Webpack

- **How to create (split) html/js/css files?**
- **Bundler!**
  - In the JavaScript world: understand the concepts not the tools
  - E.g., Spesen Project as Master project started (task runner - minify, uglify, linting, compiling)
  - May 2015: Grunt for project (supervision, state-of-the-art)
  - Dec 2015: Gulp for pdfsign.js (done by me)
  - … Browserify (missed that one :)
  - June 2016: Webpack for modum.io (done by me + Vue.js)
  - January 2017: Webpack 2 for modum.io (supervision)
  - March 2018: Still Webpack 2? Rollup?
  - FS19: Webpack 4, Rollup, or Parcel



URL: b.socrative.com/student/

Room: HSR

Questions?

Prof. Dr. Thomas Bocek
thomas.bocek@hsr.ch