Tampereen yliopisto
Tampere University

COMP.CS.510 Web Development 2 - Architecting

SPA - Single-page applications

David Hästbacka

Content

- Single-page applications

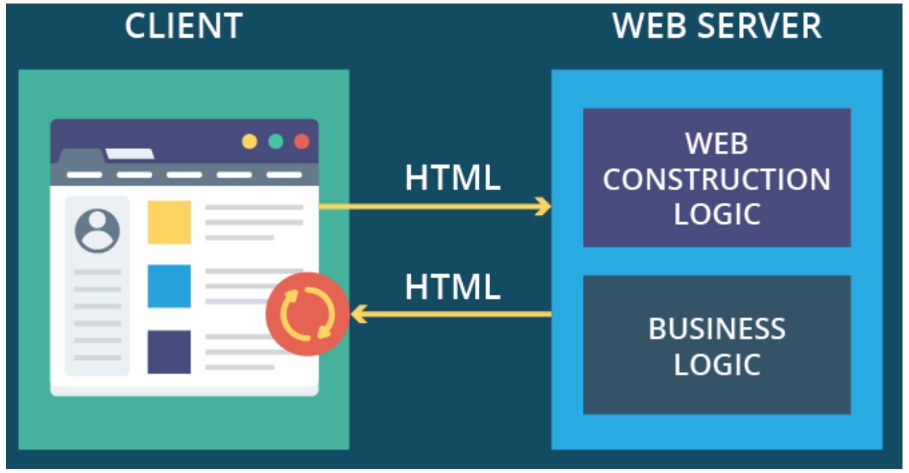
Tampereen yliopisto
Tampere University

Course topics outline

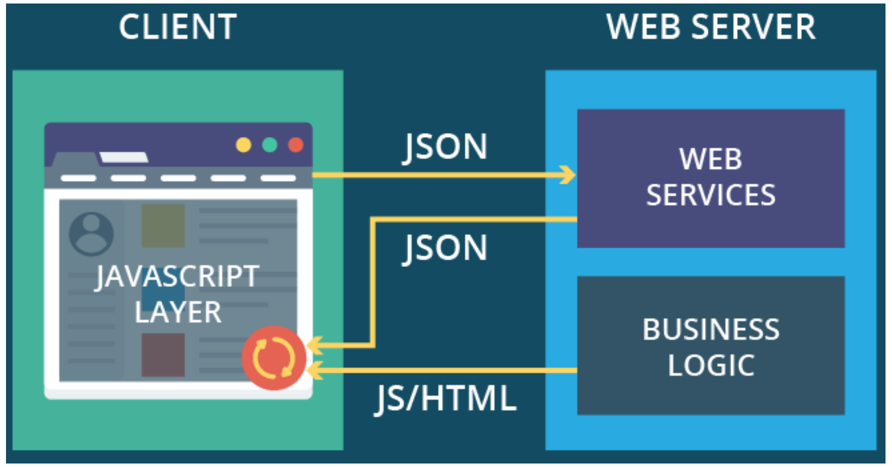
- Web as a platform and means of communication, web application components
- **Single-page Applications, software and systems architecture considerations**
- Web Services (REST, SOAP, …)
- Service Oriented Architecture, API ecosystems, service composition
- Other web based communication (web sockets, asynchronous communication, message buses, DDS, OPC UA, …)
- Microservices, serverless computing and FaaS, event-driven scalability (implementing back end logic)
- Information security and ensuring Quality of Service in distributed web applications
- Data and semantics for interoperable web applications (beyond JSON and XML, information models, …)
- Applications of distributed web applications and as the means of communication
    - IoT, Industrial Internet, Industry4.0, Smart Cities …

Traditional web applications vs Single-page applications

- Traditional HTML output
  - The server outputs HTML shown in the browser, typically the entire page
  - Increases data exchange, server work load, latencies, …



- SPA
  - UI Client
  - The client has a JavaScript layer that communicates with the server
  - Fast, efficient, reduces server work load but increases client (browser) work load

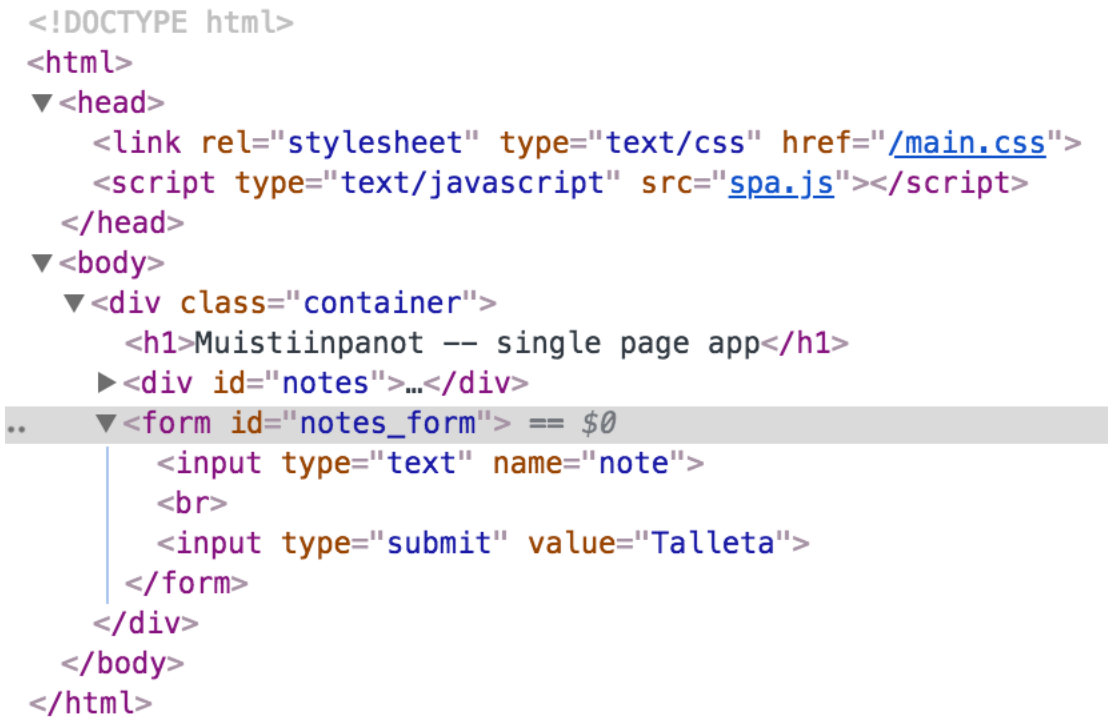


Tampereen yliopisto
Tampere University

Based on: mluukkai@cs.helsinki.fi
https://fullstackopen.github.io
Creative Commons BY-NC-SA 3.0

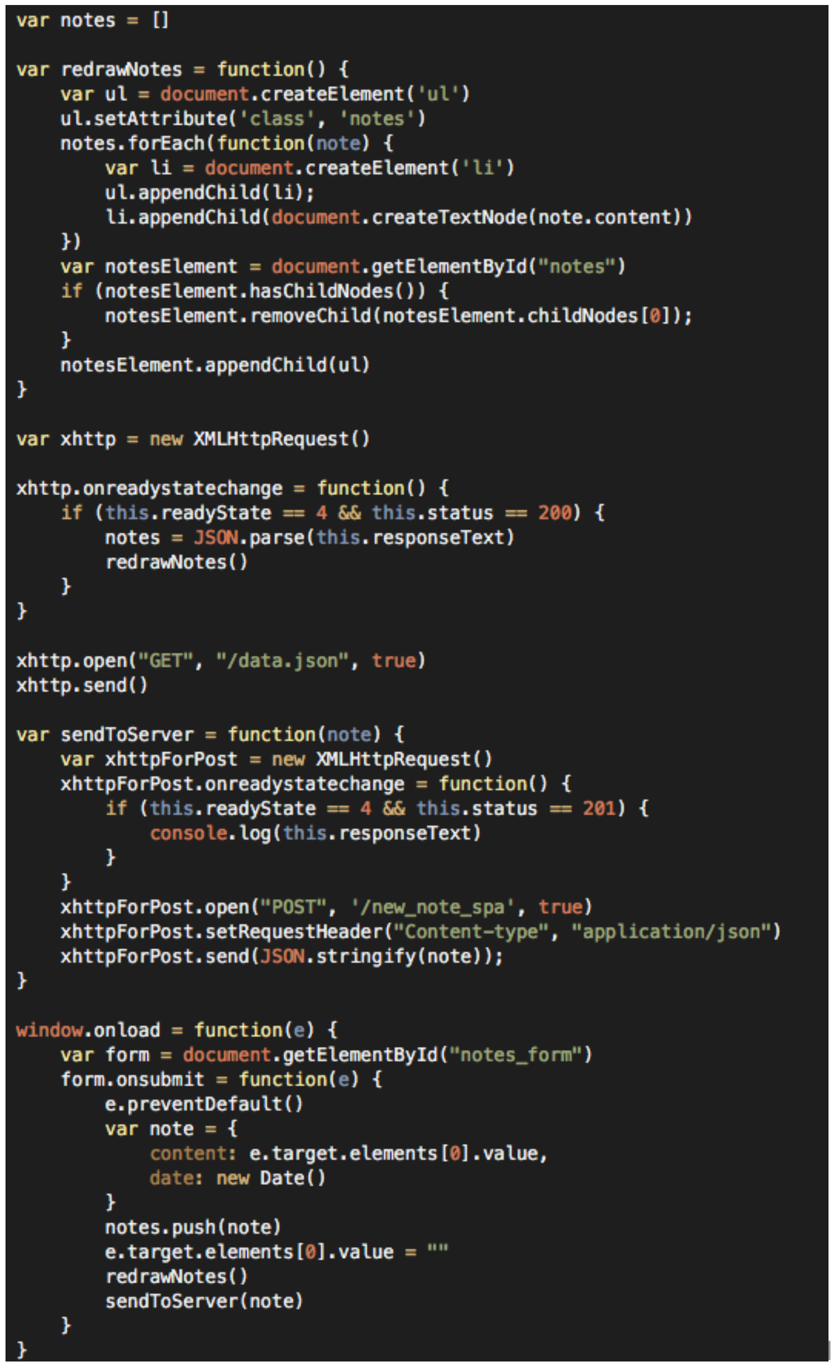
SPA

- Example: https://fullstack-exampleapp.herokuapp.com/spa



```
<!DOCTYPE html>
<html>
▼<head>
    <link rel="stylesheet" type="text/css" href="/main.css">
    <script type="text/javascript" src="spa.js"></script>
  </head>
▼<body>
  ▼<div class="container">
      <h1>Muistiinpanot -- single page app</h1>
    ▶<div id="notes">…</div>
..  ▼<form id="notes_form"> == $0
        <input type="text" name="note">
        <br>
        <input type="submit" value="Talleta">
      </form>
    </div>
  </body>
</html>
```

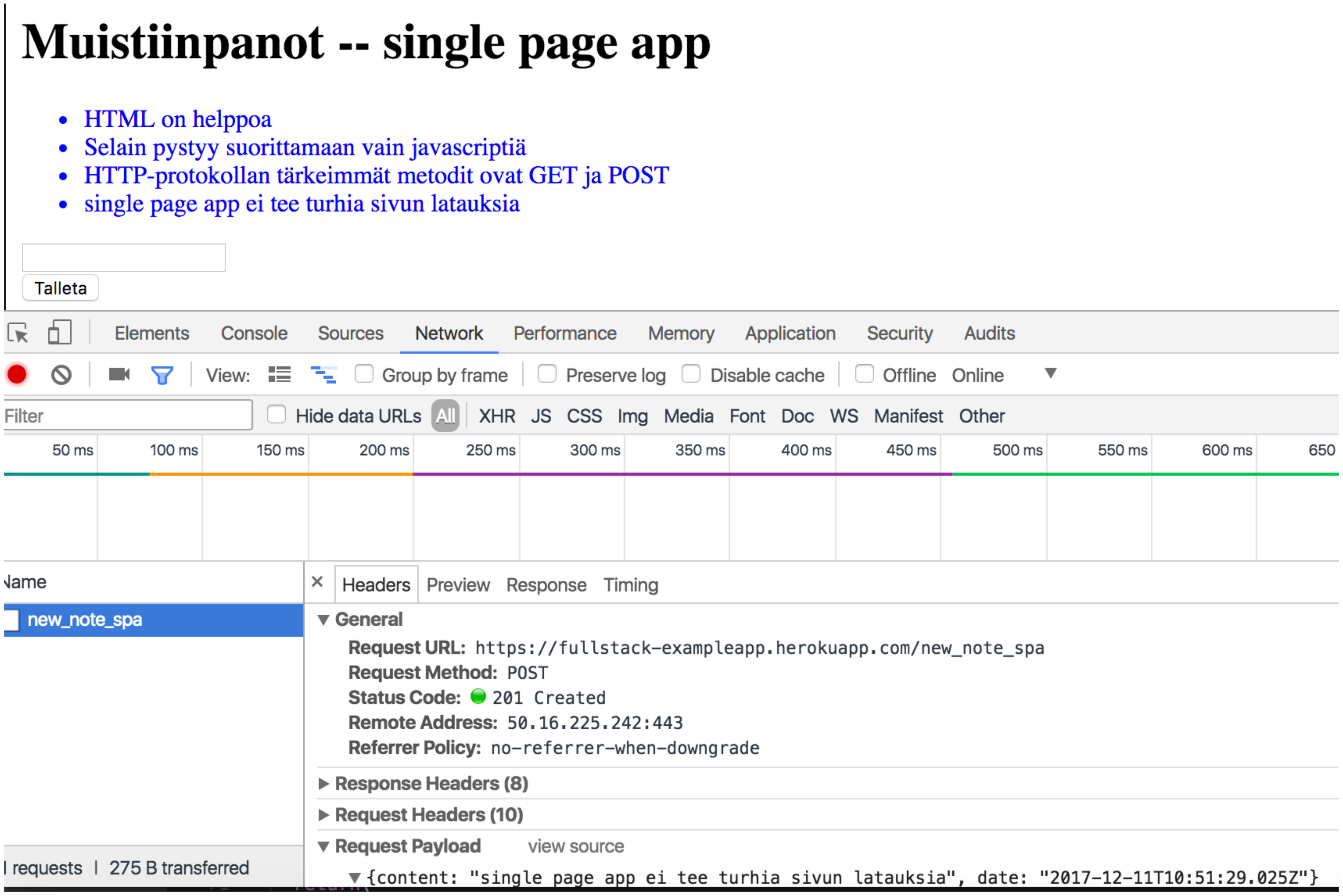
- Notice spa.js



```
var notes = []

var redrawNotes = function() {
    var ul = document.createElement('ul')
    ul.setAttribute('class', 'notes')
    notes.forEach(function(note) {
        var li = document.createElement('li')
        ul.appendChild(li);
        li.appendChild(document.createTextNode(note.content))
    })
    var notesElement = document.getElementById("notes")
    if (notesElement.hasChildNodes()) {
        notesElement.removeChild(notesElement.childNodes[0]);
    }
    notesElement.appendChild(ul)
}

var xhttp = new XMLHttpRequest()

xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        notes = JSON.parse(this.responseText)
        redrawNotes()
    }
}

xhttp.open("GET", "/data.json", true)
xhttp.send()

var sendToServer = function(note) {
    var xhttpForPost = new XMLHttpRequest()
    xhttpForPost.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 201) {
            console.log(this.responseText)
        }
    }
    xhttpForPost.open("POST", '/new_note_spa', true)
    xhttpForPost.setRequestHeader("Content-type", "application/json")
    xhttpForPost.send(JSON.stringify(note));
}

window.onload = function(e) {
    var form = document.getElementById("notes_form")
    form.onsubmit = function(e) {
        e.preventDefault()
        var note = {
            content: e.target.elements[0].value,
            date: new Date()
        }
        notes.push(note)
        e.target.elements[0].value = ""
        redrawNotes()
        sendToServer(note)
    }
}
```

SPA

- Sending the form does not reload the full page - only a POST request is sent from the JavaScript layer



Muistiinpanot -- single page app

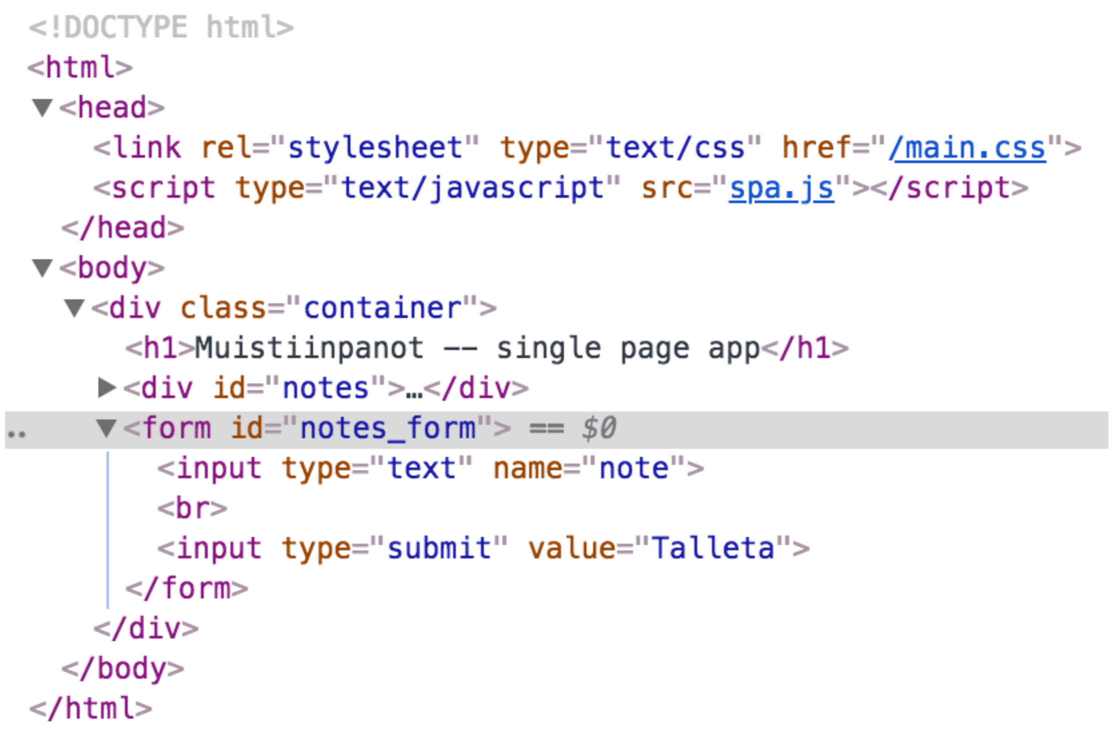
- HTML on helppoa
- Selain pystyy suorittamaan vain javascriptiä
- HTTP-protokollan tärkeimmät metodit ovat GET ja POST
- single page app ei tee turhia sivun latauksia

Talleta
Elements Console Sources Network Performance Memory Application Security Audits
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
50 ms 100 ms 150 ms 200 ms 250 ms 300 ms 350 ms 400 ms 450 ms 500 ms 550 ms 600 ms 650
Name
new_note_spa
requests | 275 B transferred
Headers Preview Response Timing

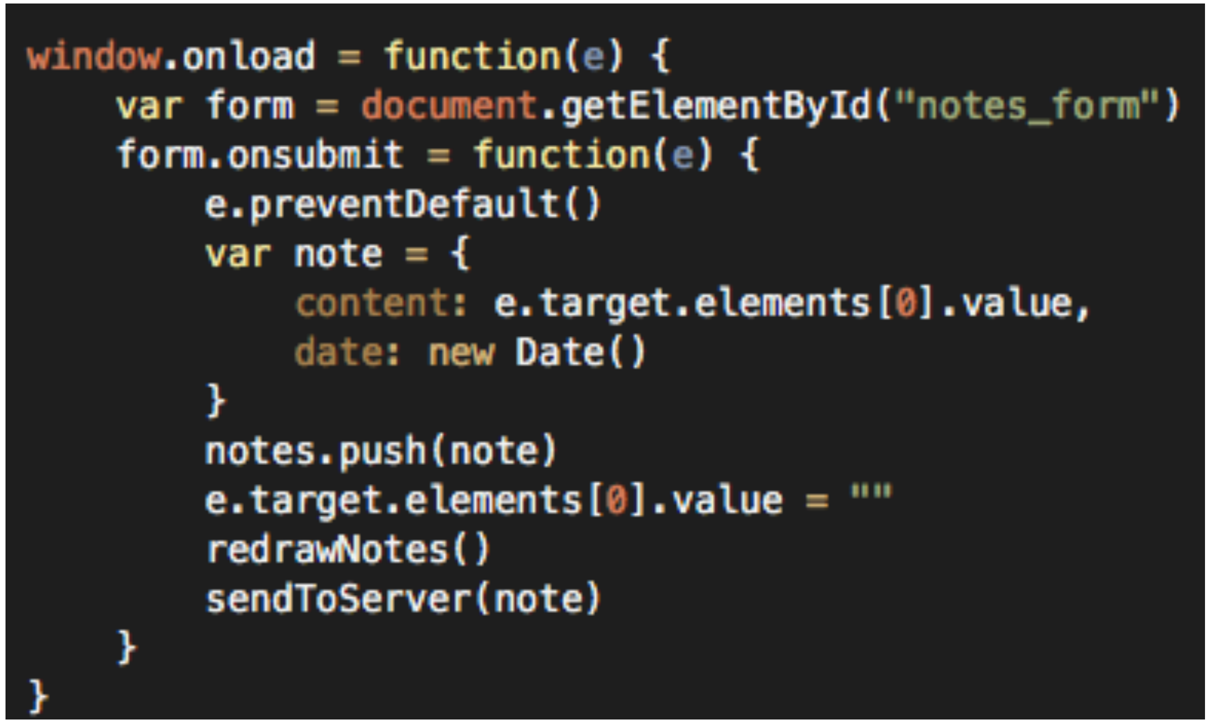
General
Request URL: https://fullstack-exampleapp.herokuapp.com/new_note_spa
Request Method: POST
Status Code: ● 201 Created
Remote Address: 50.16.225.242:443
Referrer Policy: no-referrer-when-downgrade
Response Headers (8)
Request Headers (10)
Request Payload view source

```
{content: "single page app ei tee turhia sivun latauksia", date: "2017-12-11T10:51:29.025Z"}
```

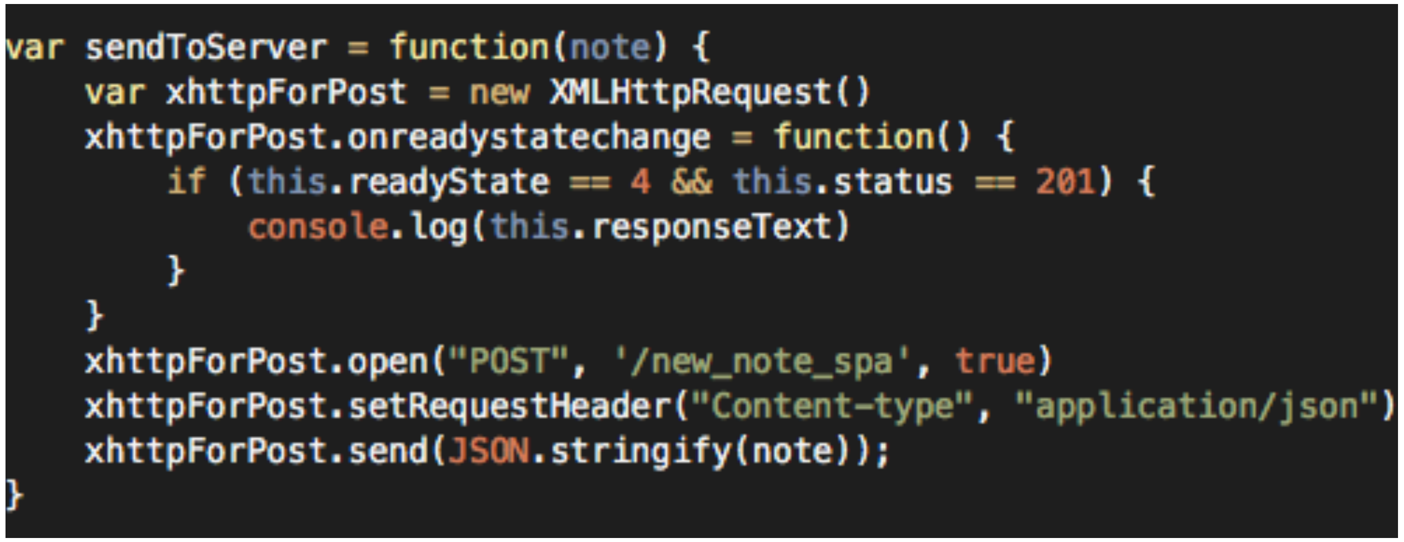
Tampereen yliopisto
Tampere University



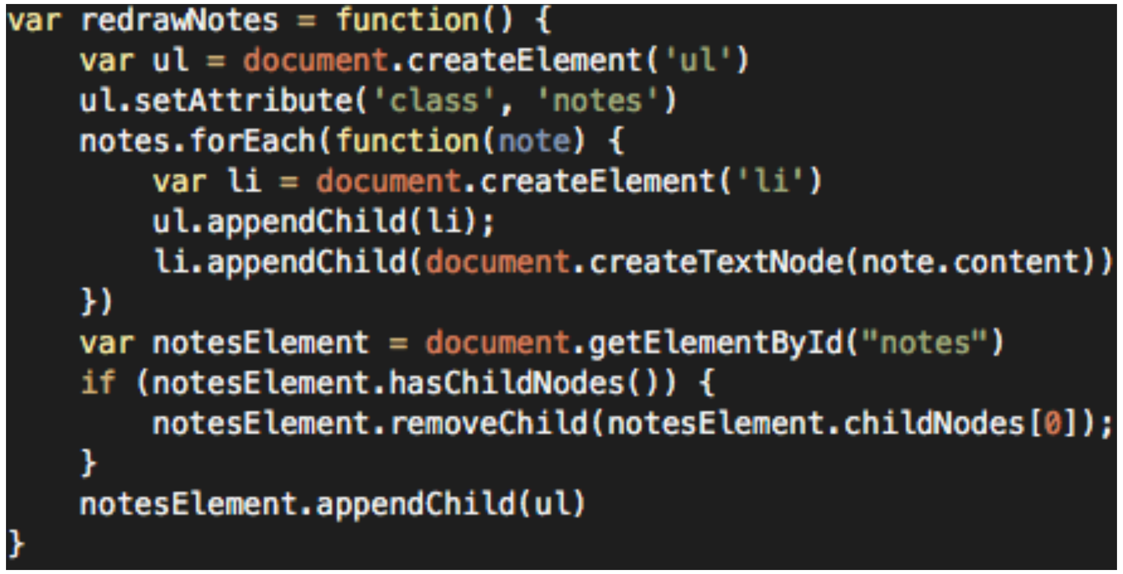
```
<!DOCTYPE html>
<html>
▼<head>
   <link rel="stylesheet" type="text/css" href="/main.css">
   <script type="text/javascript" src="spa.js"></script>
 </head>
▼<body>
  ▼<div class="container">
     <h1>Muistiinpanot -- single page app</h1>
    ▶<div id="notes">…</div>
..  ▼<form id="notes_form"> == $0
       <input type="text" name="note">
       <br>
       <input type="submit" value="Talleta">
     </form>
   </div>
 </body>
</html>
```



```
window.onload = function(e) {
    var form = document.getElementById("notes_form")
    form.onsubmit = function(e) {
        e.preventDefault()
        var note = {
            content: e.target.elements[0].value,
            date: new Date()
        }
        notes.push(note)
        e.target.elements[0].value = ""
        redrawNotes()
        sendToServer(note)
    }
}
```



```
var sendToServer = function(note) {
    var xhttpForPost = new XMLHttpRequest()
    xhttpForPost.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 201) {
            console.log(this.responseText)
        }
    }
    xhttpForPost.open("POST", '/new_note_spa', true)
    xhttpForPost.setRequestHeader("Content-type", "application/json")
    xhttpForPost.send(JSON.stringify(note));
}
```



```
var redrawNotes = function() {
    var ul = document.createElement('ul')
    ul.setAttribute('class', 'notes')
    notes.forEach(function(note) {
        var li = document.createElement('li')
        ul.appendChild(li);
        li.appendChild(document.createTextNode(note.content))
    })
    var notesElement = document.getElementById("notes")
    if (notesElement.hasChildNodes()) {
        notesElement.removeChild(notesElement.childNodes[0]);
    }
    notesElement.appendChild(ul)
}
```

Manipulating the DOM

- In the previous example the browser document object model was modified manually
- JS libraries such as Jquery, BackboneJS, AngularJS, React …
- Example in JQuery:

```
$( "button.continue" ).html( "Next Step..." )
```

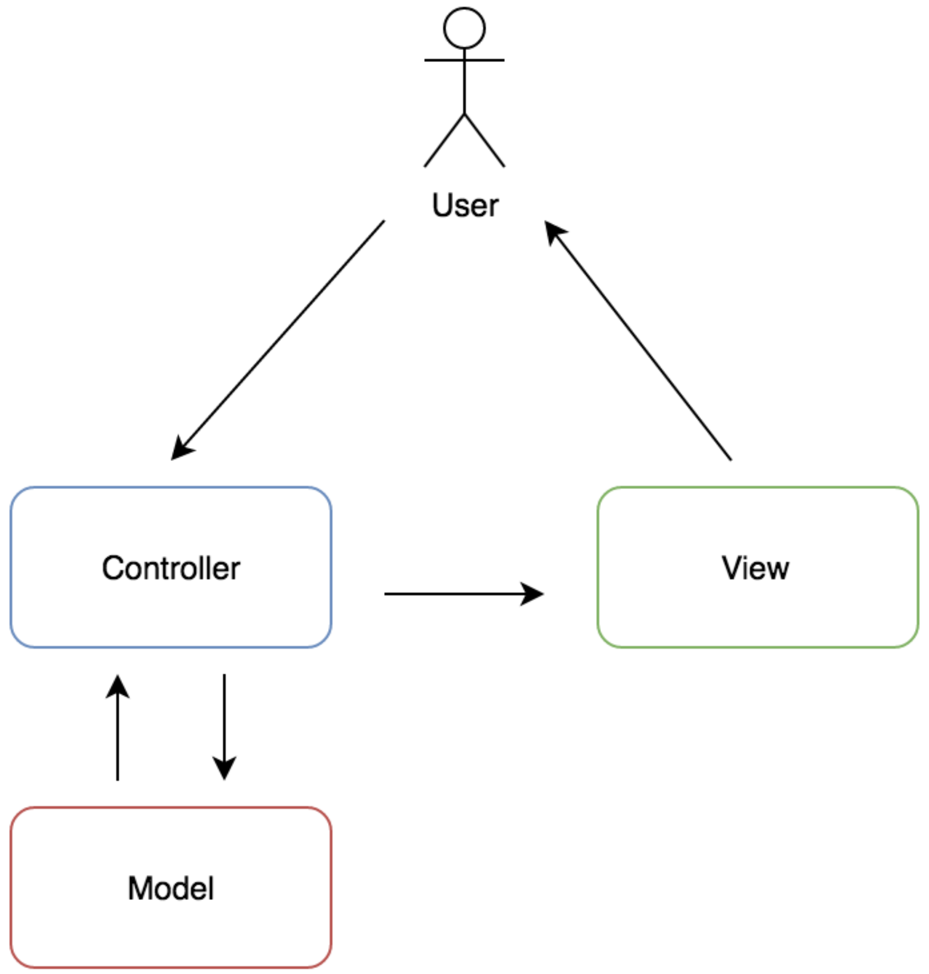
```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event ) {
  hiddenBox.show();
});
```

```
$.ajax({
  url: "/api/getWeather",
  data: {
    zipcode: 97201
  },
  success: function( result ) {
    $( "#weather-temp" ).html( "<strong>" + result + "</strong> degrees" );
  }
});
```

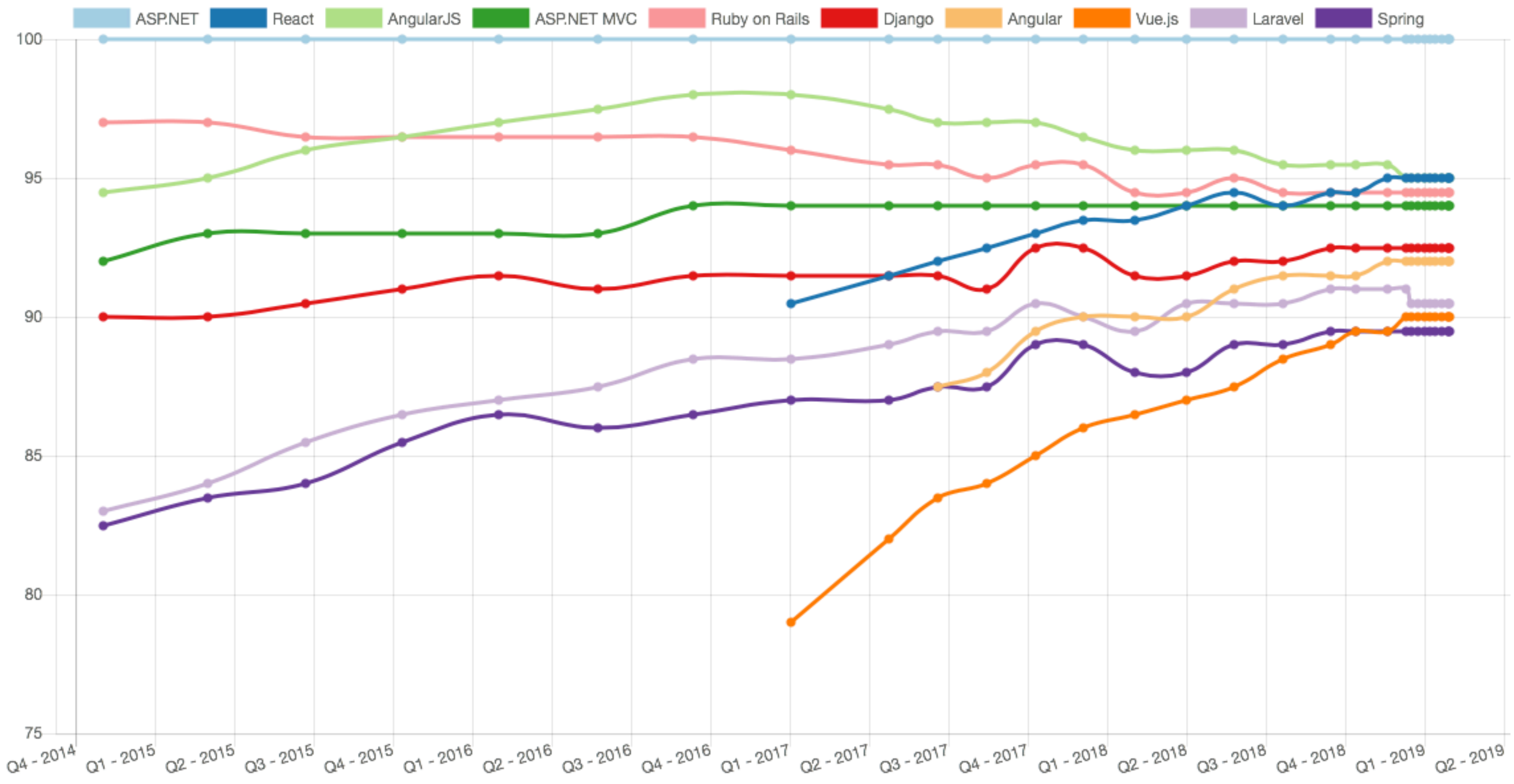
Based on: Daniel Deutsch
https://medium.com/of-all-things-tech-progress/understanding-mvc-architecture-with-react-6cd38e91fefd

MVC

- Model-view-controller -pattern - a way to structure your application
- Pros
  - better and easier code maintenance and reusability
  - easier to coordinate in teams due to the separation
  - ability to provide multiple views
  - support for asynchronous implementations
- Cons
  - an increased complex setup process
  - dependencies, i.e. changes in the model or controller affect the whole entity



Frameworks
(Source: https://hotframeworks.com, now discontinued)



Based on: Daniel Deutsch
https://medium.com/of-all-things-tech-progress/understanding-mvc-architecture-with-react-6cd38e91fefd

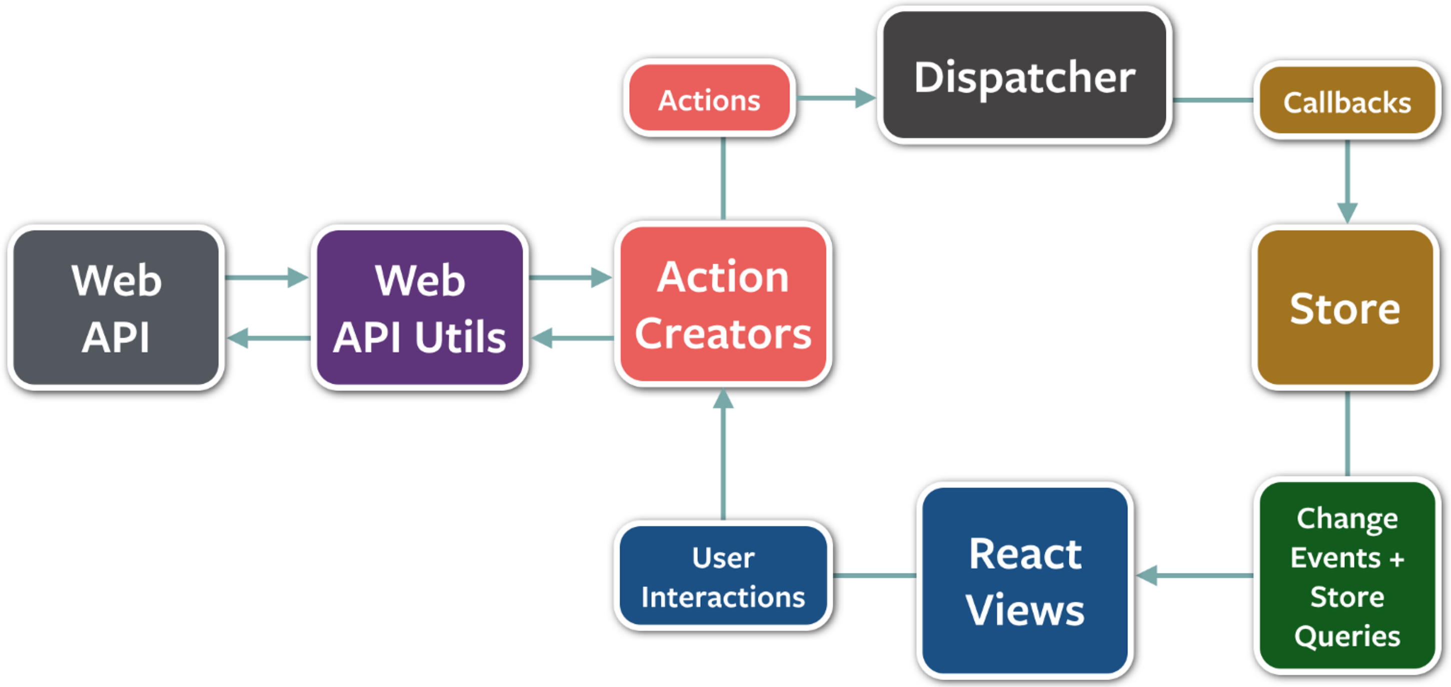
React

- A library created by Facebook
- Features
  - declarative: Design different views for each state, which will be efficiently updated and re-rendered
  - component-based: Build components, that manage their own state and structure them together into more complex UIs
  - maintains an internal representation of the rendered UI (“virtual DOM”), that renders only the changed elements

Based on: Daniel Deutsch
https://medium.com/of-all-things-tech-progress/understanding-mvc-architecture-with-react-6cd38e91fefd

React, Flux

- Flux to remove the need of bidirectional communication and reduce cascading effects
  - **Actions**
    - Objects with property and data.
  - **Stores**
    - Contain the application's state and logic.
  - **The Dispatcher**
    - Processes registered actions and callbacks.
  - **Views**
    - Listen to changes from the stores and re-render themselves.



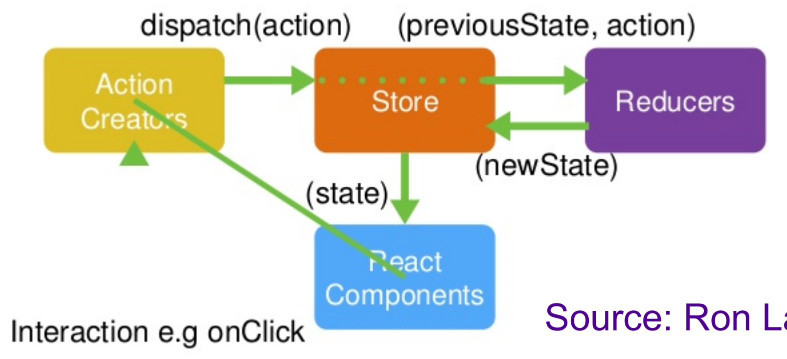
Compared to typical MVC
- Unidirectional instead of bidirectional!
- Stores are able to store any application related state, whereas the model in MVC was designed to store single objects the initiating point.
- Dispatcher makes debugging much easier

Based on: Daniel Deutsch
https://medium.com/of-all-things-tech-progress/understanding-mvc-architecture-with-react-6cd38e91fefd

Redux

- Builds on Flux with following principles
  - Only one single source of truth
  - The state of your entire application is stored in a single store.
  - State is read-only
  - The only way to change the state is to emit an action (an object describing what happened).
  - Changes are made with pure functions
  - Specify the transformation by actions with reducers, which allow to navigate through states.

Redux Flow



Source: Ron Lavit

- Compared to plain Flux
  - Redux does not have the concept of a *dispatcher* because it relies on pure functions instead of event emitters.
  - Redux assumes you never mutate your data. You don't mutate them in a reducer but rather return a new object