

Todo

- ☐ Go Deep
- ☐ Lue alkuperäinen dokumentaatio: (React, Redux, Node, Express, Mongo + HTTP requests, Mozilla JS, W3Schools)
- ☐ Tee template sovellukset eri tekniikoilla
- ☐ NoSql data modelling
- ☐ Juho Vepsäläinen, SurviveJS React
- ☐ You Don't Know JS (book series) - <https://github.com/getify/You-Dont-Know-JS>
- ☐ Chrome devtools
- ☐
- ☐ Kursseja: <https://mooc.fi/>
- ☐ Linkkejä: <https://fullstack-hy.github.io/linkit/>
- ☐
- ☐ Tietoturva
- ☐ AI: TensorFlow, Firebase + ML Kit + Big Query
- ☐ Digital Reality: AR, VR, Digital Twins

Web sovelluksen osa-alueet

Components
Debugging
Styling
HTTP requests
Routing
Form validation
Authentication
Testing
Deployment

Työkaluja

Visual Studio Code

- VSCoDe extensions
 - Simple React Snippets
 - Prettier code formatter

Codepen <https://codepen.io>
Babel <https://babeljs.io/>
jshint työkalulla voi tutkia javascript koodia

API

Lista maista: <https://restcountries.eu/#api-endpoints-all> <https://restcountries.eu/rest/v2/all>
Palauttaa random 200x200 kuvan: <https://picsum.photos/200>

2019 Trends

Node JS + Basic Web development
GraphQL
Testing (headless)
Security
Frontend Frameworks
Web Components (=custom HTML elements): Stencil
JAM stack + static web pages
Hybrid Apps (native=kaikille alustoille): Ionic

Serverless:	AWS Lambda, Azure, Google Cloud
AI/Bots	

2018 Trends

Serverless	AWS Lambda
Progressive Web Apps	Mobile App Experiences on Web pages
Web components	Stencil
Static Web Pages	Jekyll, Hugo, NextJS

Full-Stack

Web

HTML5, CSS3 (Flexbox, CSS grid, W3.CSS)
JavaScript + TypeScript, React&Context&State / React&Redux / Angular&NgRx
PWA + Web components (Polymer component library)

HTTPS Integration

GraphQL(+Apollo), REST API (JSON)

Backend

Node.js, Express.js / Spring boot

Data Base

MongoDB / MySQL

DevOps

Npm + Webpack + Git
Containers + Microservices
Docker + Kubernetes + Istio
Security

Hosting

AWS Lambda / Zeit Now / Open Shift

Low Code

Outsystems, Google App Builder

Trends

Consumers use mobile devices instead of computers
Responsiveness and Progressive Web Apps (PWA)
Single Page Applications (SPA)
Microservices
Event driven programming, Event sourcing
DevOps, Continuous Integration (CI) and Deployment (CD)
SAFe

WEB

HTML5, (ES6), CSS3, SSL Certificates
CSS Frameworks: Twitter Bootstrap, Semantic ui, MaterialUI, Metro(Facebook), Zurb Foundation, W3.CSS, Materialize CSS, Bulma, ant.design
CSS utility Frameworks: Tailwind CSS
Template Engines: Pug, EJS, Mustache, Dust
CSS Pre-Processor: Sass, Less
Web APIs: REST, JSON, XHR(AJAX), SOAP, GraphQL (Facebook)
HTTP Request libraries: Fetch API, Axios, Superagent, Prototype, jQuery, Node HTTP
Static Web Pages: Jekyll, Hugo, Naxjs
Web component libraries: Polymer, Stencil,
State management: Redux, NgRx (Angular), Vuex (Vue), MobX, Flux (vanha)
Integration of Web server and Backend application: CGI (Common Gateway Interface)
Graphics: SVG, WebGL
Desktop: Electron(JS)
AI: Python, TensorFlow.js, Brain.js

Languages

Frontend languages: JavaScript, TypeScript (Microsoft), CoffeeScript
Backend languages: JavaScript, Python, PHP, Ruby, C#, Cold Fusion, Java
High performance languages: Go (Google), Rust,
Other languages: Haskell, Kotlin (Google), Scala, Clojure (LISP murre), Elixir, Pearl, Groovy
Web Assembly mahdollistaa eri kielten käyttämisen selaimissa

Web Development Frameworks

Frontend JS Frameworks: React.js (Facebook), Angular.js (Google), Vue.js, Backbone.js, Ember.js, Aurelia, Cycle.js, D3.js, Knockout.js

Server side JS Frameworks: Express, Next.js, Nuxt.js, Angular Universal, Hapi.js, Sails, Koa, Adonis, Loopback, Swagger

Server side Python Frameworks: Django, Flask, Web2py, Pylons, Pyramid, Tornado, Bottle, Diesel, Pecan, Falco

Server side PHP Frameworks: Laravel, Symfony, CodeIgniter, Yii2, Zend

Server side Ruby Frameworks: Ruby on Rails, Sinatra, Nitro

Server side Java Frameworks: Spring, Grails, Google Web Toolkit (GWT), Vaadin, Blade, Play, JSF, Jakarta 5 (EES), Vert.x

Server side C# Frameworks: ASP.NET

Web site performance analysers

Lighthouse (Google)

Browsers

Chrome, Firefox, Opera, UC Browser, 360 Browser, Baidu

Version control

GIT, Apache Subversion, TFS, CVS, WebSVN

Web-based version control repository hosting service

GitHub, Bitbucket (Atlassian)

Editors

Visual Studio Code, Sublime Text, Atom, Notepad++, Emacs

Integrated Development Environments (IDE)

Vagrant, GoCD
Visual Studio, IntelliJIdea, Xcode, Eclipse, NetBeans, Komodo
PhpStorm, WebStorm, PyCharm, Android Studio, Atom by Github, Qt Creator

API development tools

Postman, jsbin.com

Unit testing, Test tools

JS: Jest, Mocha, Ava.js, Chai
JUnit, Selenium, Grinder, Karma, Enzyme(AirBnB), PHPUnit, Jasmin, Should, Expect
Testlink, Timetravel

Mobile app builders

React Native, Firebase (Google), NativeScript, Ionic, PhoneGap / Cordova, Xamarin, Flutter(Google)

Container packaging tools

Docker, Puppet for automation of AWS, Chef provides enterprise-wide analytics

Container orchestration

Kubernetes, Google Container Engine, Amazon ECS, Azure Container Service, Marathon, Ansible, Swarm, Mesos

Microservice management

Istio

Authentication

OAuth2, JWT

Low-Code software platform

OutSystems, Mendix, Applan, Salesforce lightning, PowerApps (Microsoft)
Google App maker, Zoho creator, Caspio, Betty Blocks, Fujitsu, TrackVia, AppGyver, Alpha Anywhere, Kony, Appery.io, Cotham, EachScape, Appcelerator, Appsheet, Kintone, FileMaker, Zudy, Pulp Stream, Engine Yard, WaveMaker Rapid, Servoy, Visual LANSa, VINYL, Aware IM, Pega Platform, ColdFusion Builder, Spring Boot, Spring Roo

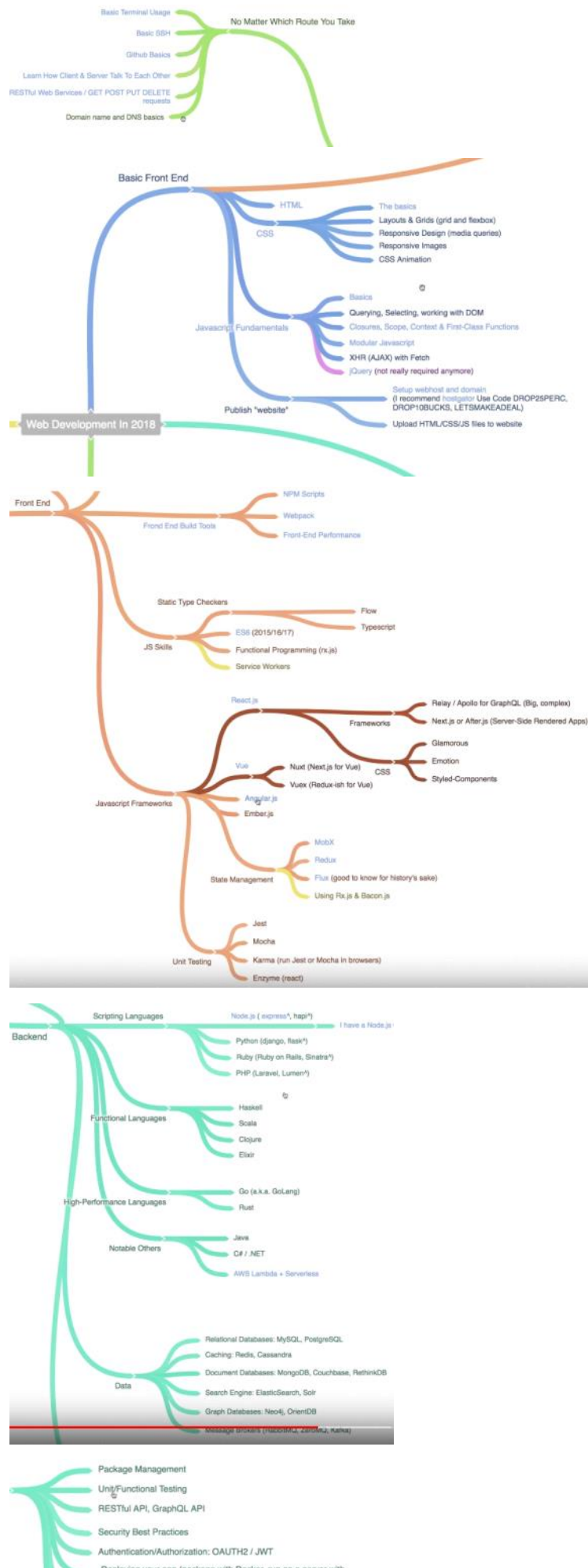
Low-code with BPM

QuickBase, Nintex, MatsSoft, FlowForma, KissFlow

<https://uk.pcmag.com/cloud-services/89789/guide/the-best-low-code-development-platforms-of-2018>

IT service management software

ServiceNow, Remedy, Flexera, Remedy, Altiris, Veeva (Microsoft), CA Technologies, Bluewin, SAP, Oracle, Glue



QuickBase, Nintex, MatsSoft, FlowForma, KissFlow

<https://uk.pcmag.com/cloud-services/89789/guide/the-best-low-code-development-platforms-of-2018>

IT service management software

ServiceNow, Requeste, Efecte, Remedy, Altris, Vendor (Microsoft), CA Technologies, Planview, SAP, Oracle, Clarizen, Smartsheet, Workfront, AppDynamics

Data bases

PostgreSQL, MySQL, MariaDB, RestDB, SQLServer, Oracle

NoSQL data bases

Document: MongoDB, CouchDB, RavenDB

Column: Cassandra, Apache Hbase

Key value: Redis(cache), Riak, Voldemort

Graph: Neo4j

Other: Dyanomite, Hypertable

NewSQL data bases

FoundationDB (Apple), Firebase(Google)

Module bundlers

Parcel, Webpack, Brunch, Gulp, Browserify, Rollup

Task runner: Grunt, Bower, Gem

Package/dependency managers: NPM, CLI (Polymer/Angular), Yarn, Bower, Composer, Nuget

Library load: CodeKit

BEM = Block, Element and Modifier

Build tools

Gradle, Ant (Gnu), Maven (Apache), Make

Transpilers / JS compiler

Babel = kääntää ES6 ja ES5 välillä

Image editors

Photoshop, Illustrator, Gimp

UX designer

Adobe XD

Middleware and Integration

WSO2, NServiceBus, MuleSoft (SalesForce), BizTalk (Microsoft)

Cloud Hosting services (PaaS, IaaS) [Serverless]

XaaS = Everything as a Service

BPaaS = Business Process as a Service

SaaS = Software as a Service

Dropbox, Salesforce, GoogleApp

PaaS = Platform as a Service

Azure, Force.com, Google App engine, ServiceNow

IaaS = Infrastructure as a Service

Amazon Web Services Lambda (AWS), OpenStack, VMware, Rackspace, Netlify

Google Cloud Platform (GCP), Heroku, Zeit Now, Digital Ocean, OpenShift/Hybrid Cloud (RedHat)

A2 hosting, Inter server, Cloudways, Fast comet, Host1plus, Siteground, Liquid web, Rose hosting, Gigapros

IBM Bluemix, Alibaba, Aliyun, SAP, Oracle, VMware, DELL, EMC

Database hosting: MongoDB Atlas, Mlab (Mongo), Cosmos DB, Redis Labs, Compose

Open source serverless frameworks: Fission, Fn, Kubeless, Apache OpenWhisk

The Serverless Framework is a free and open-source web framework

Cloud storage

Amazon S3 (S3 bucket)

Clustering

Apache Zookeeper, Apache Helix?

DevOps CD / CI

Jenkins, SonarQube, OpenShift, Hudson

Content Management Systems (CMS)

Wordpress + Elementor, Drupal, Joomla, Thumbir, Stacey, Ghost, Keystone

Static web pages

DreamWeaver, FrontPage, GoLive

Reporting tools

MS Power BI, MS Reporting Services, Excel Pivot, Power Pivot

Web Servers

Tuote	Toimittaja	Markkinaosuus
Apache	Apache	48.5%
nginx	NGINX Inc	35.4%
IIS	Microsoft	10.8%
LiteSpeed	LiteSpeed	2.9%
GWS	Google	1.1%

Application Servers

Tuote	Markkinaosuus
Tomcat	63,8%
Wildfly/JBoss	13,8%
Jetty	9,0%
Glassfish	5,6%
WebLogic	4,5%

Software bundles

MEAN (Node, Express, Angular, Mongo)

LAMP (Linux, Apache, MySQL, PHP)

XAMPP (Apache HTTP Server, MariaDB, PHP, Perl)

Web templates

Open Source pohjia web sivuille

<http://www.oswd.org/>

ETL

QlikView, Power BI, Oracle, Tableau

NPM moduilit

body-parser = parsing the body of incoming requests

React-redux = react - redux linkki

React-thunk = middleware

Redux-logger = redux logger

Morgan = logging

Multer = parsing form data bodies

Bcrypt = kryptaus / hashing

Jsonwebtoken = Token

Artificial Intelligence (AI) / Robottiikka

Blue Prism, UiPath, IBM Watson, Microsoft Bot Framework, IPsoft Amelia

Infrastructure platforms

VMware vCenter and Microsoft System Center Configuration Manager



Muita:

Project and Issue Management

Jira Commercial tool by Atlassian SW
GreenHopper Commercial tool by Atlassian

Document Management

- Knowledge Tree : OpenSource/Commercial by Knowledge Tree Inc

Collaboration, Information creation and sharing

- MediaWiki : OpenSource
- Confluence : Commercial tool by Atlassian SW
- Portals by Xoops : OpenSource
- Project Log : Tieto/OpenSource

Continuous Integration

- CI engine by Hudson : OpenSource
- Lothar test automation system : Tieto
- Hudson plugins : for Unit testing, src analysing, Jira : Tieto
- Virtual environments by VMWare : Freeware/Commercial SW
- Virtual server monitoring by Cacti : Tieto/OpenSource

Integrations and User Management

- Fedora Directory Server : OpenSource
- User Management scripts
- Plugins for Hudson and Jira : Tieto
- SingleSignOn by Cas : OpenSource

Cron: Gearman, Crunz

Search engines: ElasticSearch

Caching: Varnish,

In-memory data stores: Memcached

Team collaboration and development tool: Glitch (a coding website)

Prototyping: Ngrok

JSX

Elm

HOC

Wicket

Vert.X, wildFly

HAML

Cockpit, Flannel, Atomic

Greensock = css animations

Communities and Companies

Google

Angular, Firebase, Go, Kotlin, Polymer, Cloud Platform, Kubernetes,

Facebook:

React, GraphQL, Jest, Yarn, Metro, Watchman

Atlassian

Jira, Confluence, Bitbucket, GreenHopper

Redhat

OpenShift, WillFly(Jboss), BPM Suite, Ansible, Hybrid Cloud

Microsoft

TypeScript, C#, ASP, IIS, PowerApps, Azure

Apache

HTTP Server, Tomcat, Subversion, Maven, Struts, Tapestry, Camel

Github

Git, Atom

Gnu

Emacs, gcc

Amazon

AWS Lambda, S3 Bucket, Dynamo DB

People

Martin Fowler

Scott Allen

Maximillian Schwarzmuller / Academind, Udemy

Brad Traversy / Traversy media

Mosh Hamedani

Derek Banas

Ben Bjurstrom (Safe)

Bucky Roberts / The new Boston?

John Brunswick

Sam Newman

Academind

Simplilearn

Ydemy

Java Brains (Java)

Adam Bien (Java)

Cave of Programming

Etc

PaymentRequestAPI, ...

Digital Signage -ympäristö

R

Robotic Process Automation (RPA)

Ableton

DataDog, Consensys, Intermix, Raygun, Octopus Deploy, Mapr, G2i, sumologic, Dremio, LiveRamp, PubNov, VictorOps

HTTP

HTTP Paluuarvot

<https://www.restapitutorial.com/httpstatuscodes.html>

200	GET toiminto onnistui
201	POST tieto luotu
204	DELETE toiminto onnistui - ei palauteta mitään
302	GET palvelin ei vastaa
400	POST annettu tieto puuttuu tai väärin
401	POST Käyttäjätunnus tai salasana väärin
404	GET ei löytynyt / tuntematon
500	Järjestelmävirhe

HTTP-standardi määrittelee myös pyyntötyypin HEAD. Käytännössä HEAD:in tulee toimia kuten GET, mutta se ei palauta vastauksenaan muuta kuin statuskoodin ja headerit, viestin bodyä HEAD ei palauta ollenkaan.

HTTP pyyntöjen tulee olla safe ja idempotent

Safe = pyynnöllä ei ole sivuvaikutuksia

Idempotent = jos pyynnöllä on sivuvaikutuksia, lopputulos on sama suoritetaanko pyyntö yhden tai useamman kerran

Huom POST ei ole näitä, koska se lisää dataa joka kerta

HTT operaatiot

notes/10	GET	hakee yksittäisen resurssin
notes	GET	hakee kokoelman kaikki resurssit
notes	POST	luo uuden resurssin pyynnön mukana olevasta datasta
notes/10	DELETE	poistaa yksilöidyn resurssin
notes/10	PUT	korvaa yksilöidyn resurssin pyynnön annetulla datalla
notes/10	PATCH	korvaa yksilöidyn resurssin osan pyynnön datalla

HTML

Atribuutit

Jos painikkeiden attribuutin name arvo on kaikilla sama, muodostavat ne nappiryhmän, joista ainoastaan yksi voi olla kerrallaan valittuna

```
<a href='#>Refresh</a>
```

Jonkin osan näkyvyyttä voi säädellä attribuutilla style (CSS määrittely)

```
<div style='none'> tai <div style=''>
```

Whitespace html koodin keskellä


```
<pre> a    b    </pre>
```

HTML Events

Common HTML events:

- **onchange** An HTML element has been changed
- **onclick** The user clicks an HTML element
- **onmouseover** The user moves the mouse over an HTML element
- **onmouseout** The user moves the mouse away from an HTML element
- **onkeydown** The user pushes a keyboard key
- **onload** The browser has finished loading the page

https://www.w3schools.com/jsref/dom_obj_event.asp

Event delegation

Määritellään Formille event handler

Se koskee kaikkia formin sisäisiä elementtejä

Web / Browser

<https://developer.mozilla.org/en-US/docs/Learn>

Local Storage

local storage on selaimessa oleva avain-arvo- eli key-value-periaatteella toimivaan tietokanta. Storageen talletetut arvot säilyvät vaikka sivu uudelleenladataan.

Jokaisella selaimella käytettävällä web-sovelluksella on oma storagensa

`window.localStorage.setItem('nimi', 'juha tauriainen')` - tallettaa avain, arvo parin

`window.localStorage.getItem('nimi')` - hakee avainta vastaavan arvon

`window.localStorage.removeItem('nimi')` - poistaa avain,arvo parin

`window.localStorage.clear()` nollaa local storagen tilan kokonaan

Local Storageen talletettavat arvot ovat merkkijonoja, joten Local Storageen ei voi tallettaa Javascript-oliota

Web sockets

WebSocketien avulla on mahdollista muodostaa kaksisuuntainen kommunikaatiokanava selaimen ja palvelimen välille. Tällöin frontendin ei tarvitse pollata backendia, riittää määritellä takaisinkutsufunktiot tilanteisiin, joissa palvelin lähettää WebSocketin avulla tietoja tilan päivittämisestä. On suositeltavaa käyttää Socket.io-kirjastoa!

AJAX = Asynchronous JavaScript And XML

With AJAX you can:

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

Event delegation

When adding an Event listener to a DOM element, the listener is also added to all the children of the element.

`currentTarget` = the element the listener was added to

`target` = the element that invoked the event

Java Script

Java Script (JS)

<https://www.javascript.com/>

<https://www.w3schools.com/js/default.asp>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>

<https://developer.mozilla.org/en-US/docs/Web/API>

Survive JS: <https://survivejs.com/>

JS Info: <https://javascript.info>

Fun: <https://www.youtube.com/channel/UCO1cgjHGzsSYb1rsB4bFe4Q>

You don't know JS: <https://github.com/getify/You-Dont-Know-JS>

ES6: <http://es6-features.org/#Constants>

ES6 promises: <https://www.youtube.com/watch?v=swdWUWtGxR4>

Mastering JS: <https://www.youtube.com/watch?v=e3ms5HKvtZ8&list=PLTgRMOcmRb3PN9kK-NhSZek4xWZD-81cz>

Coding conventions: <https://google.github.io/styleguide/javascriptguide.xml>

JavaScript Language

Javascriptissä ei ole muita tyyppejä kuin: Boolean, Number, String, Object, Null, Undefined ja Symbol(ES6)

Javascriptistä on olemassa useita staattisesti tyypitettyjä versioita, suosituimmat näistä ovat Facebookin kehittämä flow ja Microsoftin typescript.

Arrow functions

Funktio voidaan välittää argumenttina kutsuvalle funktiolle. Arrow function syntax:

```
function(arg){}
```

voidaan ilmaista myös näin:

```
(arg) => {}
```

Strict mode (ES5)

JS valvoo koodin oikeellisuutta tarkasti

Kun koodiin laittaa rivin: "use strict", niin kyseinen scope käsitellään strict modena

Arguments

There is a special array-like object named arguments that contains all arguments by their index.

```
function showName() {  
  alert( arguments.length );  
  alert( arguments[0] );  
  alert( arguments[1] );  
}
```

Huom: Arrow functions do not have "arguments"

Operators

```
var name = o && o.getName();
```

```
var name = cachedName || (cachedName = getName());
```

```
var allowed = (age > 18) ? 'yes' : 'no'
```

Falsy

The specific list of "falsy" values in JavaScript is as follows:

"" (empty string) If (") -> false

0 , -0 , NaN (invalid number)

null , undefined

false

Any value that's not on this "**falsy**" list is "**truthy**". Here are some examples of those:

"hello"

42

true

[] , [1, "2", 3] (arrays)

{ } , { a: 42 } (objects)

function foo() { .. } (functions)

Scope

Scope look-up always starts at the innermost scope being executed at the time, and works its way outward/upward until the first match, and stops. (variable definitions) If no match is found, global is used.

Shadowing = The same identifier name can be specified at multiple layers of nested scope

The **let** keyword attaches the variable declaration to the scope of whatever block (commonly a { .. } pair) it's contained in

Closure

A closure is a function that remembers its outer variables and can access them. All functions in JavaScript are closure.

JavaScript määrittää ensin funktiot ja ajaa koodin vasta sitten. Jokaiselle funktiolle luodaan Lexical Environment.

You can think of closure as a way to "remember" and continue to access a function's scope (its variables) even once the function has finished running.

```
function makeAdder(x) {  
  // parameter `x` is an inner variable of makeAdder  
  // inner function `add()` uses `x`, so  
  // it has a "closure" over it  
  function add(y) {  
    return y + x;  
  }  
  return add;  
}
```

Voidaan parametroida funktion rakentaminen.

```
var plusTen = makeAdder( 10 );  
plusTen( 13 ) -> 23
```

Functions

Functions in JavaScript are Objects

In JavaScript **this** keyword is determined by how a method is called

If a function is called as a standalone function outside an object,

this is set to global object, which is the window object in browsers in strict mode

The "use strict" directive switches the engine to the "modern" mode

Objects

There are different ways to create new objects:

- Define and create a single object, using an object literal.
 - var person = { firstName:"John", lastName:"Doe" };
- Define and create a single object, with the keyword new
 - var person = new Object(); person.firstName = "John"; person.lastName = "Doe";
- Define an object constructor, and then create objects of the constructed type (melkein Luokka)
 - function Person(first, last) {

```

    this.firstName = first;
    this.lastName = last;
  }

```

```

var obj = {};
var obj = {
  name: 'Carrot',
  details: {
    color: 'orange',
    size: 12
  }
};
function Person(name, age) {
  this.name = name;
  this.age = age;
}
var dude = new Person('John', 54);

```

Huom! Nuolifunktioita ei voi käyttää Olioiden metodeina, sillä silloin this ei toimi.

JavaScript Objects are collections of key-value pairs

```

const person = {
  name: 'John',
  talk() {} // uusi tapa määritellä objektin metodi
}
person.talk()
person.name = 'Jeff' // vanha tapa
person['name'] = 'Jeff' // uusi tapa

```

```

const talk = person.talk // reference to function
walk() // this is undefined
const talk = person.talk.bind(person) // returns a new function
walk() // this is the person object

```

Classes

```

class Person {
  constructor(name){
    this.name = name
  }
  talk() {}
}
const peson = new Person('John')

```

Inheritance

```

class Teacher extends Person {
  constructor(name, degree){
    super(name)
    this.degree = degree
  }
  teach(){}
}
const teacher = new Teacher('Jeff', 'MSc')

```

Modules

Objects in modules are private by default

Jokainen luokka määritellään omassa tiedostossaan ja exportoidaan

```
import { Person } from './person'
```

```
export class Teacher extends Person { ...
```

Default exports tapa exportoida jos halutaan exportoida vain yksi luokka

```
export default class Teacher extends Person { ...
```

```
import Teacher, {jokuMuuExportoituFunktio} from './teacher' // default export käytetään ilman  
kaarisulkeita
```

Destructuring

```
const address = {  
  street: "  
  city: "  
  country: "  
}
```

Destrukturoidin avulla voimme "kerätä" olion oliomuuttujien arvot suoraan omiin yksittäisiin muuttujiin.

Asetetaan objektin ominaisuudet kolmeen muuttujaan:

```
const {street, city, country} = address
```

Jos halutaan käyttää eri muuttujan nimeä

```
const {street: katu} = address
```

Spread operator is presented using 3 dots

```
const eka = [1, 2, 3]
```

```
const tok = [4, 5, 6]
```

```
const yht = [...eka, ...tok, 7, 8, 9]
```

Coercion (=pakottaminen)

JS muuttaa arvon tyyppiä. Explicit = koodaamalla. Implicit = lennosta (42 == '42') -> true

=== produces false , because the coercion is not allowed

Strinct mode

Hoisting

var's are declared(hoisted) at the top of the function = scope is the whole function

Ie. var variable declarations are processed at function start

const and **let** are not hoisted. They are scoped in the current block of code

Siis funktiota tai var muuttujaa voidaan käyttää ennen sen määrittystä (declaration) kyseisen function scopen sisällä

Huom: Let (ES6) muuttujaa ei voi !

Undeclared, undefined and null

Undeclared = muuttujaa ei ole määritelty (declared) - ei voi käyttää vielä = (virheilmoitus voi olla myös "is not defined")

Undefined = muuttuja on määritelty (declared) mutta siihen ei vielä ole asetettu arvoa

Null = muuttujan arvo on asetettu arvoksi null (muuttujalla on siis arvo)

Tarkista arvo näin: (foo === undefined) - Huom { typeof foo } ei toimi !

Triple equals checks for equality and type.

Immediately invoked function expression (IIFE)

IIFE = Function Expression is created and immediately called.

Notice parentheses around the function:

```
(function IIFE(){  
  console.log( "Hello!" );  
})();
```

-> "Hello!"

An IIFE could be used for controlling variable scope. In order not to pollute the namespace

This

To which object 'this' points to depends on how/where the function was called.

It's important to realize that this does not refer to the function itself !

Object prototyping mechanism

Behaviour delegation or Lookup chain for properties (not inheritance, but close)

Polyfilling

Polyfilling is taking the definition of a newer (ES6 or later) feature and producing a piece of code that's equivalent to the behavior, but is able to run in older JS environments.

Transpiling

Using a tool that converts your newer code into older code equivalents.

Transpiling = Transforming + Compiling

A transpiler is used when building code to be used by browsers

Promises (ES6)

Promises are a time-independent wrapper around a "future value," which lets you reason about and compose them regardless of if the value is ready or not yet.

Event loop

Java script is single threaded -> long lasting tasks can block execution

To get around that use callback functions and promises

Browsers can use Web API's, which are multi threaded

How event loop works

Event loop pulls functions from function call stack and executes them

When a callback function is encountered, a Web API is used to execute the function in another thread

When finished web API puts the callback/promise into the event queue

When the function stack is empty, Event loop moves the callback/promise from event queue and places it to the function stack for execution

Functional programming

Functions are value and can be assigned to variables and Pass a function around

Don't code on detail level every little thing that needs to be done

Use Higher order functions instead

In functional programming we write less code, because we can compose a larger function from many smaller functions

Separate problems to small solutions and compose the big solution from them

Create generic low level functions and compose more complicated functions from them

Operate on array, but leave the original array alone

Use **forEach** instead of `for(let i= 0; i++; i<10)`

Use **map** to loop over an array and apply a callback function for each element

Use **reduce** to get a single result from a number of things (array of data -> single result)

Use **currying** (=Partial) to reuse a low level functions in order to create a high level function by adding parameters

(Lodash can implement currying to normal functions)

Reusing code as functions instead of reusing code as objects

Use **filter** to filter data

In functional programming actions change external state

In object oriented programming actions change internal state

Coding conventions

https://www.w3schools.com/js/js_conventions.asp

BDD

Behaviour Driven Development: Määrittely kirjoitetaan testien muodossa. Speksit kirjoitetaan ennen koodia.

```
describe("pow", function() {  
  it("raises to n-th power", function() {  
    assert.equal(pow(2, 3), 8);  
  });  
});
```

Write 'self-descriptive' code

Älä kommentoi koodia, vaan refaktoroi se niin, että sen ymmärtää

if (isPrime(i)) continue; // <= Name functions descriptive

Rajapinnat pitää kommentoida:

- Overall architecture, high-level view

- Function usage

- Important solutions, especially when not immediately obvious

JavaScript Kehitys työkaluja

VSCode ja LiveServer -addOn (pitää installoida), joka päivittyy automaattisesti, kun koodia muutetaan

Debug

Koodin suorituksen voi pysäyttää Chromen developer konsolin debuggeriin kirjoittamalla:

```
debugger; // <-- the debugger stops here
```

<https://developers.google.com/web/tools/chrome-devtools/>

F5 = Refresh

F8 = continue

F10 = Next step

F11 = Next step into code

Logging in web apps

- Writing into an HTML element, using innerHTML
- Writing into the HTML output using document.write()
- Writing into an alert box, using window.alert()
- Writing into the browser console, using console.log()

Global error handling

Node.JS has process.on('uncaughtException'). In a browser we can assign a function to special window.onerror property. It will run in case of an uncaught error - syntax:

```
window.onerror = function(message, url, line, col, error) { ... }
```

Node JS

NodeJS (by Twitter) is a runtime environment for Javascript language

<https://nodejs.org/en>

<https://nodejs.org/en/docs/>

<https://nodejs.org/api/documentation.html>

<https://twitter.com/nodejs>

<https://www.w3schools.com/nodejs/>

<https://github.com/nodejs/node>

https://www.youtube.com/watch?v=65a5QQ3ZR2g&list=PL55RiY5tL51oGJorjEgl6NVeDbx_fO5jR

Node is a C++ program which includes Google's V8 JavaScript engine

Node keeps running in the event loop as long as there are event listeners registred - for example httpServers

With node you can:

- write and run the server

- handle requests, REST API

- business logic, database connections, authentication, input validation,

- file system, streams, buffers, process, url handling, dns

Node version päivitys

npm install npm@latest -g

Node projektin luonti

npm init = Kysyy Node projektin perustiedot ja luo konfiguraatio tiedoston package.json

Lisätään package.json tiedostoon scripteihin: start:

```
"scripts": {  
  "start": "node index.js",
```

Node käynnistys

node index = ajaa index.js JavaScript tiedoston consolissa

Node start = käynnistää package.json tiedostossa konffatun JavaScript tiedoston

Node server pitää käynnistää JavaScript koodissa explisiittisesti

```
const http = require('http'); // Node core  
module  
function myRequestHandler(req, res) {} // Write a  
request handler  
http.createServer(myRequestHandler).listen(8000); // Start server
```

Reititys ilman Expressiä

```
if (req.url === '/api/rajapinta'){  
}
```

Node toiminnallisuus

Jokainen JavaScript ohjelma on moduuli. Jokaiselle ohjelmalle välitetään seuraavat parametrit:

- exports, require, module, __filename ja __dirname

- Jotka löytyvät Global modulista

Core Modules: http, https, fs, path, os

Other Modules: crypto

Jos halutaan uudelleen käyttää koodia, pitää JavaScript ohjelmassa (koodi.jsp) exportoida muualla käytettävät funktiot ja muuttujat. Seuraava koodi exportoi funktion objektina

```
module.exports.funktio = funktio
```

Toisessa javascript ohjelmassa voidaan ottaa ne käyttöön:

```
const muuttuja = require('./koodi');  
muuttuja.funktio(); // Voidaan käyttää objektin  
metodina
```

Seuraava koodi exportoi pelkän funktion

```
module.exports = funktio
```

Toisessa javascript ohjelmassa voidaan ottaa se käyttöön:

```
const muuttuja = require('./koodi');  
muuttuja(); // Voidaan käyttää suoraan  
funktiona
```

__dirname on Noden globaali muuttuja, joka viittaa nykyiseen hakemistoon.

Noden konventiona on määritellä projektin suoritustila ympäristömuuttujan NODE_ENV avulla

Windowsissa tarvitaan cross-env kirjasto

Tiedostossa package.json voi asettaa NODE_ENV arvon

```
"scripts": {  
  "start": "cross-env NODE_ENV=production node index.js",  
  "watch": "cross-env NODE_ENV=development nodemon index.js",  
  "test": "cross-env NODE_ENV=test jest --verbose"  
}
```

Nodemon

Nodemon will watch the files in the directory in which nodemon was started, and if any files change, nodemon will automatically restart your node application:

```
npm install --save-dev nodemon  
package.json -> "scripts": {  
  "start": "node index.js",  
  "watch": "nodemon index.js" }  
npm run watch
```

<https://nodemon.io/>

<https://github.com/remy/nodemon>

Debuggaus

Debuggaus onnistuu myös Chromen developer-konsolilla, käynnistämällä sovellus komennolla:

```
node --inspect index.js
```

Debuggeriin pääsee käsiksi kirjoittamalla chromen osoiteriville

```
chrome://inspect
```

Moduuleja

<https://nodejs.org/api/modules.html>

https://www.w3schools.com/nodejs/ref_modules.asp

https://www.w3schools.com/nodejs/nodejs_email.asp

NPM

Node Package Manager

<https://nodejs.org/>

<https://nodejs.org/en/docs/>

<https://nodejs.org/api/documentation.html>

<https://www.npmjs.com/>

NPM

NPM is a dependency management tool

NPM luo **package.json** tiedoston, joka on projektin konfigurointi tiedosto

package.json tiedostossa hallitaan projektin riippuvuudet ja modulien versiot

Jos kaikki riippuvuudet on kirjattu package.json tiedostoon, niin npm install asentaa kaiken tarvittavan

"scripts": { } kohtaan voi lisätä komentoja, joita voi ajaa npm:llä

"start": "node index.js" = Sovellus käynnistää npm start -komennolla

Projektin riippuvuudet voi päivittää komennolla: (kaikkien kirjastojen uusimmat versiot ladataan - ei major versio)

npm update

Vastaavasti jos aloitamme projektin koodaamisen toisella koneella, saamme haettua ajantasaiset, package.json:in määrittelyn kanssa yhteensopivat riippuvuudet komennolla:

npm install

Semanttinen versionti:

^4.16.2 = missä 4 = pääversio (major), 16 = 16 minor, 2 = 2 patch

^ tarkoittaa, että kun projektin riippuvuudet päivitetään, asennetaan expressistä versio, joka on vähintään 4.16.2. Pääversio EI saa olla 5, mutta patch eli viimeinen numero tai minor eli keskimäinen numero voi olla suurempi

Riippuvuudet voi päivittää komennolla: npm update tai npm install

NPX

Npx is an npm package runner

npm audit	Tarkastaa projektin käyttämien pakettien tietoturvan
npm audit -fix	Korjaa projektin käyttämien pakettien tietoturvan
npm init	Kysyy projektin perustiedot ja lue perus konfiguraation
npm --yes	Luo projektin konfiguraation oletustiedoilla
npm -y	
npm config set init-author-name "Antti"	Asettaa oletusarvon npm init komennolle
npm set init-author-name	
npm config delete init-author-name	Poistaa asetetun oletusarvon
npm --version	versio
npm -v	
npm	help
npm help	

npm install paketti --save	Asentaa paketin TAI päivittää paketin uusimpaan
-- save	Tallettaa muutokset package.json tiedostoon
--save-dev	Luo kehitystä varten devDependencies
npm install -production	Ei asenna dev kamaa (jos vaikka siirtää uuteen)
npm uninstall	Poistaa asennuksen
npm remove	poistaa asennuksen
npm rm	
npm update	Päivittää projektin kaikkien pakettien versiot
npm update paketti	Päivittää paketin version
^ version nimen edessä	päivittää uusimpaan Minor versioon
* ilman version nimeä	päivittää uusimpaan versioon. Myös Major versioon
npm install -g paketti	Suorittaa asennuksen globaalisti koneelle
npm root -g	näyttää minne globaalit moduulit asennetaan
npm list	Listaa sovelluksen riippuvuudet (käyttämät paketit)
npm list --depth 1	Riippuvuusketjun syvyys
npm outdated --depth 0	Tutkii asennettujen moduulien ajantasaisuuden

JavaScript tiedostossa

const muuttuja = require('paketti'); asettaa muuttujaan asennetun paketin
 paketti.toiminto(); Paketin funktiota voidaan kutsua suoraan

NPM paketteja (kirjastoja)

Axios	Selaimen ja palvelimen väliseen kommunikaatio. Kirjasto HTTP requestien
Babel-eslint	Eslintin Babel versio, jotta eslint ymmärtää ES6 ominaisuuksia esim nuolifunktiot ja class property -syntaksi
Babel	Transpiloi ES6 ja ES7 koodin käyttämään vanhempaa Javascript-syntaksia (babel-core, babel-loader, babel-preset-react, babel-preset-env, style- loader, css-loader, babel-plugin-transform-class-properties)
Bcrypt	Salasanojen hashaaminen
Bcryptjs	
Body-parser	Middleware http pyynnön request datan lukemiseksi. Otetaan käyttöön: app.use(bodyParser.json()) tallettaa JSONin bodyyn. Data löytyy nyt const
Connect	http server
Config	Noden yhteydessä globaalien/ympäristö muuttujien konfiguroinnin
Cookie-parser	
Cors	Tuotantoa varten pitää ottaa huomioon: CORS middleware, joka sallii muista origineista tulevat pyynnöt.
Cross-env	Windowsissa tarvitaan ympäristömuuttujille
Deep-freeze	Vahtii, ettei parametrinä annettua muuttujaa muuteta
Dotenv	Sovelluksen juurihakemistoon tehdään tiedosto nimeltään .env, jossa määritellään ympäristömuuttujat. Nodessa ne otetaan käyttöön: if

	<pre>(process.env.NODE_ENV !== 'production') { require('dotenv').config() }</pre>
Embedded JS	
Enzyme	Front Endin testaukseen Jestin lisäksi
Enzyme-adapter-react-16	React adapter for Enzyme
Eslint	Koodin tyypin analysointia (Visual Studio ESlint-plugin)
Eslint-jest-plugin	Poistaa virheilmoitukset jest testeistä
Expect	Testaus
Express	Web framework
Fetch	Selaimen ja palvelimen väliseen kommunikaatio
Jest	Test runner - Testastyökalu
Json-server	Testiserveri. Data talletetaan projektin juureen db.json "server": "json-
Jsonwebtoken	Muodostaa web tokenin, jonka voi tallettaa selaimen muistiin Local Storageen.
Live-server	kevyt serveri joka myös päivittää sovelluksen muutokset automaattisesti
Lodash	JS utilities
Mongodb	MongoDB tietokanta-ajuri
Mongojs	MongoDB ajuri on tehty mongodb ajurin päälle
Mongoose	MongoDB ajuri
Morgan	Middleware logitusta varten
Mysql	MySQL tietokanta ajuri
Nodemon	Monitorin joka automaattisesti päivittää muuttuneen applikaation.
Prop-Types	PropTypes.string.isRequired (tms) avulla varmistetaan, että oletetut propsit välittyvät React -komponentille ja että niiden tyypit ovat ok.
Pug	Template engine
Puppeteer	E2E testaus. Chromen headless Node API
React	
Reactstrap	Reactin kanssa käytettävä Bootstrap UI kirjasto
React-bootstrap	Reactin kanssa käytettävä Bootstrap UI kirjasto
React-redux	Helpottaa kontekstin käyttöä { Provider } { connect }
React-router-dom	React-sovelluksen reitityksen hallinta
React-router-bootstrap	Makdillistaa
Redis	NoSql tietokanta
Redux	Hoitaa tilan hallinnan. Korvaa Reactin State:n Storella ja reducereilla
Redux-devtools-extension	Redux sovelluksen debuggaukseen. Tarvitaan Chromen Redux DevTools
Redux-thunk	Mahdollistaa asynkronisten actionien luomisen. Se on ns. redux-middleware, joka mahdollistaa sen, että komponentit käyttävät propsina saamaansa funktiota, välittämättä siitä että taustalla tapahtuu todellisuudessa palvelimen kanssa tapahtuvaa asynkroonista
Sequelize	SQL tietokantojen kanssa kommunikointiin
semantic-ui-react	Reactin kanssa käytettävä UI kirjasto kuten Bootstrap
Socket	Server side websockets

Supertest	Testaustyökalu
webpack	Paketoit sovelluksen tuotantoa varten
webpack-cli	
webpack-dev-server	

Modulien asentaminen luo sovelluksen alle hakemiston: node_modules, joka paisuu suureksi

Muista laittaa gitignore tiedostoon

Node build-in moduleja ovat: os, fs, events, http

Moduleja voi kirjoittaa itse: jokainen tiedosto on moduuli. Sen funktiot ovat tyyppiä: private

require('./public/js/yEmitter') -komennolla viitataan toisen modulin funktioihin

global luokan metodeja: console, setTimeout(), clearTimeout(), setInterval(), clearInterval(),

Modulin funktioita voi julkaista käytettäväksi seuraavasti:

module.exports.ulosNakyvaFunktio = modulinPrivaattiToiminto;

yms: Node ajaa modulin kamat kuten ne olisi koodattu kolmen pisteen paikalle:

(function (exports, require, module, __filename, __dirname){...})

Express JS

Express JS

Expressillä hallitaan Web sovelluksen routing eli siirtyminen eri sivuille

<https://expressjs.com>

Asennus ilman generaattoria

Jos ei käytetä Express generaattoria, niin tarvitaan seuraavaa

```
npm install express --save = asentaa Expressin package.json tiedostoon
```

serveri voidaan käynnistää "npm start"-komennolla, koska se on määritetty package.json tiedostossa

```
npm start
```

Tai debug moodissa

```
DEBUG=express:* node ./bin/www
```

Lisäpaketteja asennetaan tarpeen mukaan esim:

```
npm install --save express-session
```

```
npm install --save express-validator
```

Asennus Express generaattorilla

Asennetaan Express generaattori

```
npm install -g express-generator
```

Luodaan uusi projekti generaattorilla

```
express projektinNimi --view=pug // muuten view engine on jade, jota ei enää ole
```

Asennetaan kaikki riippuvuudet

```
npm init
```

Express käyttö

```
const express = require('express');
```

```
const app = express();
```

Body-parser

body-parseria käytetään HTTP POST -pyynnön käsittelyyn

```
const bodyParser = require('body-parser')
```

```
app.use(bodyParser.json())
```

Middleware

Middlewareet ovat funktioita, joiden avulla voidaan käsitellä request- ja response-olioita.

Middlewareen voi kirjoittaa itse. Se on funktio, joka saa kolme parametria:

```
const logger = (request, response, next) => {  
  console.log('Method:', request.method)  
  console.log('Path: ', request.path)  
  console.log('Body: ', request.body)  
  console.log('---')  
  next()  
}
```

Middleware kutsuu lopussa parametrina olevaa funktiota next, jolla se siirtää kontrollin seuraavalle middlewarelle.

Middleware otetaan käyttöön seuraavasti:

```
app.use(logger)
```

Deploy

Lisätään staattinen hakemisto deployykseen

```
app.use(express.static('build'))
```

React JS

<https://reactjs.org/>

<https://reactjs.org/tutorial/tutorial.html>

<https://reactjs.org/docs/hello-world.html>

<https://github.com/facebook/react>

<https://www.npmjs.com/package/react>

<https://www.w3adda.com/react-js-tutorial>

<https://reactpatterns.com/>

Academind: <https://www.youtube.com/watch?v=pgAvVxowaYU>

EggHead: <https://egghead.io/courses/start-learning-react>

EggHead: <https://egghead.io/courses/the-beginner-s-guide-to-reactjs>

SurviveJS: <https://survivejs.com/react/>

React by Facebook

React is a JavaScript **Library** for building single components

Reusable components = custom HTML elements

ReactDOM renders components to the DOM

React is maintained by Facebook and uses JSX JavaScript Syntax EXtension

React hoitaa Prefix:it, jotta vanhat browserit toimii myös

Uusi projekti / sovellus

`npm install -g create-react-app` // Ei tarvita, jos on jo kerran tehty

`create-react-app mun-sovellus`

React luo pohjan sovellukselle - myös hakemiston mun-sovellus. (sovelluksen nimi ei saa sisältää isoja kirjaimia)

`index.html` ja `index.js` tiedostoihin ei juurikaan tarvitse tehdä muutoksia.

`cd mun-sovellus`

`npm install axios --save` // Asenna tarvittavat lisäosat

`npm start` - Starts the development server

`npm run build` - Bundles the app into static files for production

`npm test` - Starts the test function

Coding conventions

React-komponenttien nimien tulee alkaa isolla kirjaimella.

Component class names and directories are capitalized so that components as custom HTML elements can be separated from native HTML elements.

A component is a function that returns JSX

Komponenttien presentational vs. container -jaottelu on hyväksi havaittu tapa strukturoida React-sovelluksia

Tila eli State

State property can be used in classes which extend Component

State property is a special property = if it changes the page will be rerendered

State can not be changed directly -> use `this.setState()` -method instead

Huom! React kutsuu funktiota `setState` asynkroonisesti

Jos komponentti tarvitsee tilaa, on luokkasyntaksin käyttäminen välttämätöntä. Reactin filosofian mukaista on sijoittaa Tila eli State mahdollisimman ylös komponenttihierarkiaan. Pattern: create only a few classes where the state is handled. The rest of the classes are used for rendering only.

Redux

Käytä Redux kirjastoa sovelluksen tilan hallintaan!!! Kts seuraava sivu ->

Reititys

React tarvitsee (react-router-dom) kirjaston reititystä varten
Reititys, eli komponenttien ehdollinen, selaimen urliin perustuva renderöinti otetaan käyttöön sijoittamalla komponentteja Router-komponentin lapsiksi, eli Router-tagien sisälle

Http liikenne Backendille

React tarvitsee Axios tai Fetch kirjaston http requesteille

Rakenne

Jaa React projekti hakemistoihin:

- actions - Redux tapahtumien käsittelijät omiin tiedostoihin
- components - UI komponentit, jotka eivät tunne tilaa
- containers - UI komponentit, jotka tuntevat tilan
- reducers - Redux reducers
- services - Palvelut backendin rajapintoihin
- store.js - oma tiedosto varastolle

Sovellus.js koodi

```
class Sovellus extends React.Component
implementoi render() metodi, joka palauttaa JSX koodia
export default Sovellus
```

JSX

React-komponentti palauttaa JSX muotoa, joka kääntyy javascriptiksi
Aaltosulkeiden sisällä oleva Javascript-koodi evaluoidaan

```
const a = 10
const b = 20
<p>{a} plus {b} is {a + b}</p>
class => className
onclick => onClick
Only one root element can be returned in render() method (joskin tämän voi kiertää)
```

Luokka

Local state is available only to classes (extending React.component)
The only place where you can assign this.state is the constructor

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date(), name:"Antti"};
  }
  render() {
    return (
      <div>
        <h1>Hello, {this.state.name}!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
ReactDOM.render(<Clock />, document.getElementById('root'))
);
```

```
// Käytä komponentin metodin määrittelyä tällä muotoa
handleClick = () => {
  this.setState({ counter: this.state.counter + 1 })
}
```

```
// Jos funktio pitää välittää toiselle komponentille, käytä funktiota, joka palauttaa funktion (kaksi  
nuolta)  
handleNumberChange = () => {  
  return (event) => {  
    this.setState({ newNumber: event.target.value })  
  }  
}  
// Älä tuki Call Stackiä - Renderöinti jonottaa sen tyhjentymistä  
// Siirtää funktiokutsun event looppiin  
setTimeout (function myF(){..}, 0); // 0 = nolla sekuntia
```

Ohjeita

Suorita datan palvelimelta haku React.Component:in metodissa componentDidMount
Varaudu siihen, että ensimmäinen renderöinti tapahtuu ennen kuin palvelimelta haettava data on saapunut

Debug

Chroomeen kannattaa ehdottomasti asentaa React developer tools -lisäosa, joka tuo konsoliin uuden tabin React

Työkalut

React kehitysympäristö pyörii Nodella
create-react-app on työkalu, jolla perustetaan React projekti
Webpack is a Bundler, which bundles multiple files to one at development
Babel and Presets are precompilers
Redux is used for state handling
React-router-dom is used for routing

Questions and answers

Why does React need a root element

React hooks to the element with id=root
Reanders it's own DOM tree

What is the difference between state and props

components store internal state in State
Props are passed to components from parent element
Use 10% state and 90% props

What is context

Context is a global prop. It is globally available to all components

What are prop types and what are the benefits and drawbacks of them

Prop types is a way to know what types a component is expecting.
Prop type definitions are not always updated -> problems

Which life cycle event is the most common

componentWillMount, componentDidMount
componentWillReceiveProps can be used to check if you need to do something before a component is receiving data in props

When do you use pure component and when do you use React class

Pure components are used for performance
Classes are used only when state is needed

Explain how the React rendering works

React listens for DOM updates and renders the DOM tree every change but React checks if the update will result in a change (uses component diffing)

What is Redux

A tools for storing state in React. It uses Flux pattern.
Stores the state in a big object and use actions to change that state.

Explain how Redux works

You declare a reducer that takes in an action and a state.
When you dispatch an action the state gets updated and React renders to DOM

When do you use Redux

When you need to share data with many components
When you need global state that is shared among many components
Complicated applications most likely need Redux (start with out it)

What is a container component

A container component is a way to separate logic from JSX of a component.
Should only be used when component is doing more than just showing information
Containers make the code more complex, so don't use containers if the is no logic involed.

What is a view component

View contains all the things you want to show to the user.
View contains JSX code. Views are rarely reusable.

Snippets

https://github.com/FullStack-HY/FullStack-Hy.github.io/blob/master/snippet_ohje.md
<https://marketplace.visualstudio.com/items?itemName=xabikos.ReactSnippets>

Redux

Redux kirjastolla hallitaan sovelluksen tilaa

Components (View/App) can subscribe to the Redux Store.

When the state of a variable is changed, the component will be notified.

```
npm install redux --save
```

```
npm install react-redux --save - asentaa linkin Reactin ja Reduxin väliin
```

```
npm install redux-thunk --save - asentaa asynkronisen middleware komponentin
```

```
npm install redux-logger --save - asentaa lokkerin
```

Tarvitaan

```
import React from 'react'
```

```
import ReactDOM from 'react-dom'
```

```
import { createStore, combineReducers, applyMiddleware } from 'redux'
```

```
import { Provider } from 'react-redux'
```

```
import { connect } from 'react-redux'
```

```
import thunk from 'redux-thunk'
```

Luo Store omaan tiedostoon ja anna argumenteiksi omat reducerit

```
const reducer = combineReducers({ blogs: blogReducer, users: userReducer })
```

```
const store = createStore( reducer, applyMiddleware(thunk) )
```

```
export default store
```

Laita App komponentti Providerin sisään ja välitä store propertynä providerille (index.js)

```
const render = () => {  
  ReactDOM.render(  
    <Provider store = { store }>  
      <App />  
    </Provider>,  
    document.getElementById('root'))  
}
```

Tilaa (index.js) Redux store

```
store.subscribe(render)
```

```
render()
```

Tee jokaiselle loogiselle kokonaisuudelle (komponentille) oma reducer - parametreina vanha tila ja action

```
const myReducer = (store = initialValue, action) => { if (action.type === 'TEE_JOTAIN') { return  
newState } }
```

HUOM: Action type kannattaa olla yksilöivä, koska Redux kutsuu kaikkia reducereita kaikilla actioneilla !!!

Tee muutama oma action metodi:

```
export const myFunction = (arg1, arg2) => { dispatch({ type: 'SET_USER', data: user }) }
```

Yhdistä komponentti (react-redux) connectilla reduceriin - voi sisältää useiden reducereiden tiloja ja metodeja

```
const mapStateToProps = (state) => { return { theUser: state.users.user } } // ownPropsilla voi lisätä  
omat parametrit
```

```
const mapDispatchToProps = { myFunction1, myFunction2 }
```

```
export default connect( mapStateToProps, mapDispatchToProps )( Komponentti )
```

Käytä komponentissa storen tilaa ja metodeja propertyjen kautta

```
this.props.theUser
```

```
this.props.myFunction1()
```

Kaikki palvelut myös tietokanta (backend) haut käynnistetään reducereista

Redux

Reducereilla hallitaan tilaa Storen avulla: createStore(reducer)

Reducer saa argumenttina vanhan tilan ja toiminnon. Se palauttaa uuden tilan. Huom: Se ei muuta vanhaa tilaa!

Sovellus muuttaa storen tilaa komennolla: store.dispatch

React Redux -kirjaston määrittelemä funktio connect on paras ratkaisu siihen, miten Redux-store saadaan välitettyä React-componenteille

Yms

Asenna Redux devtools Chrome extension

Mongo DB

MongoDB

<https://www.mongodb.com/>

<https://university.mongodb.com/>

<https://github.com/mongodb/mongo>

<https://www.tutorialspoint.com/mongodb>

<https://www.youtube.com/watch?v=VELru-FCWDM>

https://www.youtube.com/watch?v=-0X8mr6Q8Ew&list=PLGLfVvz_LVvRfdt8_W0dV311Xa8SayfCY

NoSQL <-> SQL

Database = Database

Collection = Table

Document = Row

Komentoja

show dbs

use mydb creates a DB if does not exist

db shows current db

show collections

db.mycollection.find().pretty()

db.mydb.drop Droppaa 'mydb' kannassa olevat collectionit

db.mycollection.count() Palauttaa dokumenttien lukumäärän kyseisessä collectionissa

db.mycollection.distinct('name')

db.mycollection.find().explain('executionStats')

db.mycollection.ensureIndex({'address.city': 1}, {'unique': true, 'dropdups': true, 'sparse': true})

db.mycollection.getIndexes()

db.mycollection.dropIndex('xxxxxx')

Lokaali asennus

cd "C:\Program Files\MongoDB\Server\4.0\bin"

mongod --help - listaa käytettävissä olevat komennot

mongod --dbpath "C:\data\db" - Asettaa tietokannan löytymään eri paikasta

mongod - Käynnistää MongoDB tietokannan

Toimii jos näkyy: [initandlisten] waiting for connections on port 27017

mongo - Käynnistää mongo shellin

use omadb - luo uuden tietokannan nimeltä: omadb

Show dbs

db.omaCollection.insertOne({ x: 1 }); - luo uuden collectionin nimeltä omaCollection

```
db.createUser({user:"Antti",  
              pwd:"123",  
              roles: ["readwrite", "dbadmin"]  
            });
```

db.creteCollection('kauppa');

show collections

```
db.kauppa.insert({  
  first_name: "John",  
  last_name: "Doe",  
  address: {  
    street: "Katu 123",  
    city: "Helsinki"
```

```

    }
  });
db.kauppa.find().pretty();
db.kauppa.update({yksiloivald:"1234"},{first_name:"John", last_name: "Doe"}); - päivittää kaikki tiedot
db.kauppa.update({yksiloivald:"1234"},{$set:{gender:"male"}}); - muuttaa kentän ja lisää jos ei ole vielä olemassa
db.kauppa.update({yksiloivald:"1234"},{$unset:{gender:1}}); - poistaa kentän
db.kauppa.update({yksiloivald:"1234"},{$rename:{gender:"sex"}}); - muuttaa kentän nimen
db.kauppa.update({yksiloivald:"1234"},{$inc:{age:5}}); - lisää ikään 5 vuotta
  $mul: = multiply, $div: = division
db.kauppa.update({yksiloivald:"1234"},{$addToSet:{gender:"male"}}); - lisää dataa arrayhyn, jos ei ole vielä olemassa
db.kauppa.update({yksiloivald:"1234"},{$push:{new:"male"}}); - Lisää dataa kentän
db.kauppa.remove({yksiloivald:"1234"}); - poistaa tietueen
db.kauppa.remove({first_name:"John"}, {justOne:true}); - poistaa ensimmäisen löytämänsä
db.kauppa.find({first_name:"John"});
db.kauppa.find($or:[{first_name:"John"}, {first_name:"Jeff"}]);
db.kauppa.find({age:{$lt:40}}); - etsii alle 40 vuotiaat lt=less than
db.kauppa.find({age:{$gt:40}}); - etsii yli 40 vuotiaat gt=greater than
db.kauppa.find({age:"40"}).count; - montako tasan 40 v
db.kauppa.find().limit(4).sort({lastname:1});
db.kauppa.find().forEach(function(doc){print("Nimi on " + doc.fist_name)});

```

Mongoose

npm install mongoose --save

Mlab

<https://mlab.com/login/?user=antpent>

account name=antpentt, email= asjrj200@gmail.com, username=antpent, password=Karkula9!

Kanta: puhlu, Käyttäjä: mina, Salasana: pois99

mongodb://<dbuser>:<dbpassword>@ds257551.mlab.com:57551/puhlu

MongoDB Atlas

Cloud-hosted MongoDB service on AWS, Azure, and GCP.

Deploy, operate, and scale a MongoDB database in just a few clicks.

<https://www.mongodb.com/>

Data Modeling

Which approach to choose: Embedding or Referencing

- Frequent changes should be done to minimum number of documents
- Jos samassa collectionissa on useita eri luokkia, voidaan samalla haulla palauttaa useiden eri luokkien dataa
- Use keywords array[] for seraching large amounts of data
- When Metadata is larger than the data itself, partition it

Embedded

Typically leads to better READ performance

For things that are queried together

Dependencies

One-to-one relationship

One-to-few relationship

Data that changes about the same phase (Volatility = how often does the data change)

Hierarchical data

Referenced

One-to-many relationships

Many-to-many relationships

Large number of items

Data changes often (likes, comments, reviews, ...)

Combined

First view to data (snippet, summary, number of comments)

Larger set of data is retrieved only on demand

Git

Minun GitHub profiili

<https://github.com/Asjrj>

LOCAL GIT

git help	Help
git --version	Versio (git -v)
git init	Luo uuden repositoryn tästä hakemistosta
git init <myproject>	Luo uuden repositoryn ja luo uuden hakemiston
git add <file>	Lisää tiedoston repositoryn staging alueelle (muutettu)
git add .	Lisää kaikki muutokset version staging alueelle (muutettu)
git rm <file>	Poistaa tiedoston versiosta
git commit -m "muutos"	Tallettaa muutoksen eli version repositoryyn
git status	Näyttää repositoryn tilan
git log	Näyttää repositoryn lokin
git diff	Näyttää työversion erot staging areaan verrattuna
git checkout <114ca>	Palataan katsomaan versiota 114ca
git reset --hard <114ca>	Muutetaan versio 114ca masteriksi
git checkout -- .	Palataan tämän version alkuun
git branch	Näyttää missä haarassa ollaan
git checkout -b <haara>	Luodaan tästä uusi haara
git master	Siirrytään master haaraan
git <haara>	Siirrytään toiseen haaraan
git branch -D <haara>	Poistetaan haara
git merge <haara>	Yhdistetään haara masteriin

DISTRIBUTED GIT - GITHUB

Luo lokaali repository ja sitten - Luo repository Githubiin ja sitten - Yhdistä lokaali repository Github repositoryyn:

git remote add origin https://github.com/Asjrj/AYTKT21009.git	(Saattaa toimia myös luomalla etänä: git remote add AYTKT21009 https://github.com/Asjrj/)
git remote	Kertoo lokaalin github repositoryn
git remote -v	Kertoo remote github repositorioiden URL:t
git remote add origin https://github.com/Asjrj/react-app.git	
git push -u origin master	Tallettaa lokaalin master haaran GitHub repositoryyn (origin)
git branch -r	Näyttää github haaran
git clone https://github.com/Asjrj/hello-world.git	Kloonaa koko repositoryn (esim. toiselle koneelle tai käyttäjälle) (esim. ladattiin: https://github.com/FullStack-HY/redux-anecdotes)
git pull	Hakee viimeiset muutokset githubista (esim. jonkun muun käyttäjän tekemät)
git remote rm <nimi>	Poistaa Github repositoryn (https://github.com/Asjrj/hello-world.git)
git remote rm https://github.com/Asjrj/hello-world.git	

GitHub ohjeet repositoryn luomisen jälkeen

```
echo "# AYTKT21009_Blog" >> README.md
git init
git add README.md
```

```
git commit -m "first commit"
git remote add origin https://github.com/Asjrj/AYTKT21009\_Blog.git
git push -u origin master
```

```
git remote add origin https://github.com/Asjrj/node-app.git
https://github.com/Asjrj/node-app.git
```

Käyttäjän tietojen asetus

```
git config --global user.name 'Antti Penttinen'
git config --global user.email 'asjrj200@gmail.com'
git config -l                      Näyttää konfiguraation
git config --list                   Näyttää konfiguraation
```

```
.gitignore                         Lisää tiedostot tai hakemistot (/dir2), joita ei versioida
```

Ohje

<https://github.com/mluukkai/otm-2018/blob/master/tehtavat/viikko1.md#gitin-alkeet>

Angular JS

Angular JS by Google

Angular is a complete client side JavaScript **Framework** for single page applications: including Router and form validation. Angular uses TypeScript (by Microsoft)

Angular CLI = Command Line Interface

npm install -g @angular/cli - Asentaa Angularin

ng new projektin-nimi - Lue uuden projektin

ng serve - käynnistää serverin ja tekee ensimmäisen kännöksen

**** Angular Live Development Server is listening on localhost: 4200, open your browser on <http://localhost:4200/> ****

ng generate component uusiKomponentti - Luo tarvittavat tiedostot uudelle komponentille

= ng g c uusiKomponentti

Directives = instructions in the DOM

Inbuild directives: *ngIf (ja else) = structural directive

Attribute directives:

ngStyle="{backgroundColor: getColor()}",
ngClass,
ngFor="let x of lista"

Use:

Angular essentials extension for MS Visual Studio Code

Augury Chrome extension for detailed analysis what is going on at run time

Redux DevTools

Passing data from component A to Component B

Property binding

Listeners: RxJS Subject subscribe

Angular creates only the Client side - That's what it is good at

Use REST API for Backend - Always load the index.html from the server (for refresh)

Huom! stateless! Backendissä Ei ole sessiota

Tila voidaan hoitaa

Cookies, Local storage tai Tietokanta (paras)

Redux can be used with Angular

NgRx is an angular implementation of Redux

Deployment on vain Static tiedostojen paketoitua. Application serveriä ei tarvita.

Käyttöliittymän muotoiluun käytetään usein Bootstrap:iä

Quick Start

Projektin kokoonpano

GIT

Luo lokaali repository

```
git init
```

Tee Readme.md tiedosto

Tee .gitignore -tiedosto, jossa vähintään rivi
/node_modules

Luo repository Githubiin

Yhdistä lokaali repository Github repositoryyn:

```
git remote add origin https://github.com/Asjri/AYTKT21009\_backend.git
```

Lataa lokaali (origin) Github repositoryyn (master)

```
git push -u origin master
```

React

create-react-app mun-sovellus

npm install axios --save // Asenna tarvittavat lisäosat

npm start - Starts the development server

React kutsuu componentDidMount-metodia sen jälkeen kun konstruktori on suoritettu ja render-metodi on suoritettu ensimmäistä kertaa.

Redux

Käytetään Reduxia tilan hallintaan. Vain pienet sovellukset pärjäävät Reactin omalla tilan hallinnalla
React Context API ja Rect Hooks tulevat syrjäyttämään Reduxin

Browser

Local Storageen talletetut arvot säilyvät vaikka sivu uudelleen ladataan.

```
window.localStorage.setItem('nimi', 'juha tauriainen') - tallettaa avain, arvo parin
```

Node

npm init

Kysy Node projektin perustiedot ja lue konfiguraatio tiedoston package.json

Asenna nodemon ja lisää package.json konfiguraatioon "watch" -komento

```
npm install --save-dev nodemon
```

```
"watch": "nodemon index.js"
```

bodyParser tarvitaan, jotta voidaan lukea request.body

Hakemistot:

Controlllers: Express reititys moduulit

Models: Tietokanta määrittelyt: yhteydet, schema ja model

Express

npm install express --save

MongoDB ja Mongoose

Asenna mongoose

```
npm install mongoose --save
```

Tee ympäristömuuttuja, johon talletetaan tietokantayhteys (käyttäjä sekä salasana)

```
BLOG_DB_URL=mongodb://mina:pois99@ds235053.mlab.com:35053/bloglist
```

Testaus

Tee hakemisto

Requests

Tee tiedosto esim: get_persons.rest. Laita sinne komento:

GET <http://localhost:3001/api/persons>

Yksikkötestejä varten asenna esim jest

npm install --save-dev jest

Ja tee testejä hakemistoon "tests"

Lisää package.json tiedostoon komento:

"test": "jest --verbose"

"jest": { "testEnvironment": "node" }

Tietyt testit voi ajaa yksi kerrallaan:

npx jest -t 'ajetaan, jos tämä string löytyy'

Backendin API:en testaamiseen tarvitaan: supertest

npm install --save-dev supertest

Fontendin yksikkötestaukseen käytetään Enzyment shallow tai mount komponenttia

npm install --save-dev enzyme enzyme-adapter-react-16

Fontendin integraatiotestissä käytetään Jestin Manula Mock menetelmää: hakemisto __mocks__

jest.mock('./services/blogs')

Testauskattavuus saadaan helposti selville suorittamalla testit komennolla

CI=true npm test -- --coverage

Työkaluja

Git Bash terminal

Avaa sovellushakemisto terminaalin

Visual Studio Code (VSCode)

Avaa sovellushakemisto VSCodeen

Asenna extensionit: Rest client, ...

Node

Nodemon: Käynnistää Node serverin kuun source koodi muuttuu

ESLint: Tarkistaa koodin - koodin staattinen analyysi

npm install eslint --save-dev

ESLint-konfiguraatio muodostetaan komennolla

node_modules/.bin/eslint --init

SAAS

Mlab for Mongo

<https://mlab.com/login/?user=antpent>

account name=antpentt, email= asjrj200@gmail.com, username=antpent, password=Karkula9!

Tee uusi kanta:

Deployment: puhlu

Tee kannalle käyttäjä: mina, Salasana: pois99

To connect using a driver via the standard MongoDB URI:

mongodb://mina:pois99@ds257551.mlab.com:57551/puhlu

Zeit Now

now --public -e MONGO_URL=mongodb://mina:pois99@ds257551.mlab.com:57551/puhlu

Help

REST API

Visual Studio Codessa, voit postmanin sijaan käyttää VS Coden [REST client](#) -pluginia

CORS

Cross-origin resource sharing (CORS) on selainten turvallisuus ominaisuus.

[Wikipedia](#) Websovelluksen selaimessa suoritettava Javascript-koodi saa oletusarvoisesti kommunikoida vain samassa originissa olevan palvelimen kanssa. (eri portti = eri origin)
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

Palvelimella voidaan listata hyväksyttävät originit:

```
res.header("Access-Control-Allow-Origin", "http://localhost:8000");
res.header("Access-Control-Allow-Origin", $ORIGIN);
$ORIGIN = if(inWhitelist(requestOriginHeader) return requestOriginHeader
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
Access-Control-Allow-Credential: true
```

Nodelle voi asentaa cors -middlewareen (<https://github.com/expressjs/cors>)

```
npm install cors --save
```

Koodissa:

```
const cors = require('cors')
app.use(cors())
```

Cors:in voi konfiguroida hoitamaan

Voi tehdä myös oman middlewaren

```
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", '*');
  if (req.method === 'OPTIONS'){
    res.header('Access-Control-Allow-Methods', 'GET, PUT, POST');
    return res.status(200).json({});
  }
  next();
});
```

JSON server

```
npm install json-server --save
```

Lisätään tiedoston package.json osaan scripts rivi

```
"scripts": {
  "server": "json-server -p3001 db.json",
}
```

Käynnistetään komennolla: npm run server.

Template

a

Bootstrap

React Bootstrap

<https://react-bootstrap.github.io/getting-started/introduction>

```
npm install --save react-bootstrap
```

Lisää public/index.html tiedoston head-tagin sisään bootstrapin css-määrittelyt lataava rivi:

```
<link rel="stylesheet"
```

```
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
```

```
crossorigin="anonymous">
```

Importoi käytettävät react-bootstrap kirjaston komponentit

```
import { FormGroup, ControlLabel, FormControl, Button } from 'react-bootstrap'
```

Semantic ui

Semantic ui on toinen käyttöliittymäkirjasto

Lint

ESLINT

Javascript-maailmassa tämän hetken johtava työkalu staattiseen analyysiin on **ESLint**.

Monissa yrityksissä määritellään yrityksen laajuiset koodausstandardit ja näiden käyttöä valvova ESLint-konfiguraatio. Viime aikoina monissa projekteissa on omaksuttu Airbnb:n Javascript-tyyliohjeet: <https://github.com/airbnb/javascript>

ESLint asennetaan backendiin kehitysaikaiseksi riippuvuudeksi komennolla

```
npm install eslint --save-dev
```

Tämän jälkeen voidaan muodostaa alustava ESLint-konfiguraatio komennolla

```
node_modules/.bin/eslint --init
```

npm-skripti linttausta varten:

```
{
  "scripts": {
    "start": "node index.js",
    "watch": "nodemon index.js",
    "lint": "eslint ."
  },
}
```

Tee projektin juureen tiedosto .eslintignore ja lisää sinne hakemistot, joita ei tarkasteta

Visual Studioon kannattaa ladata ESLint-plugin:

<https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint>

ESLintille on määritelty suuri määrä sääntöjä, joita on helppo ottaa käyttöön muokkaamalla tiedostoa .eslintrc.js.

Esim sääntö eqeqeq varoittaa, jos koodissa yhtäsuuruutta verrataan muuten kuin käyttämällä kolmea = -merkkiä.

```
module.exports = {
  "env": {
    "browser": true,
    "es6": true,
    "node": true,
    "jest/globals": true
  },
  "extends": [
    "eslint:recommended",
    "plugin:react/recommended"
  ],
  "parser": "babel-eslint",
  "parserOptions": {
    "ecmaFeatures": { "jsx": true },
    "ecmaVersion": 2018,
    "sourceType": "module"
  },
  "plugins": [
    "react", "jest"
  ],
  "rules": {
    "indent": [ "error", 2 ],
```

```
"linebreak-style": [ "error", "windows" ],
"quotes": [ "error", "single" ],
"semi": [ "error", "never" ],
"eqeqeq": "error",
"no-trailing-spaces": "error",
"object-curly-spacing": [ "error", "always" ],
"arrow-spacing": [ "error", { "before": true, "after": true } ],
"no-console": 0,
"react/prop-types": 0
}
};
```

ESlint ja REACT (front end)

Create-react-app on asentanut projektille eslintin valmiiksi, joten ei tarvita muuta kuin .eslintrc.js.

Tiedoston voi generoida komennolla

```
npx eslint --init
```

Testeissä olevista turhista huomautuksista pääsee eroon asentamalla eslint-jest-plugin:

```
npm add --save-dev eslint-plugin-jest
```

ESlint ei vielä osaa tulkita uutta syntaksia kunnolla.

Jotta ESLint osaisi tulkita class property -syntaksia, asennetaan myös babel-eslint-plugin:

```
npm install babel-eslint --save-dev
```


Webpack

Vaikka ES6-moduulit ovatkin Javascript-standardissa määriteltyjä, ei mikään selain vielä osaa käsitellä moduuleihin jaettua koodia (2018)

Webpack asennetaan:

```
npm install --save-dev webpack webpack-cli
```

Komennolla **npm run build** npm-skripti suorittaa bundlauksen webpackia käyttäen. Tuloksena on joukko hakemistoon **build** sijoitettavia tiedostoja. **Package.json** tiedostoon määritellään build -skripti

```
"scripts": {  
  "build": "webpack --config=webpack.config.js"  
},
```

[webpack.config.js](#) -tiedoston sisältö:

```
const path = require('path')  
const config = {  
  entry: './src/index.js',  
  output: {  
    path: path.resolve(__dirname, 'build'),  
    filename: 'main.js'  
  }  
}  
module: {  
  rules: [  
    {  
      test: /\.js$/,  
      loader: 'babel-loader',  
      query: {  
        presets: ['env', 'react']  
      }  
    },  
    {  
      test: /\.css$/,  
      loaders: ['style-loader', 'css-loader']  
    }  
  ]  
}  
module.exports = config
```

Huom !

`__dirname` on Noden globaali muuttuja, joka viittaa nykyiseen hakemistoon
Koska Webpack ei ymmärrä Reactin JSX:ää, pitää konfiguroida Reactin käyttämän JSX:n normaaliksi Javascriptiksi muuntava loaderi. Loaderi ja sen tarvitsemat kirjastot asennetaan kehitysaikaiseksi riippuvuudeksi:

```
npm install --save-dev babel-core babel-loader babel-preset-react
```

CSS:ää varten on otettava käyttöön `css-` ja `style-`loaderit:

```
npm install style-loader css-loader --save-dev
```

Transpilauus

ES6, ES7 ja uudempi Javascript koodi voidaan transpilata ES5 tason koodiksi, jotta kaikki Browserit

ymmärtävät sitä.

Asennetaan preset 'env', joka sisältää kaiken hyödyllisen, minkä avulla uusimman standardin mukainen koodi saadaan transpiloitua ES5-standardin mukaiseksi koodiksi:

```
npm install babel-preset-env --save-dev
```

Webpack 4

entryllä ja outputilla on Webpack versiossa 4 oletusarvo, eli ne voi jättää määrittelemättä

Versiossa on muuotksia, minkä vuoksi tarvitaan:

```
npm install --save-dev @babel/core
```

```
npm install --save-dev @babel/preset-env
```

```
npm install --save-dev @babel/preset-react
```

```
npm install --save-dev @babel/plugin-proposal-class-properties
```

```
npm install --save-dev @babel/polyfill
```

Testaus

Backend = Node

Testit ajetaan Jest:llä

<https://jestjs.io/>

Testitiedoston nimi tulee olla muotoa: tiedosto.test.js

```
describe.only('UUSI LOHKO ', () => {  
  test('UUSI TESTI', () => {  
    let x = 1  
    expect(x).toBe(1)  
  })  
})
```

Noden API:n testaamiseen käytetään Jestin apuna supertest-kirjastoa

```
"scripts": {  
  "test": "cross-env NODE_ENV=test jest --verbose"  
},  
"devDependencies": {  
  "jest": "^23.6.0",  
  "supertest": "^3.3.0"  
},  
"jest": {  
  "testEnvironment": "node"  
}
```

Mongo-mock kirjastoa, voidaan käyttää MongoDB:n simulointiin.

<https://github.com/williamkapke/mongo-mock>

Front End = React ja Redux

Jest on valmiiksi konfiguroitu create-react-app:illa luotuihin projekteihin.

Jestin lisäksi käytetään AirBnB:n kehittämää enzyme-kirjastoa.

```
npm install --save-dev enzyme enzyme-adapter-react-16
```

Enzyme testit on tapana tehdä samaan hakemistoon, missä komponentitkin sijaitsevat

Jos komponentille tehdään yksikkötestejä, shallow-renderöinti on useimmiten riittävä.

Frontendin integraatiotestausta varten käytetään komponentit kokonaisuudessaan renderöivään mount:iin

End 2 End

Web-sovellusten E2E-testaus tapahtuu simuloidun selaimen avulla esimerkiksi Selenium-kirjastoa käyttäen.

Toinen vaihtoehto on käyttää ns. headless browseria eli selainta, jolla ei ole ollenkaan graafista käyttöliittymää.

Puppeteer kirjastoa käytetään headless testaukseen

Heroku, Zeit, MLAB

Heroku

<https://devcenter.heroku.com/articles/getting-started-with-nodejs>
<https://devcenter.heroku.com/articles/git#creating-a-heroku-remote>

Projektin juureen pitää tehdä tiedosto Procfile, joka kertoo Herokulle, miten sovellus käynnistetään
web: node index.js

CMD tool

Mene projektin juureen

git remote -v Listaa remote repositoryt

Heroku login

git add .

git commit -m "Tähän kommentti"

git push heroku master

heroku config:set BLOG_DB_URL=mongodb://test:pois99

@ds135653.mlab.com:35653/blogtest

heroku config:set BLOG_SECRET=salainen_avain

heroku run bash - Avaa Terminaalin

AYTKT21009 Blog sovellus

<https://fierce-scrubland-87231.herokuapp.com/>

hello-world sovellus

<https://protected-brook-40661.herokuapp.com/> | <https://git.heroku.com/protected-brook-40661.git>

Zeit Now

All we have to do now is run now from within the my-app directory:

now --public -e MONGO_URL=mongodb://mina:pois99@ds257551.mlab.com:57551/puhlu

now --public -e BLOG_SECRET=salainen_avain -e BLOG_PORT=3003 -e

BLOG_DB_URL=mongodb://mina:pois99@ds235053.mlab.com:35053/bloglist

now --public -e BLOG_SECRET=salainen_avain -e BLOG_PORT=3003 -e

BLOG_DB_URL=mongodb://test:pois99@ds135653.printenv.mlab.com:35653/blogtest

Blog lista:

Puhelinluettelo: <https://puhback-bwjcktktmz.now.sh>

Lista ihmisistä: <https://puhback-yerghltpig.now.sh/api/persons/>

Lista ihmisistä: <https://puhback-qyzxnjrnl.now.sh/api/persons>

aytk21009blog-yekvkhvsw.now.sh

Mlab

<https://mlab.com/login/?user=antpent>

account name=antpentt, email= asjrj200@gmail.com, username=antpent, password=Karkula9!
Kanta: puhlu, Käyttäjä: mina, Salasana: pois99

To connect using the mongo shell:

```
mongo ds257551.mlab.com:57551/puhlu -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI:

```
mongodb://<dbuser>:<dbpassword>@ds257551.mlab.com:57551/puhlu
```

```
mongodb://<dbuser>:<dbpassword>@ds133533.mlab.com:33533/puhettelo
```

```
mongodb://<dbuser>:<dbpassword>@ds235053.mlab.com:35053/bloglist
```

```
mongodb://<dbuser>:<dbpassword>@ds135653.mlab.com:35653/blogtest
```

Create user with password

```
mongodb://mina:pois99@ds257551.mlab.com:57551/puhlu
```

```
mongodb://mina:pois99@ds133533.mlab.com:33533/puhettelo
```

```
mongodb://mina:pois99@ds235053.mlab.com:35053/bloglist
```

```
mongodb://test:pois99@ds135653.mlab.com:35653/blogtest
```

```
heroku config:set BLOG_DB_URL=mongodb://test:pois99@ds135653.mlab.com:35653/blogtest
```

GraphQL

GraphQL:ssä aina lähetetään POST request yleensä osoitteeseen: /graphql

Haku eli query lähetetään request bodyn mukana. Esim:

```
{
  query {           // Operation type
    user {          // Operation endpoint
      name,         // Field 1
      age           // Field 2
    }
  }
}
```

GraphQL operation types:

- Query tiedon hakemiseksi

- Mutation tiedon muuttamiseksi

- Subscriptions realtime websocket connection

resolvers = logiikka, jolla data haetaan (ovat kuin controllers)

Template

Frontent react-app

Deployed ??

Frontend in GitHub

<https://github.com/Asjrj/react-app>

git push -u origin master

UI

<https://material-ui.com/>

Backend node-app

Deployed in Heroku <https://agile-temple-75130.herokuapp.com/>

asjrj200@gmail.com Karkula9!

<https://www.heroku.com/>

<https://devcenter.heroku.com/articles/getting-started-with-nodejs>

Deploy in command tool

git remote -v Listaa remote repositoryt
heroku login Avaa browserin ja varmistaa käyttäjätilin (login)
git add . Lisätään kaikki tiedostot gittiin
git commit -m "Kommentti" Kommitoidaan oma git repositorio
Herokussa pitää olla tehtynä Heroku remote, jonka nimeksi tulee automaattisesti: heroku
Heroku remote luodaan komennolla: heroku create =>

<https://devcenter.heroku.com/articles/git#creating-a-heroku-remote>

C:\a\node-app>heroku create

» Warning: heroku update available from 7.18.10 to 7.19.4

Creating app... done, ● agile-temple-75130

<https://agile-temple-75130.herokuapp.com/> | <https://git.heroku.com/agile-temple-75130.git>

Deploy into Heroku

Push the code from your local repository's **master** branch to your **heroku** remote:

git push heroku master

remote: -----> Launching...

remote: Released v3

remote: <https://agile-temple-75130.herokuapp.com/> deployed to Heroku

remote:

remote: Verifying deploy... done.

To <https://git.heroku.com/agile-temple-75130.git>

* [new branch] master -> master

Backend in GitHub

<https://github.com/Asjrj/node-app>

git push -u origin master

API reititys

http operation tyytit (verbit)

api/data/10 GET hakee yksittäisen resurssin

api/data	GET	hakee kokoelman kaikki resurssit
api/data	POST	luo uuden resurssin pyynnön mukana olevasta datasta
api/data/10	DELETE	poistaa yksilöidyn resurssin
api/data/10	PUT	korvaa yksilöidyn resurssin pyynnön mukana olevalla datalla
api/data/10	PATCH	korvaa yksilöidyn resurssin osan pyynnön mukana olevalla datalla

Mongo Database

Deploy in Mongo Atlas !! *** Ei vielä tehty

Tietokanta ja Collection

Database = mydb

Collection = mycustomers

Collection = myorders

Collection = myproducts

Customer + Address

```
db.mycustomers.insertOne( {id: 123, name:'Tony Thomson', address: {street:'Katu', zip: '02150',
city:'Espoo'} })
db.mycustomers.insertMany([
{id: 124, name:'Max Smith', email: 'Max.Smith@email.com', address: {street:'Katu 5', zip: '02150',
city:'Espoo'}},
{id: 125, name:'John Wilson', email: 'John.Wilson@email.com', address: {street:'Polku 19', zip:
'02150', city:'Espoo'}},
{id: 126, name:'Ben Johnson', email: 'Ben.Johnson@email.com', address: {street:'Kuja 23', zip:
'02170', city:'Helsinki'}},
{id: 127, name:'David Cooper', email: 'David.Cooper@email.com', address: {street:'Kuja 24', zip:
'02170', city:'Helsinki'}}
])
db.mycustomers.update({id: 123},{ $set:{name:'Tony Thomson'}})
db.mycustomers.find({name:'Max Smith'}, {"address.street": 1})
db.mycustomers.distinct('name')
```

Products

```
db.myproducts.insertMany([
{id: 1001, class: 'Computers', make: 'Sony', model: 'XYZ200', price:800, instock: 55},
{id: 1002, class: 'Phones', make: 'Samsung', model: 'S8', price:300, instock: 65},
{id: 1003, class: 'Phones', make: 'Nokia', model: 'Lumia500', price:400, instock: 45}
])
```

Orders

```
db.myorders.insertOne(
{customerid: {$ref: 'mycustomers', $id: ObjectId("5c61234d2d2cdc368149d253")}, $db: 'mydb'},
date: new Date(),
products:[{$ref: 'myproducts', $id: ObjectId("5c6161a72d2cdc368149d257"), $db: 'mydb'},
{$ref: 'myproducts', $id: ObjectId("5c6161a72d2cdc368149d258"), $db: 'mydb'}]
})
db.myorders.insertOne(
{customerid: {$ref: 'mycustomers', $id: ObjectId("5c6137ba2d2cdc368149d254")}, $db: 'mydb'},
date: new Date(),
products:[{$ref: 'myproducts', $id: ObjectId("5c6161a72d2cdc368149d259"), $db: 'mydb'}]
})
```


Events, Comments, Contact info, ...

Hakuja

```
db.mycustomers.find().pretty()
```

Descriptions

AWS AppSync

AWS AppSync is a fully managed, serverless GraphQL service that fast-tracks your API development. You can define a single type and AppSync auto-generates a schema, queries, mutators, and subscriptions for you.

PWA

Progressive Web Apps allow a user to use your web app online or offline, and lets them install the app onto their iOS or Android device just like a native app. (React, Node)