IŠ PRAEITOS PASKAITOS

- Jeigu atrodo, kad NIEKO nesuprantante
  - eikite po vieną skraidrę, darykite copy/paste, sekite instrukcijas, skaitykite, kas parašyta
    - jeigu ir tada neaišku - klauskite. Drąsiai!
- Jeigu atrodo, kad VISKAS suprantama (lyg ir)
  - darykite tą patį
  - atrasite, kad vis tik kažkas nesigauna
- Supratimas ateis, kai viską padarysite PATYS
- Kuo daugiau klausinėsite, tuo daugiau pateiksiu atsakymų

IŠ PRAEITOS PASKAITOS

- Norint panaudoti PropTypes su naujausiu React 16, reikalinga

```
import PropTypes from 'prop-types';
```

- ir į package.json įsidėti

```
"prop-types": "^15.6.2"
```

- kaip teisingai pastebėjote, ES5 React 15.x turi `React.createClass()`, o React 16.x reikalinga `create-react-class` biblioteka ir turi `createReactClass()`
  - mes naudosime ES6 React 16.x `React.Component`

IŠ PRAEITOS PASKAITOS

- Naujo komponento šablonas su PropTypes už klasės

```
import PropTypes from 'prop-types';

class Component extends React.Component {
  render() {
    return (
      <div>
        <!-- Component view -->
      </div>
    );
  }
};

Component.propTypes = {
  // Properties JSON
};
```

IŠ PRAEITOS PASKAITOS

- Naujo komponento šablonas su PropTypes už klasėje

```
import PropTypes from 'prop-types';

class Component extends React.Component {
  static propTypes = {
    // Properties JSON
  }

  render() {
    return (
      <div>
        <!-- Component view -->
      </div>
    );
  }
};
```

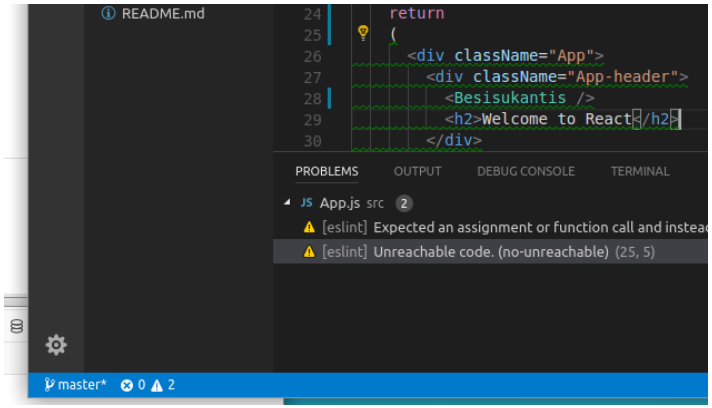
IŠ PRAEITOS PASKAITOS

- ESLint - suponuoja taisykles, kaip reikia rašyti kodą
  - jei klaidose matote žodį LINT tai galbūt nepadėjote kur nors kabliataškio,
  - gal negalima iš naujos eilutės naudoti JSX skliaustelio
  - gal importuotą biblioteką reikia būtinai panaudoti arba neimportuoti
  - susikurtą kintamąjį reikia būtinai panaudoti

ESLINT

- Pavyzdys kaip sukonfigūruoti VSCode ESLint extension:
  - Ctrl+Shift+X , suieškoti ESLint, install ir reload
  - būtent šiam pluginui reikia į package.json įdėti:

```
"eslintConfig": {
  "extends": "react-app"
}
```



IŠ PRAEITOS PASKAITOS

- turite parašyti kodą patys
  - kitu atveju tiesiog bus per daug medžiagos, kad ją būtų įmanoma išmokti nerašius kodo

AKADEMIJA.IT

INFOBALT IR TECH CITY

JAVASCRIPT MODULIAI. ARROW/MAP/FILTER/REDUCE/KOLEKCIJOS

Andrius Stašauskas

andrius@stasauskas.lt

http://stasauskas.lt/itpro2018/

TURINYS

- Javascript moduliai
- Map/filter/Reduce
- Arrow funkcijos
- Kolekcijos

JAVASCRIPT MODULIAI

NPM - NODE PACKAGE MANAGER

- npm projektas - package.json
  - name
  - dependencies
  - devDependencies
  - scripts
- src
- public
- node_modules

NPM - NODE PACKAGE MANAGER

package.json

```
{
  "name": "hello-world",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.6.3",
    "react-dom": "^16.6.3",
    "react-scripts": "2.1.1",
    "prop-types": "^15.6.2",
    "bootstrap": "4.1.3"
  },
  "devDependencies": {},
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
```

VERSIJŲ FIKSAVIMAS

- užfiksuoti versijas, kad nereiktų migruoti kodo, vietoj

```
"dependencies": {
  "react": "^16.6.3",
  "react-dom": "^16.6.3",
  "react-scripts": "2.1.1",
  "prop-types": "^15.6.2",
  "bootstrap": "4.1.3"
}
```

- naudoti (nors ^ ir reiškia compatible)

```
"dependencies": {
  "react": "16.6.3",
  "react-dom": "16.6.3",
  "react-scripts": "2.1.1",
  "prop-types": "15.6.2",
  "bootstrap": "4.1.3"
}
```

NPM - NODE PACKAGE MANAGER

- pagrindinės npm komandos kurias naudosime:

```
$ npm install
  // dirba su package.json
  // https://docs.npmjs.com/cli/install
$ npm install <paketas>
$ npm install -g <paketas>
$ npm uninstall
$ npm install <paketas>@<versija>
$ npm help <komanda>
$ npm run <scripts-komanda>
$ npm run start // = npm start
```

KLASĖ

```
class Polygon {
  constructor(height=2, width=3) {
    this.height = height;
    this.width = width;
  }
  get area() {
    return this.calcArea()
  }
  calcArea() {
    return this.height * this.width;
  }
}
var poly = new Polygon; // galima be skliaustų
console.log(poly.calcArea());
console.log(poly.area); // nėra skliaustų
polygon.area = 3; // negalime - reiktų sukurti set area() metodą
```

NAUJAS MODULIS

Sukuriamas src kataloge `Modulis/Modulis.js`

```
class Polygon { /* ... */ }
export var P1 = Polygon;
export var P2 = Polygon;
export default Polygon;
```

Babel'is konvertuoja į CommonJS formatą, pvz.:

```
var Polygon = function Polygon() { /* ... */ }
module.exports = Polygon;
module.exports.P1 = Polygon;
module.exports.P2 = Polygon;
```

BABEL KOMPILIATORIUS

- Konvertuoja JS kodą iš vieno formato į kitą
- Dažniausiai naudojamas konvertuoti iš ES2015 į ES5 formatą
- Arba TypeScript kodą į ES2015, o tada į ES5 ir t.t.
- Tiesiai nenaudosim; galim paleisti iš `node_modules`

```
$ npm install --save-dev @babel/core @babel/cli @babel/preset-env
$ node ./node_modules/.bin/babel src/babel-test.js
  --out-file compiled.js --presets=@babel/env
```

- Arba https://es6console.com/

BABEL

Konvertuoja `src/babel-test.js` iš ES2015

```
class Person {}
var dave = new Person
```

į `compiled.js`

```
"use strict";
function _classCallCheck(instance, Constructor) {
if (!(instance instanceof Constructor)) {
  throw new TypeError("Cannot call a class as a function"); } }
var Person = function Person() {
  _classCallCheck(this, Person);
};
var dave = new Person();
```

MODULIŲ UŽKROVIMAS

index.js

```
import {P} from './Modulis/Modulis';
import {P as Pavadinimas} from './Modulis/Modulis';
import Polygon from './Modulis/Modulis';
import Polygon, {P} from './Modulis/Modulis';
```

Iš tikrųjų babel'is konvertuoja iš ES2015

```
import express from 'express';
```

į CommonJS require formą

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

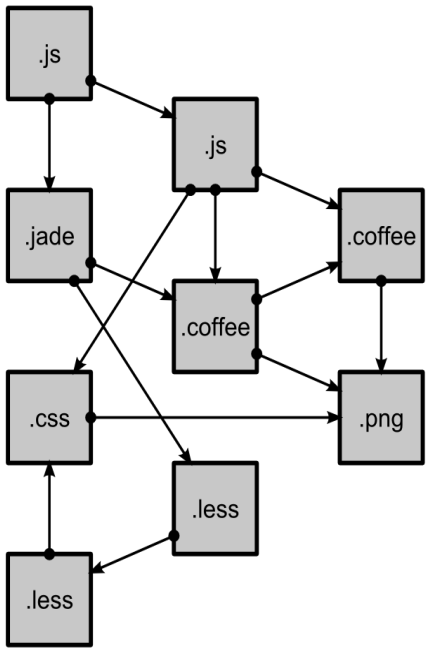
MODULIAI - GERA PRAKTIKA

- Atskiras komponentas - atskiram modulyje
- Atskiras modulis - savo kataloge
- Panašu į java packages ir import
- Visada `export default`
  - pagrindinis funkcionalumas
  - pagrindinė klasė
- Papildomos klasės, interfeisai ar konstantos
  - `export var`

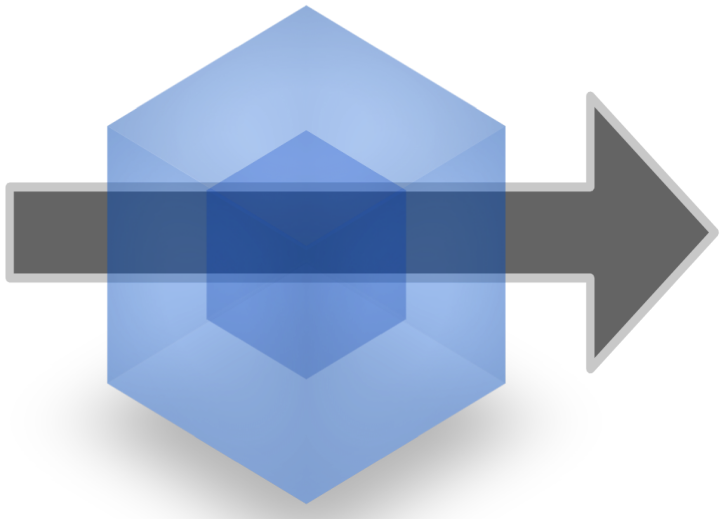
MODULIŲ PAKAVIMAS - WEBPACK

- Kas atliks visus tuos konvertavimus?
- Kas pasirūpins, kad viskas atsidurtų reikiamuose kataloguose serveryje?
- Kaip naršyklė supras mūsų "advanced" technologijas?
  - nesupras. Bent jau ne visas ir ne šiandien
- Webpack leidžia apjungti visas technologijas
  - naudojama viskas advanced, latest, etc. ir paduodama webpack'ui
  - webpack sukuria gražų ES5 kodą ir failų struktūrą, kurią supranta naršyklė

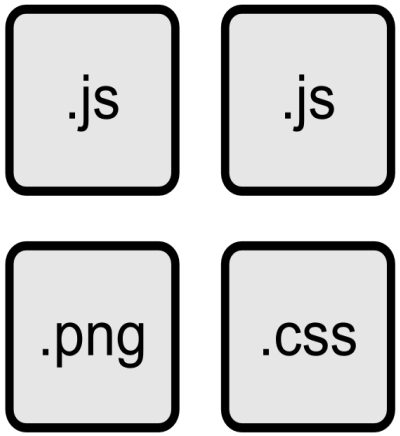
KAS YRA WEBPACK



modules
with dependencies



webpack
MODULE BUNDLER



static
assets

NAUJO KOMPONENTO ŠABLONAS

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

class Komponentas extends Component {
  render() {
    return (
      <div>
        {/* Component view */}
      </div>
    );
  }
};
Komponentas.defaultProps = { /* Properties JSON */ };
Komponentas.propTypes = { /* Properties JSON */ };
```

UŽDUOTIS 1

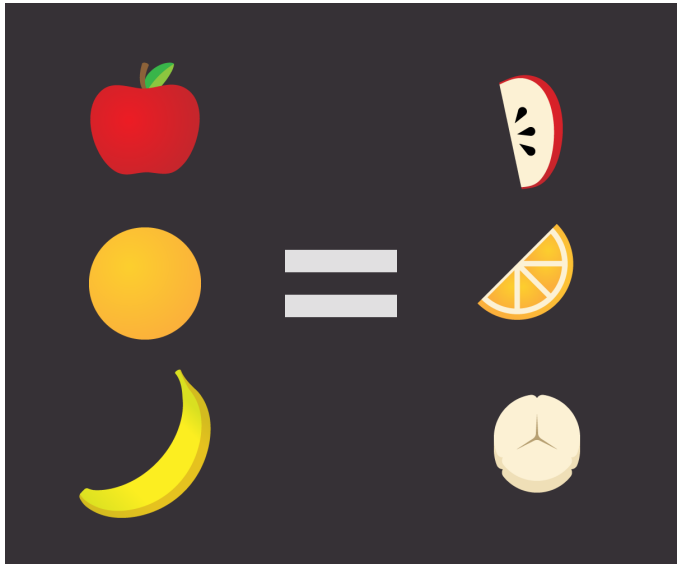
- Konvertuoti praeitoje paskaitoje padarytus ProductCardComponent ir ProductListComponent į naująją projekto struktūrą
  - react-create-app struktūra
  - Card ir List komponentai atskiruose kataloguose ir failuose
  - App komponentas - irgi savo modulyje

ATVAIZDIS, FILTRAS IR REDUKCIJA

ATVAIZDIS, FILTRAS IR REDUKCIJA

- Atvaizdis (map), filtras (Filter) ir redukcija (Reduce) naudojami elementų masyvą paversti į kažką kitą

ATVAIZDIS (MAP)



- Turime masyvą elementų ir norime transformuoti kiekvieną iš elementų.
- Rezultatas būtų naujas tokio paties dydžio masyvas, kuriame yra pakeisti elementai

ATVAIZDIS (MAP)

```
var people = [{
    name: 'Antanas Atanaitis',
    drives: 'Car',
    team: 'Studentai'
  }, {
    name: 'Petras Petraitis',
    drives: 'Truck',
    team: 'Moksleiviai'
  }, {
    name: 'Vilkas Vilkaitis',
    drives: 'Formula 1',
    team: 'Studentai'
  }, {
    name: 'Vilija Vilimaitė',
    drives: 'Car',
    team: 'Moksleiviai' } ];
```

ATVAIZDIS (MAP)

```
map(callback(item));
map(callback(item[, index]));
```

Su funkcija:

```
function getDrives(pupil) {
  return pupil.drives;
}
var automobiles = people.map(getDrives);
```

ATVAIZDIS (MAP)

Su anonimine (be vardo) funkcija:

```
automobiles = people.map(function(pupil) {
  return pupil.drives;
});

console.log(automobiles);
```

ATVAIZDIS (MAP)

Su arrow funkcija:

```
automobiles = people.map(pupil => pupil.drives);

console.log(automobiles);
```

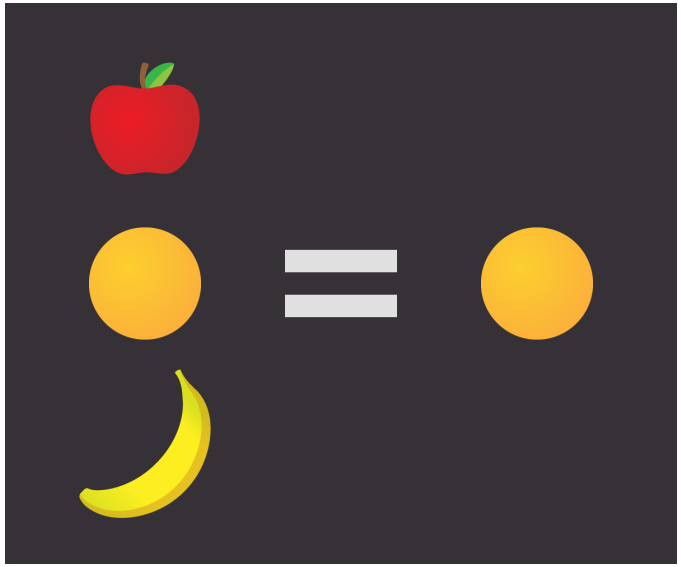
ARROW FUNKCIJA

- Storosios rodyklės funkcija

```
param1 => param1;
param1 => { return param1; }
param1 => param1 * param1;
(param1, param2) => param1 + param2;
param1 => ({atributas: param1});
() => false;
var f = () => false;
var f = () => ({false}); // https://es6console.com/
```

- Svarbu: arrow funkcija išlaiko leksikologinį kontekstą - jos viduje veikia išorinio konteksto nuoroda `this`
  - todėl nebereikia "hack'ų" `bind(this)` ir `var self = this`

FILTRAS (FILTER)



- Išrinkti iš masyvo elementus. Rezultatas yra naujas masyvas su visais elementais, išskyrus išfiltruotus
- Naujo masyvo ilgis toks pats kaip senojo jeigu niekas neišfiltruojama arba trumpesnis

FILTRAS (FILTER)

```
filter(callback(element));
filter(callback(element[, index]));
```

pvz.:

```
var studentai = people.filter(pupil => pupil.team === 'Studentai');

console.log(studentai);
```

ATVAIZDIS IR FILTRAS

be formatavimo

```
studentai = people.filter(pupil => pupil.team === 'Studentai').map(pu

console.log(studentai);
```

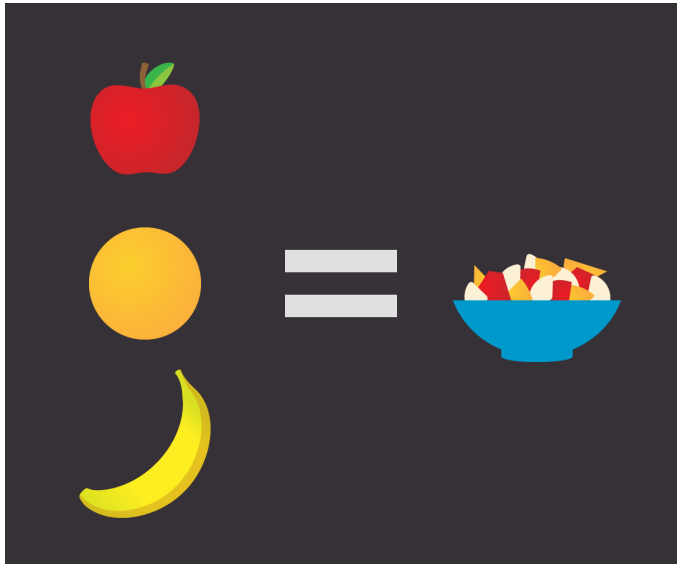
ATVAIZDIS IR FILTRAS

su formatavimu

```
studentai = people
              .filter(pupil => pupil.team === 'Studentai')
              .map(pupil => pupil.name);

console.log(studentai);
```

REDUKCIJA (REDUCE)



- Turint elementų masyvą galima apskaičiuoti naują reikšmę pereinant per kiekvieną iš elementų
- Rezultatas gali būti bet kas: kitas masyvas, naujas objektas, true/false reikšmė ir pan.

REDUKCIJA (REDUCE)

```
reduce(callback(accumulator, currentValue));
reduce(callback(accumulator, currentValue[, currentIndex])[, initialV
```

pvz.:

```
var names = people.reduce(function(sum, pupil) {
  return sum + ', ' + pupil.name;
});

console.log(names);
```

REDUKCIJA (REDUCE)

```
names = people.reduce(function(sum, pupil, currentIndex) {
  if (currentIndex === 0) {
    return sum + '' + pupil.name;
  } else {
    return sum + ', ' + pupil.name;
  }
}, "Vardai: ");

console.log(names);
```

REDUKCIJA (REDUCE)

```
names = people.reduce((sum, pupil) => sum + ', ' + pupil.name);

console.log(names);
```

REDUKCIJA (REDUCE)

```
names = people
          .reduce((sum, pupil, currentIndex) =>
             sum + (currentIndex > 0 ? ', ' : '') + pupil.name,
             "Vardai: ");

console.log(names);
```

FILTRAS, ATVAIZDIS IR REDUKCIJA

```
studentai = people
              .filter(pupil => pupil.team === 'Studentai')
              .map(pupil => pupil.name)
              .reduce((vardai, vardas) => vardai + ' ' + vardas);

console.log("Studentai:", studentai);
```

UŽDUOTIS 2

- Sukuriame prekių sąrašo masyvą
- Elementai - produktai
- Produktą apibūdinantys laukai: title, imageUrl, description, price, quantity
- Prikuriame produktų su kaina nuo 1 eur iki 100 eur
- Sukuriam vaizdinį be imageUrl ir be description
- Išfiltruojam prekes, kurių kaina mažesnė negu 10
- Redukuojame prekių masyvą į vieną skaičių - visų prekių kainų, padaugintų iš kiekio, sumą
  - visai kaip krepšelio galutinė suma

ES6 KOLEKCIJOS

ES5 KOLEKCIJOS

- Masyvai
- Objektai
- Simbolių eilutės (string)
- Pseudo-masyvas - argumentai funkcijoje

ES5 PROBLEMOS

- Masyve elementai neunikalūs
- Objektų raktai (key) verčiami į string
- turi papildomų savybių, tokių kaip toString, **proto**

ES6 KOLEKCIJOS

- Aibė (Set)
- Žemėlapis (Map)
- WeakMap (silpnas)

AIBĖ (SET)

- Unikalios reikšmės (tiek primityvios, tiek objektai)
- Išsaugo tvarką

```
const aibe = new Set(); // Set [ ]
aibe.add('vardas'); // Set [ "vardas" ]
aibe.add('miestas'); // Set [ "vardas", "miestas" ]
aibe.add('miestas'); // neįdės - dublikatas
aibe.has('miestas'); // true
aibe.delete('vardas'); // Set [ "miestas" ]
aibe.size; // 1
aibe.clear(); // Set [ ]
```

UNIKALUS MASYVAS

- ES3 - per sunku, tai reik naudot biblioteką, pvz. Lodash

```
_.uniq([2, 1, 2]);
```

- ES5 - filtruoti

```
[2, 1, 2].filter(function(elem,pos,arr){
  return arr.indexOf(elem) === pos;
}
```

- ES6 - Set

```
Array.from(new Set([2, 1, 2]));
[... new Set([2, 1, 2])];
// spread operator ... turns iterables to function arguments
```

UNIKALŪS OBJEKTAI

Nėra hash funkcijos, todėl

```
const aibe = new Set();
const obj = {a:1,b:2};
aibe.add(obj);
aibe.add({a:1,b:2}); // ok - kitas objektas
console.log(aibe); // 2 objektai
```

ŽEMĖLAPIS (MAP)

- Kolekcijos raktas + reikšmė (key+value)
- Objektai neverčiami į string
- Naudojam, kai iš anksto nežinom raktų

```
const zem = new Map();
zem.set(1, "vardas");
zem.get(1);
zem.size; // 1
zem.has(1); // true
zem.delete(1);
zem.clear();
```

MAP/SET ELEMENTAI

- Per elementus galima pereiti naudojant iteratorių arba su:
  - for..of
  - forEach

```
for (let item of aibe) {
    console.log(item);
}

aibe.forEach(item => console.log(item));
```

SILPNAS ŽEMĖLAPIS (WEAKMAP)

- Raktai - TIK objektai
- Užima mažiau vietos atmintyje
- Iteruoti per elementus negalima
- Elementai išvalomi (surenkami GarbageCollecotr'iaus) kai jų niekas nenaudoja
- Išvalyti viso (clear) negalima

UŽDUOTIS 3

- Sukurti naują klasę Produktas su produkto atributais
- Prekių masyvo elementus map'inti į naują masyvą, kur elementai būtų new Produktas(...)
- Gautą produktų sąrašą paversti į objektų aibę
- Gautą produktų sąrašą paversti į objektų žemėlapį
- pasinaudojant Array.from iš objektų map'o ir vėl gauti masyvą, kurį suredukuoti iki vienos eilutės, kurioje būtų prekės title ir kaina
  - Obuolys (1 eur), Samsung (100 eur)

KITOJE PASKAITOJE

React komponentų būsenos, gyvavimo ciklas. Navigacija. Darbas su serveriu