Tehtäväsetti 2

Joonas Kangaskoski, n3303

**Tehtävä 1.**

*// luodaan json filun sisällöstä interface*

interface Varoitus {

  id: number;

  city: string;

  created\_at: Date;

  updated\_at: Date;

}

*// tyypitetään muuttujat*

const haeVaroitukset = async (*vuosi*: number, *kaupunki*: string) => {

  const res = await fetch(

    `https://liukastumisvaroitus-api.beze.io/api/v1/warnings?filter=city:${*kaupunki*}`

  );

  if (!res.ok) {

    throw new Error();

  }

*// tyypitetään data*

  const data: Varoitus[] = await res.json();

  return data.filter((*w*) => new Date(*w*.created\_at).getFullYear() === *vuosi*);

};

haeVaroitukset(2020, 'Jyväskylä').then((*a*) => {

*a*.forEach((*b*) => {

    console.log(*b*.created\_at)

  })

*// otetaan kiinni errori*

}).catch((*err*: Error) => console.log(*err*.message))

**Tehtävä 2.**

interface Varoitus {

  id: number;

  city: string;

  created\_at: Date;

  updated\_at: Date;

}

const haeData = async <T>(*url*: string): Promise<T> => {

  const res = await fetch(*url*);

  if (!res.ok) {

    throw new Error();

  }

  return await res.json();

};

haeData<Varoitus[]>('https://liukastumisvaroitus-api.beze.io/api/v1/warnings/')

  .then((*a*) => {

*a*.forEach((*b*) => {

      console.log(*b*.created\_at + ': ' + *b*.city)

    })

*// otetaan kiinni errori*

  }).catch((*err*: Error) => console.log(*err*.message))

haeData<Varoitus>('https://liukastumisvaroitus-api.beze.io/api/v1/warnings/1')

  .then((*a*) => {

    console.log('Varoitus kaupunkiin ' + *a*.city + ' id:' + *a*.id)

  })

*// otetaan kiinni errori*

  .catch((*err*: Error) => console.log(*err*.message))

**Tehtävä 3.**

interface StarWars {

  name: string;

  height: string;

  mass: string;

  hair\_color: string;

  skin\_color: string;

  eye\_color: string;

  birth\_year: string;

  gender: string;

  homeworld: string;

  films?: (string)[] | null;

  species?: (null)[] | null;

  vehicles?: (string)[] | null;

  starships?: (string)[] | null;

  created: string;

  edited: string;

  url: string;

}

const getDataF = <T>(*url*: string): Promise<T> => {

  return fetch(*url*).then((*res*) => {

    if (!*res*.ok) {

      throw new Error();

    }

    return *res*.json();

  });

};

getDataF<StarWars>('https://swapi.dev/api/people/1')

  .then((*a*) => console.table(*a*))

  .catch((*err*: Error) => console.log(*err*.message))

**Tehtävä 4.**

class Laatikko<T> {

  private sisalto: T | null

  constructor(*arvo*?: T) {

    if (*arvo*) {

      this.sisalto = *arvo*

    } else {

      this.sisalto = null

    }

  }

*// tehdään metodit, laita korvaa olemassa olevan arvon x:llä, toinen tulostaa mitä laatikossa on ja asettaa null,*

*// viimeinen näyttää kurkkaa laatikkoon.*

  laita(*x*: T) {

    this.sisalto = *x*

  }

  ota() {

    console.log(this.sisalto)

    this.sisalto = null

  }

  katoSisalto() {

    console.log(this.sisalto)

  }

}

*// tehdään laatikko*

const boksi = new Laatikko()

*// laitetaan laatikkoon tavaraa, se voi olla tyypiltään mikä vaan.*

boksi.laita('karhu')

boksi.laita(10)

boksi.ota()

boksi.katoSisalto()

**Tehtävä 5.**

*// tyypitetään niin että argumentit voivat olla vain objekteja*

function yhdistaObjektit<T extends object, U extends object>(*a*: T, *b*: U) {

  return Object.assign(*a*, *b*)

}

const eka = {

  etunimi: 'Raipe',

  sukunimi: 'Helminen',

  postinumero: 40520

}

const toka = {

  oppilaitos: 'Jamk',

  kaupunki: 'Jyväskylä',

  opnumero: 832

}

const yhdistelma = yhdistaObjektit(eka, toka)

console.log(yhdistelma)

*// paluutyyppi on T & U eli molemmista objekteista sisällöt*

**Tehtävä 6.**

Tehtiin tunnilla. Debuggauksen pystyi tekemään joko visual codessa tai selaimessa chromen devtoolsseilla. Asetettiin koodiin haluttuun kohtaan breakpoint ja ajettiin koodia pykälä kerrallaan, seuraten miten halutut muuttujat muuttuivat.

**Tehtävä 7.**

*// kääritään ensimmäinen funktio nimiavaruuteen ja exportataan funktio*

namespace Kerroin {

  export function kertotaulu(*x*: number, *y*: number) {

    const kertsi: number[] = [];

    for (let i = 1; i <= *y*; i++) {

      kertsi.push(i \* *x*);

    }

    return kertsi;

  }

}

const kertotaulu = (*a*: number[]) => {

*a*.forEach((*luku*) => console.log(*luku*));

};

*// ajetaan toisella kertotaululla, käärityn kertotaulun tulos*

kertotaulu(Kerroin.kertotaulu(3, 4))

**Tehtävä 8. EXTRA**

*// kertsi.ts pitää sisällään Kerroin namespacen, tästä triple-slash viittaus*

namespace Kerroin {

  export function kertotaulu(*x*: number, *y*: number) {

    const kertsi: number[] = [];

    for (let i = 1; i <= *y*; i++) {

      kertsi.push(i \* *x*);

    }

    return kertsi;

  }

}

*// alempi 7 tehtävän funktio, viittaus kertsi namespace tiedostoon triple-slash*

*/// <reference path="./kertsi.ts" />*

const kertotaulu = (*a*: number[]) => {

*a*.forEach((*luku*) => console.log(*luku*));

};

*// ajetaan toisella kertotaululla, käärityn kertotaulun tulos*

kertotaulu(Kerroin.kertotaulu(3, 4))

tsconfig.json -tiedostoon outFile -kohtaan arvo "./build/index.js" ja kohtaan module arvo "amd".

**Tehtävä 9.**

*// app.js*

const readlineSync = require('readline-sync');

const ympyra = require('./ympyra');

let halkaisija;

do {

  do {

    const vastaus = readlineSync.question('Anna halkaisija metreissa? ');

    halkaisija = parseInt(vastaus);

  } while (

    typeof halkaisija !== 'number' ||

    isNaN(halkaisija) ||

    halkaisija < 1

  );

  console.log('Ympyrän kehä on ' + ympyra.keha(halkaisija) + 'm');

  console.log('Ympyrän pinta-ala on ' + ympyra.pinta(halkaisija) + 'm²');

} while (readlineSync.keyInYNStrict('Uudestaan '));

*// ympyra.js*

const PII = 3.14;

const ympyranKeha = (*halkaisija*) => {

  if (typeof *halkaisija* !== 'number' || isNaN(*halkaisija*)) {

    throw Error('Halkaisijan tulee olla numero!');

  }

  return PII \* *halkaisija*;

};

const pintaAla = (*halkaisija*) => {

  let sade = *halkaisija* / 2;

  return PII \* Math.pow(sade, 2);

};

module.exports = {

  keha: ympyranKeha,

  pinta: pintaAla,

};

**Tehtävä 10.**

*// vakiot.js*

*// luodaan vakioista oma moduuli*

const PII = 3.14;

const PUTOAMISKIIHTYVYYS = 9.81;

*// exportataan moduulista tietoa*

module.exports = {

  pii: PII,

  putoaminen: PUTOAMISKIIHTYVYYS,

};

*// ympyra.js*

*// otetaan vakiot moduulista käyttöön tietoa*

const vakiot = require('./vakiot.js');

const ympyranKeha = (*halkaisija*) => {

  if (typeof *halkaisija* !== 'number' || isNaN(*halkaisija*)) {

    throw Error('Halkaisijan tulee olla numero!');

  }

  return vakiot.pii \* *halkaisija*;

};

const pintaAla = (*halkaisija*) => {

  let sade = *halkaisija* / 2;

  return vakiot.pii \* Math.pow(sade, 2);

};

module.exports = {

  keha: ympyranKeha,

  pinta: pintaAla,

};

**Tehtävä 11.**

*// index.ts*

import { ympyranKeha, pintaAla } from "./ympyra.js";

*// testataan defaultilla.*

let halkaisija: number = 30;

console.log('Ympyrän kehä on ' + ympyranKeha(halkaisija) + 'm');

console.log('Ympyrän pinta-ala on ' + pintaAla(halkaisija) + 'm²');

*// ympyra.ts*

*// otetaan vakiot moduulista käyttöön tietoa*

import { PII, PutoamisKiihtyvyys } from './vakiot.js'

export const ympyranKeha = (*halkaisija*: number) => {

  if (typeof *halkaisija* !== 'number' || isNaN(*halkaisija*)) {

    throw Error('Halkaisijan tulee olla numero!');

  }

  return PII \* *halkaisija*;

};

export const pintaAla = (*halkaisija*: number) => {

  let sade = *halkaisija* / 2;

  return PII \* Math.pow(sade, 2);

};

export const putoamisVauhti = (*lahtoNopeus*: number) => {

  return *lahtoNopeus* \* PutoamisKiihtyvyys

}

*// vakiot.ts*

export const PII: number = 3.14;

export const PutoamisKiihtyvyys: number = 9.81;

**Tehtävä 12.**

*// index.ts*

*// importataan koko ympyra*

import Ympyra from "./ympyra.js";

*// testataan defaultilla.*

let halkaisija: number = 30;

console.log('Ympyrän kehä on ' + Ympyra.ympyranKeha(halkaisija) + 'm');

console.log('Ympyrän pinta-ala on ' + Ympyra.pintaAla(halkaisija) + 'm²');

*// ympyra.ts*

*// otetaan vakiot moduulista käyttöön tietoa*

import { PII, PutoamisKiihtyvyys } from './vakiot.js'

*// Tehdään luokka ympyrä ja laitetaan methodit staattisiksi*

export default class Ympyra {

  constructor() {

  }

  static ympyranKeha(*halkaisija*: number) {

    if (typeof *halkaisija* !== 'number' || isNaN(*halkaisija*)) {

      throw Error('Halkaisijan tulee olla numero!');

    }

    return PII \* *halkaisija*;

  };

  static pintaAla(*halkaisija*: number) {

    let sade = *halkaisija* / 2;

    return PII \* Math.pow(sade, 2);

  };

  static putoamisVauhti(*lahtoNopeus*: number) {

    return *lahtoNopeus* \* PutoamisKiihtyvyys

  }

}

**Tehtävä 13. EXTRA**

*// importataan kaikki lodashista*

import \_ from 'lodash'

*// kirjoitellaan pikku funktio*

function ekaKirjainIsolla(*teksti*: string) {

*// käännetään ensin lodashin avulla parametri pieniin kirjaimiin*

*teksti* = \_.toLower(*teksti*)

*// tulostellaan teksi jolloin joka sana alkaa isolla kirjaimella*

  console.log(\_.startCase(*teksti*))

}

const teksti =

  'Just sit right back hear the tale, \

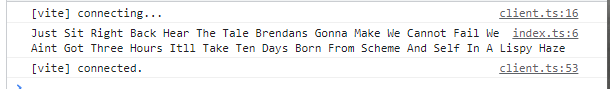
Brendan’s gonna make, we cannot fail. \

We ain’t got three hours, it’ll take ten days. \

Born from scheme and self in a lispy haze.';

*// tulostus*

ekaKirjainIsolla(teksti)



**Tehtävä 14.**

function laskeTippi(*pros*: number) {

  return function (*y*: number) {

    return *pros* / 100 \* *y*

  }

}

*// luodaan muuttujat jotka onkin funktioita.*

const tip20 = laskeTippi(20)

const tip30 = laskeTippi(30)

*// tulostellaan*

console.log(`Lasku 100, tippi ${tip20(100)}`)

console.log(`Lasku 150, tippi ${tip20(150)}`)

console.log(`Lasku 100, tippi ${tip30(100)}`)

console.log(`Lasku 150, tippi ${tip30(150)}`)

**Tehtävä 15.**

const HP = (*healthStart*: number) => {

  let health = *healthStart*

*// palauttaa objektin, jossa kolme metodia.*

  return {

*// hetkinen hp*

    get() {

      return health;

    },

*// iskuja potkuja hittiä damageee*

    hit(*x*: number) {

      health -= *x*

      return health

    },

*// hp lisää / parannus*

    revive(*y*: number) {

      return health += *y*

    }

  }

}

const player1HP = HP(100)

const player2HP = HP(100)

console.log(`Pelaaja 1 hp : ${player1HP.get()}`)

console.log(`Pelaaja 2 hp : ${player2HP.get()}`)

player2HP.hit(20)

console.log(`Pelaaja 2 ottaa hittiä parikymmentä ja hp sen jälkeen ${player2HP.get()}`)

player1HP.revive(20)

console.log(`Pelaaja 1 syö taikasienen ja saa lisähp eli  nyt hp ${player1HP.get()}`)

**EXTRA:**

*// luokkana*

class HP {

  constructor(private *health*: number) {

    this.health = *health*

  }

  get() {

    return this.health;

  }

  hit(*x*: number) {

    return this.health -= *x*

  }

  revive(*y*: number) {

    return this.health += *y*

  }

}

const player1HP = new HP(100)

const player2HP = new HP(100)

**Tehtävä 16. EXTRA**

**Tehtävä 17. EXTRA**

**Tehtävä 18.**

function fibonatsi(*n*: number): any {

  if (*n* == 0) {

    return 0

  }

  if (*n* == 1) {

    return 1

  }

  if (*n* > 1) {

    return fibonatsi(*n* - 1) + fibonatsi(*n* - 2);

  }

}

console.log(fibonatsi(10))

**Tehtävä 19. EXTRA**

**Tehtävä 20.**

const luvut = [50, -20, 100, 10, -100, 10, 30];

*// muunnetaan käsiteltävä luku positiiviseksi math.abs() avulla,*

*// palauttaa absoluuttisen arvon.*

const summa = luvut.reduce((*tulos*, *luku*) => *tulos* + Math.abs(*luku*), 0)

console.log(summa)

**Tehtävä 21.**

const luvut = [50, -20, 100, 10, -100, 10, 30];

const summa = luvut.reduce((*tulos*, *luku*) => {

  if (*luku* > 0) {

    return *tulos* + *luku*

  } else {

    return *tulos*

  }

}, 0)

console.log(summa)

**Tehtävä 22.EXTRA**

**Tehtävä 23.**

const maxNro = 40;

const arvottavienLkm = 7;

const numerot: Set<number> = new Set()

for (let i = 1; i <= maxNro; i++) {

  numerot.add(i)

}

function arvoNumero(*n*: Set<number>) {

  const kaytettavatNumerot = Array.from(*n*)

  return kaytettavatNumerot[Math.floor(Math.random() \* kaytettavatNumerot.length)]

}

let tulos: number[] = []

let lisanumerot: number[] = []

for (let i = 1; i <= arvottavienLkm; i++) {

  const arvottu = arvoNumero(numerot)

  tulos.push(arvottu)

  tulos.sort((*a*, *b*) => *a* - *b*)

  numerot.delete(arvottu)

}

*// extrana lisänumerot*

for (let i = 1; i <= 3; i++) {

  const arvottu = arvoNumero(numerot)

  lisanumerot.push(arvottu)

  lisanumerot.sort((*a*, *b*) => *a* - *b*)

  numerot.delete(arvottu)

}

console.log(`Ja illan lottonumerot on: ${tulos}`)

console.log(`sekä lisänumerot: ${lisanumerot}`)

**Tehtävä 24.EXTRA**

**Tehtävä 25.**

const konffi = new Map([

  ['hakemisto', '/home'],

  ['tiedosto', 'muistio.txt']

])

*// luodaan funktiot*

const avattavaTiedosto = () => {

*// tarkistetaan löytyykö avain-arvo parit*

  if (konffi.has('hakemisto' && 'tiedosto')) {

    return `${konffi.get('hakemisto')}/${konffi.get('tiedosto')}`

  } else {

*// asetetaan uusi arvo ja palautetaan päivitettynä*

    konffi.set('tiedosto', 'uusi.txt')

    return `${konffi.get('hakemisto')}/${konffi.get('tiedosto')}`

  }

}

const asetaTiedosto = (*nimi*: string | null) => {

*// tarkistetaan mikä arvo löytyy*

  if (*nimi* === null) {

    konffi.delete('tiedosto')

  }

  else {

*// asetetaan uusi arvo*

    konffi.set('tiedosto', *nimi*)

  }

}

console.log(avattavaTiedosto())

asetaTiedosto('vanha.txt');

console.log(avattavaTiedosto());

asetaTiedosto(null);

console.log(avattavaTiedosto());

**Tehtävä 26.EXTRA**

**Tehtävä 27.EXTRA**

**Tehtävä 28.**

* Koen osaavani opintojakson asiat hyvin, ajoittain tietysti syviä ajatuksia. Asiat käytiin hyvin ja perusteellisesti läpi, ainakin tärkeimmät. Tehtävämäärä oli hyvä ja monipuolinen, plussaa siitä, että tunnilla tehtiin yhdessä osa tehtävistä. Setämiehellä aikaa kuluu uutta oppiessa aina runsaasti, reilusti yli tuon 80h. Ihan ei kaikkia materiaaleja keritty käymään läpi tunneilla, mutta osaahan nuo lukea.