Kuis koding fundamental javascript

* **Variable dan data**

/\*\*

\* TODO:

\* Buatlah variabel firstName, lastName, age, isMarried dengan ketentuan:

\* - firstName: bertipe data string, dengan nilai nama depan Anda.

\* - lastName: bertipe data string, dengan nilai nama belakang Anda.

\* - age: bertipe data number, dengan nilai umur Anda.

\* - isMarried: bertipe data boolean, dengan bebas Anda tentukan.

\*/

// TODO

let firstName = "Andira";

let lastName = "Muhammad";

let age = 20;

let isMarried = false;

console.log(firstName);

console.log(lastName);

console.log(age);

console.log(isMarried);

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = {

firstName, lastName, age, isMarried,

};

* **Logika operator dan if**

/\*\*

\* Buatlah logika if untuk mengevaluasi nilai score dengan ketentuan:

\* 1. Jika score bernilai 90 atau lebih

\* - Isi variabel result dengan nilai: 'Selamat! Anda mendapatkan nilai A.'

\* 2. Jika score bernilai 80 hingga 89

\* - Isi variabel result dengan nilai: 'Anda mendapatkan nilai B.'

\* 3. Jika score bernilai 70 hingga 79

\* - Isi variabel result dengan nilai: 'Anda mendapatkan nilai C.'

\* 4. Jika score bernilai 60 hingga 69:

\* - Isi variabel result dengan nilai: 'Anda mendapatkan nilai D.'

\* 5. Jika score bernilai di bawah 60:

\* - Isi variabel result dengan nilai: 'Anda mendapatkan nilai E.'

\*

\*

\* Note: - Mohon untuk tidak menghapus kode yang sudah ada sebelumnya.

\* - Anda tidak perlu membuat variabel result dan score secara manual.

\* Gunakan variabel yang sudah disediakan.

\*

\*/

function scoreChecker(score) {

let result;

// TODO

if(score >= 90){

result ='Selamat! Anda mendapatkan nilai A.';

}

else if(score >= 80 && score <= 89){

result = 'Anda mendapatkan nilai B.';

}

else if(score >= 70 && score <= 79){

result = 'Anda mendapatkan nilai C.';

}

else if(score >= 60 && score <= 69){

result = 'Anda mendapatkan nilai D.';

}

else if(score < 60){

result = 'Anda mendapatkan nilai E.';

}

// Jangan hapus kode ini

return result;

}

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = scoreChecker;

Kuis koding struktur data

* **Object**

/\*\*

\* TODO

\* 1. Buatlah variabel dengan nama restaurant yang bertipe object dengan ketentuan berikut:

\* - Memiliki properti bernama "name"

\* - Bertipe data string

\* - Bernilai apa pun, asalkan tidak string kosong atau null.

\* - Memiliki properti bernama "city"

\* - Bertipe data string

\* - Bernilai apa pun, asalkan tidak string kosong atau null.

\* - Memiliki properti "favorite drink"

\* - Bertipe data string

\* - Bernilai apa pun, asalkan tidak string kosong atau null.

\* - Memiliki properti "favorite food"

\* - Bertipe data string

\* - Bernilai apa pun, asalkan tidak string kosong atau null.

\* - Memiliki properti "isVegan"

\* - Bertipe data boolean

\* - Bernilai boolean apa pun.

\*

\* 2. Buatlah variabel bernama name.

\* Kemudian isi dengan nilai name dari properti object restaurant

\* 3. Buatlah variabel bernama favoriteDrink.

\* Kemudian isi dengan nilai "favorite drink" dari properti object restaurant

\*/

// TODO

const restaurant = {

name: "Faqih",

city: "Semarang",

"favorite drink": "Es teh",

"favorite food": "nasi goreng",

isVegan: false

};

const {name} = restaurant;

const {"favorite drink": favoriteDrink} = restaurant;

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = { restaurant, name, favoriteDrink };

* **Array**

/\*\*

\* TODO:

\* Buatlah sebuah variabel dengan nama evenNumber yang merupakan sebuah array dengan ketentuan:

\* - Array tersebut menampung bilangan genap dari 1 hingga 100

\*

\* Catatan:

\* - Agar lebih mudah, gunakanlah for loop dan logika if untuk mengisi bilangan genap pada array.

\*/

// TODO

let evenNumber = [];

for(let i = 1;i<=100;i++){

if(i%2 == 0){

evenNumber.push(i);

}

}

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = evenNumber;

* **Map**

/\*\*

\* TODO:

\* 1. Buatlah variabel currency yang merupakan Map dengan kriteria:

\* - key "USD", value 14000

\* - key "JPY", value 131

\* - key "SGD", value 11000

\* - key "MYR", value 3500

\* 2. Buatlah variabel priceInIDR yang bernilai dari hasil perkalian:

\* - priceInJPY dengan nilai currency JPY

\*/

const priceInJPY = 5000;

// TODO

const currency = new Map([

["USD", 14000],

["JPY", 131],

["SGD", 11000],

["MYR", 3500]

]);

let priceInIDR = priceInJPY \* currency.get("JPY");

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = { currency, priceInJPY, priceInIDR };

Kuis Function

* **FUNCTION**

/\*\*

\* TODO:

\* 1. Buatlah fungsi bernama minimal dengan ketentuan berikut:

\* - Menerima dua buah argumen number, a dan b.

\* - Mengembalikan nilai terkecil antara a atau b.

\* - Bila nilai keduanya sama, maka kembalikan dengan nilai a

\*

\* contoh:

\* minimal(1, 4) // 1

\* minimal(3, 2) // 2

\* minimal(3, 3) // 3

\*

\* 2. Buatlah fungsi bernama power dengan ketentuan berikut:

\* - Menerima dua buah argumen number, a dan b.

\* - Mengembalikan nilai dari hasil perkalian a sebanyak b (fungsi kuadrat).

\*

\* contoh:

\* power(7, 3) // 343

\* power(3, 3) // 27

\* power(4, 0.5) // 2

\*/

// TODO

function minimal(a,b){

if(a < b){

return a;

}

else if(b < a){

return b;

}

else if(a == b){

return a;

}

}

function power(a,b){

return a\*\*b;

}

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = { minimal, power };

Kuis Object Oriented Programming

* **OOP**

/\*\*

\* TODO:

\* 1. Buatlah class bernama Animal dengan ketentuan:

\* - Memiliki properti:

\* - name: string

\* - age: int

\* - isMammal: boolean

\* - Memiliki constructor untuk menginisialisasi properti:

\* - name

\* - age

\* - isMammal

\* 2. Buatlah class bernama Rabbit dengan ketentuan:

\* - Merupakan turunan dari class Animal

\* - Memiliki method:

\* - eat yang mengembalikan nilai string `${this.name} sedang makan!`

\* - Ketika diinstansiasi, properti isMammal harus bernilai true

\* 3. Buatlah class bernama Eagle dengan ketentuan:

\* - Merupakan turunan dari class Animal

\* - Memiliki method:

\* - fly yang mengembalikan nilai string `${this.name} sedang terbang!`

\* - Ketika diinstansiasi, properti isMammal harus bernilai false

\* 4. Buatlah instance dari class Rabbit bernama "myRabbit" dengan ketentuan:

\* - properti name bernilai: "Labi"

\* - properti age bernilai: 2

\* 5. Buatlah instance dari class Eagle bernama "myEagle" dengan ketentuan:

\* - properti name bernilai: "Elo"

\* - properti age bernilai: 4

\*/

// TODO

class Animal{

constructor(name,age){

this.name = name;

this.age = age;

this.isMammal = true;

}

}

class Rabbit extends Animal{

constructor(name,age){

super();

this.name = name;

this.age = age;

this.isMammal = true;

}

eat(){

return `${this.name} sedang makan!`;

}

}

class Eagle extends Animal{

constructor(name,age){

super();

this.name = name;

this.age = age;

this.isMammal = false;

}

fly(){

return`${this.name} sedang terbang!`;

}

}

const myRabbit = new Rabbit("Labi",2);

const myEagle = new Eagle("Elo",4);

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = {

Animal, Rabbit, Eagle, myRabbit, myEagle,

};

Kuis Coding Funtional Programming

* **FP**

/\*\*

\* TODO:

\* Buatlah variabel greatAuthors yang merupakan array

\* berdasarkan hasil filter() dan map() dari books:

\* - Gunakan fungsi filter untuk mengembalikan nilai item books

\* yang hanya memiliki nilai sales lebih dari 1000000.

\* - Gunakan fungsi map pada books yang sudah ter-filter,

\* untuk mengembalikan nilai string dengan format:

\* - `${author} adalah penulis buku ${title} yang sangat hebat!`

\*

\* Catatan: Jangan ubah nilai atau struktur dari books

\*/

const books = [

{ title: 'The Da Vinci Code', author: 'Dan Brown', sales: 5094805 },

{ title: 'The Ghost', author: 'Robert Harris', sales: 807311 },

{ title: 'White Teeth', author: 'Zadie Smith', sales: 815586 },

{ title: 'Fifty Shades of Grey', author: 'E. L. James', sales: 3758936 },

{ title: 'Jamie\'s Italy', author: 'Jamie Oliver', sales: 906968 },

{ title: 'I Can Make You Thin', author: 'Paul McKenna', sales: 905086 },

{ title: 'Harry Potter and the Deathly Hallows', author: 'J.K Rowling', sales: 4475152 },

];

// TODO

const greatAuthors = books.filter((item) => item.sales > 1000000).map((item) => {return `${item.author} adalah penulis buku ${item.title} yang sangat hebat!`});

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = { books, greatAuthors };

Kuis Coding Module

* **File Main.js**

/\*\*

\* TODO 1 (Tiger.js):

\* Ekspor nilai dari class Tiger

\*

\* TODO 2 (Wolf.js)

\* Ekspor nilai dari class Wolf

\*

\* TODO 3 (main.js)

\* Impor class Tiger dan Wolf

\*

\* TODO 4 (main.js)

\* Ekspor fungsi fight, myTiger, myWolf, dan result

\*

\*/

// TODO 3

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

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

const fight = (tiger, wolf) => {

if (tiger.strength > wolf.strength) {

return tiger.growl();

}

if (wolf.strength > tiger.strength) {

return wolf.howl();

}

return 'Harimau dan serigala sama-sama kuat!';

};

const myTiger = new Tiger();

const myWolf = new Wolf();

const result = fight(myTiger, myWolf);

// TODO 4

module.exports = {fight,myTiger,myWolf,result};

console.log(myTiger);

console.log(myWolf);

console.log(result);

* **Tiger.js**

class Tiger {

constructor() {

this.strength = Math.floor(Math.random() \* 100);

}

growl() {

return 'grrrrrrr';

}

}

// TODO 1

module.exports = Tiger;

* **Wolf.js**

class Wolf {

constructor() {

this.strength = Math.floor(Math.random() \* 100);

}

howl() {

return 'Auuuuuuuuu';

}

}

// TODO 2

module.exports = Wolf;

Kuis Coding Error Handling

* **Error Handling**

/\*\*

\* Saat ini, Anda sudah memiliki fungsi detectTriangle yang berguna untuk

\* mendeteksi jenis segitiga berdasarkan nilai argumen.

\* Contoh:

\* - 1, 1, 1 -> Segitiga sama sisi

\* - 4, 4, 2 -> Segitiga sama kaki

\* - 3, 4, 6 -> Segitiga sembarang

\*

\* Namun fungsi detectTriangle belum berjalan dengan baik karena

\* bila ada argumen fungsi yang bukan number, alih-alih error, ia akan mengembalikan "Segitiga sembarang".

\* Contoh:

\* - 1, false, 1 -> Segitiga sembarang

\* - 'a', 3, 5 -> Segitiga sembarang

\* - 12, 2, null -> Segitiga sembarang

\* Kondisi yang diharapkan:

\* - 1, false, 1 -> Argumen kedua harus number

\* - 'a', 3, 5 -> Argumen pertama harus number

\* - 12, 2, null -> Argumen ketiga harus number

\*

\* Tugas Anda adalah memperbaiki fungsi detectTriangle agar berjalan dengan kondisi yang diharapkan.

\* Pastikan Anda menggunakan teknik Throwing dan Handling Error yah.

\*

\* TODO 1:

\* - Buatlah class ValidationError yang merupakan custom error dengan spesifikasi berikut:

\* - Turunan dari class Error

\* - Memiliki constructor(message)

\* - this.name harus bernilai "ValidationError"

\*

\* TODO 2:

\* - Buatlah fungsi validateNumberInput yang memvalidasi 3 buah input (argumen) dengan spesifikasi berikut:

\* - Menerima 3 argumen

\* - Bila argumen pertama bukan number:

\* - throw ValidationError dengan pesan 'Argumen pertama harus number'

\* - Bila argumen kedua bukan number:

\* - throw ValidationError dengan pesan 'Argumen kedua harus number'

\* - Bila argumen ketiga bukan number:

\* - throw ValidationError dengan pesan 'Argumen ketiga harus number'

\*

\* TODO 3:

\* - Panggil fungsi validateNumberInput di dalam fungsi detectTriangle untuk memvalidasi nilai argumen a, b, dan c.

\* - pastikan Anda memanggil validateNumberInput menggunakan try .. catch.

\* - bila block catch terpanggil, kembalikan fungsi detectTriangle dengan pesan error yang dibawa fungsi validateNumberInput.

\*/

// TODO 1

class ValidationError extends Error {

constructor(message){

super(message);

this.name = "ValidationError";

}

}

// TODO 2

function validateNumberInput(sisi1,sisi2,sisi3){

if(typeof sisi1 !== "number"){

throw new ValidationError('Argumen pertama harus number');

}

else if(typeof sisi2 !== "number"){

throw new ValidationError('Argumen kedua harus number');

}

if(typeof sisi3 !== "number"){

throw new ValidationError('Argumen ketiga harus number');

}

}

const detectTriangle = (a, b, c) => {

// TODO 3

try{

validateNumberInput(a,b,c);

if (a === b && b === c) {

return 'Segitiga sama sisi';

}

if (a === b || a === c || b === c) {

return 'Segitiga sama kaki';

}

return 'Segitiga sembarang';

}

catch(error){

return error.message;

}

};

console.log(detectTriangle(5,5,5));

console.log(detectTriangle(7,7,5));

console.log(detectTriangle(7,6,5));

console.log(detectTriangle(5,false,5));

/\*\*

\* Jangan hapus kode di bawah ini

\*/

module.exports = { ValidationError, validateNumberInput, detectTriangle };

Kuis Coding asynchronous

* Kuis 1

/\*\*

\* @TODO

\* Ubahlah fungsi asynchronous callback-based getProvinces menjadi Promise-based.

\*

\* Catatan:

\* - Anda boleh menggunakan util.promisify untuk mengubah fungsi callback-based menjadi Promise-based.

\* - Jika nama fungsinya berubah, sesuaikan nilai yang diekspor tanpa mengubah nama properti di akhir berkas ini.

\*/

function getProvinces(countryId, callback) {

setTimeout(() => {

if (countryId === 'id') {

callback(null, [

{ id: 'id-jk', name: 'Jakarta' },

{ id: 'id-bt', name: 'Banten' },

{ id: 'id-jr', name: 'Jawa Barat' },

]);

return;

}

callback(new Error('Country not found'), null);

}, 1000);

}

module.exports = { getProvinces: getProvinces };

* Kuis 2

/\*\*

\* Ini adalah program untuk mendapatkan nama user dari internet.

\* Terdapat dua fungsi yang sudah dibuat, berikut penjelasanya:

\* - fetchingUserFromInternet:

\* - fungsi ini digunakan untuk mendapatkan data user seolah-olah dari internet.

\* - fungsi ini menerima dua argumen yakni callback, dan isOffline.

\* - Argumen callback membawa dua nilai yakni error dan user:

\* - error: NetworkError akan dibawa oleh callback bila isOffline bernilai true.

\* - user: data user akan dibawa oleh callback bila isOffline bernilai false.

\* - gettingUserName:

\* - fungsi ini memanggil fungsi fetchingUserFromInternet dengan nilai isOffline: false untuk mendapatkan data user name dari internet.

\* - fungsi ini harus mengembalikan nilai user.name, namun sulit karena menggunakan pola callback.

\* - Maka dari itu, ubahlah fetchingUserFromInternet dari callback menjadi promise

\* - Dengan begitu, Anda bisa memanfaatkan .then atau async/await untuk mendapatkan user.name.

\*

\* TODO: 1

\* - Ubahlah fungsi fetchingUserFromInternet dengan memanfaatkan Promise. Anda bisa menghapus implementasi callback.

\*

\* TODO: 2

\* - Ubahlah cara mengonsumsi fungsi fetchingUserFromInternet dari callback ke Promise.

\* - Tips:

\* - Agar penulisan kode lebih bersih dan mudah dibaca, coba manfaatkan async/await

\*

\*

\* Notes:

\* - Jangan ubah struktur atau nilai dari objek user yang dibawa oleh callback sebelumnya.

\* - Tetap gunakan NetworkError untuk membawa nilai error pada Promise

\*/

class NetworkError extends Error {

constructor(message) {

super(message);

this.name = 'NetworkError';

}

}

// TODO: 1

const fetchingUserFromInternet = (isOffline) => {

return new Promise((resolve,reject) =>{

setTimeout(() => {

if (isOffline) {

reject(new NetworkError('Gagal mendapatkan data dari internet'));

}

resolve({ name: 'John', age: 18 });

}, 500);

});

};

// TODO: 2

async function gettingUserName(){

try{

const user = await fetchingUserFromInternet(false);

console.log(user.name);

return user.name;

}

catch(error){

console.log(error.message);

return error.message;

}

};

gettingUserName();

/\*\*

\* Abaikan kode di bawah ini

\*/

module.exports = { fetchingUserFromInternet, gettingUserName, NetworkError };