Pertemuan 08
Code Splitting di Javascript dan PHP

Pendahuluan
Di pemrograman kita terkadang memiliki project yang sangat besar dan memiliki banyak baris, hal ini bisa mengurangi produktifitas kita karena kita akan sering men-scroll atas kebawah, di dunia nyata kita sering menggunakan-nya untuk memisah kode-kode yang tidak memiliki hubungan dengan nama file, misalnya file yang memiliki tentang Filesystem kita memberikan nama Filesystem, yang tidak termasuk dengan Filesystem disingkirkan.

Di dalam Javascript Web kita bisa menggunakan code splitting dengan memberi script tag attribute <script src=”app.js” type=”module”></script>

Code Splitting Basic di Javascript

Untuk memisahkan code di javascript kita harus membuat function atau class atau variabel didalam file tersendiri, lalu kita taruh keyword *export* di paling bawah untuk memberitahu kalau ini function/class/variabel ini dapat diakses di file lain.

Contoh
Pada file external misal imported-file.js

```
let user = {
    id: 14,
    name: "UnknownRori",
    email: "UnknownRori@mail.com"
}

function printUser() {
    console.log(user);
}

const reset = () => {
    user = {};
}

export {user, printUser, reset};

```

kita dapat export macam-macam variabel, function, ataupun class, tetapi di javascript ada fitur yaitu *export default* cara kerja-nya hampir sama dengan export tetapi export default hanya dapat dipanggil sekali dan hanya dapat men-export 1 function/class/variabel, ataupun juga membuat banyak jenis export dengan menggunakan { }, fitur ini sering digunakan dikarenakan sangat memudahkan untuk melakukan pemisahan kode.

Contoh
di file export-default-file.js

```
let user = {
    id: 14,
    name: "UnknownRori",
    email: "UnknownRori@mail.com"
}

export default user;

```

Cara seperti ini sangat sering digunakan daripada cara yang :

```
let user = {
    id: 14,
    name: "UnknownRori",
    email: "UnknownRori@mail.com"
}

function printUser() {
    console.log(user);
}

const reset = () => {
    user = {};
}

export default {user, printUser, reset};

```

Karena dapat menghasilkan, variabel import (ketika mengambil variabel/function/class dari yang di-export oleh file lain) yang berbentuk Javascript Object.

```
{
  user: { id: 14, name: 'UnknownRori', email: 'UnknownRori@mail.com' },
  printUser: [Function: printUser],
  reset: [Function: reset]
}
```

Untuk melakukan import kita tinggal definisikan import diatas file yang kita ingin mendapatkan data yang di-export

Contoh :

```
import {user, printUser, reset as resetUser} from './imported-file.js';
import exportDefaultedUser from './export-default-file.js';

console.log(exportDefaultedUser);
console.log(user);
resetUser(); // bukan reset() lagi
console.log(exportDefaultedUser.user);

```

Karena kita tidak memakai module bundler (kita pasti akan pakai besok kedepan-nya) kita harus menspesifik file yang kita import, dengan menggunakan (./)

Kita liat ini, yang memakai export biasa kita dapat mengimpor nama-nama yang sudah didefinisikan saat export dan menggunakan as untuk memberitau Javascript bahwa variabel tersebut di-aliaskan.

Tetapi kalau kita lihat file yang export default kita dapat memberikan nama yang bebas ketika import, tetapi cara mengakses variable tersebut dengan cara kita mengakses Javascript Object.

Atau kita dapat men-import semua-nya (semua yang di definisikan oleh keyword export tadi)

```
import * as importedFile from './imported-file.js';
import exportDefaultedUser from './export-default-file.js';

console.log(exportDefaultedUser);
console.log(importedFile.user);
importedFile.reset();
console.log(exportDefaultedUser.user);

```

Cara akses nya lebih mirip dengan export default.

Kita dapat melakukan apapun seperti deklarasi Object Class di file lain, lalu kita import variabel tersebut di file yang kita butuhkan, hal ini dapat mengurangi duplikasi kode.

Code Splitting di PHP
Di PHP cara Code Splitting sedikit berbeda dengan Javascript, karena di javascript kita harus spesifik apa yang mau di export, kalau di PHP kita menyambungkan antara dua file, jadi bayangin aja 2 kertas disambungkan menjadi 1, kode-nya tetap jalan dari atas kebawah tetapi PHP akan membaca apapun sebelum import-nya lalu ketika masuk dibaris import PHP pindah ke file yang tadi di definisikan untuk dibaca dari atas kebawah.

Di PHP tidak ada keyword import, tetapi kita menggunakan include dan require, kedua-nya tersebut kegunaan dan cara kerja-nya sama tetapi ketika file-nya tidak ada kedua keyword tersebut menghasilkan error yang berbeda, include akan memberi warning, kalau require akan mengeluarkan exception(error) yang menghentikan server.

Contoh :

di file Stack.php kita definisikan class Stack.

```php
<?php

class Stack
{
    public $data = [];

    public function top()
    {
        $panjangArrayData = count($this->data) - 1;
        return $this->data[$panjangArrayData];
    }

    public function push($dataBaru)
    {
        $this->data[] = $dataBaru; // sama dengan array_push($this->data, $dataBaru);
    }

    public function pop()
    {
        $panjangArrayData = count($this->data) - 1;
        $hasil = $this->data[$panjangArrayData];

        unset($this->data[$panjangArrayData]);

        return $hasil;
    }
}

```

dan di file yang kita ingin mendapatkan access dari class Stack

```php
<?php

include "Stack.php";
include_once "Stack.php";
require "Stack.php";
require_once "Stack.php";

$myStack = new Stack();

$myStack->push(1);
$myStack->push(2);
$myStack->push(3);
$myStack->push(4);

echo $myStack->pop();

var_dump($myStack->data);

```

Nah ini pasti penasaran kok ada _once di keyword include dan require, itu sangat di perlukan ketika kita hanya meng-inginkan file tersebut hanya dibaca sekali, karena deklarasi class dan function hanya dapat sekali, setelah di deklarasi kita tidak dapat men-deklarasi ulang.

Di contoh di-atas kita import berkali-kali tetapi, kita sebenarnya hanya perlu import sekali saja dengan 4 keyword tersebut, karena ini pasti akan muncul error tentang deklarasi class Stack sudah ada.

Sistem Namespace di PHP
Nah karena di PHP import-nya itu cuman membaca dari atas kebawah, kita terkadang akan kehabisan nama karena di PHP kita tidak dapat menggunakan alias seperti di Javascript tetapi kita akan bisa menggunakan alias ketika menggunakan Namespace.

Untuk deklarasi Namespace kita hanya menggunakan keyword
namespace namespaceku\nested_namespace_ku

Contoh :

```php
<?php

namespace MyNamespace;

class Stack
```

untuk import-nya kita tinggal menggunakan use.
use namespace\nama-yang-di-import

```php

include "Stack.php";

use MyNamespace\Stack;

$myStack = new Stack();
```

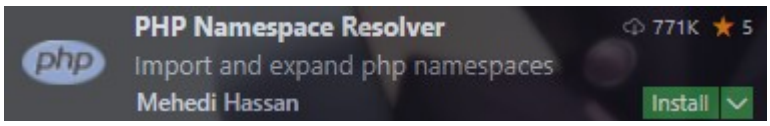
Untuk menggunakan alias kita tinggal

```php
include "Stack.php";

use MyNamespace\Stack as myStack;

$myStack = new myStack();
```

Note : untuk pengguna Visual Studio Code bisa men-download extensi agar lebih mudah menggunakan namespace, khusus-nya ketika mulai nested



Untuk tugas kali ini tidak ada, jadi experimen konsep code splitting ini agar besok ketika kita membuat project besar atau menggunakan framework kita pasti disuruh code splitting.