ONGIS SCHOOL : FRONTEND DEV

SESI

1. KONSEP WEB COMPONENTS & TASK RUNNER
2. CUSTOM ELEMENTS
3. SHADOW DOM
4. HTML IMPORT
5. HTML TEMPLATE
6. FINISHING PROJECT

SESI 1 : KONSEP WEB COMPONENTS & TASK RUNNER

WEB COMPONENTS

Web component adalah konsep kerja / kumpulan API Platform web yang memungkinkan Frontend Dev membuat tag HTML khusus dengan penamaan sesuai keinginan pengguna, yang dapat digunakan kembali pada setiap bagian website, atau dikelompokkan menjadi satu pustaka untuk digunakan pada website atau aplikasi website. Komponen dan widget pada konsep ini khusus dibangun diatas standar pengembangan web, Frontend Dev tidak perlu khawatir akan dukungan browser karena hampir bekerja cukup baik di seluruh browser modern, juga dapat digabungkan penggunaan nya dengan pustaka atau kerangka Javascript yang sesuai dengan HTML

Dalam pengembangan project, web components ada baiknya didasarkan pada standar web yang ada, semua fitur untuk mendukung konsep ini ditambahkan ke spesifikasi HTML dan DOM. Frontend Dev leluasa mengembangkan HTML dengan element baru dengan fitur yang disesuaikan dengan website atau aplikasi web yang sedang dikembangkan.

Spesifikasi utama Web Components ada 4, yaitu :

1. Custom Elements
2. Shadow DOM
3. HTML Import
4. HTML Template

Tag HTML Standar

<!DOCTYPE html>

<html lang=”en”>

<head>

<title> … </title>

<meta charset=”UTF-8”>

<meta name=”” content=”width=device-width, initial-scale=1”>

<link href=”.css” rel=”stylesheet”>

<script src=”.js” type=”text/javascript”></script>

</head>

<body>

<header>

…

</header>

<section>

…

</section>

<footer>

…

</footer>

</body>

</html>

Struktur Folder

css

--------- .css

js

--------- .js

image

fonts

index.html

Tag HTML dengan konsep web components

<!DOCTYPE html>

<html lang=”en”>

<head>

<title> … </title>

<meta charset=”UTF-8”>

<meta name=”” content=”width=device-width, initial-scale=1”>

<link href=”.css” rel=”stylesheet”>

<script src=”.js” type=”text/javascript”></script>

<link rel=”import” href=”./app/header.html”>

<link rel=”import” href=”./app/section.html”>

<link rel=”import” href=”./app/footer.html”>

atau

<script src=”./app/header.js” type=”text/javascript”></script>

<script src=”./app/section.js” type=”text/javascript”></script>

<script src=”./app/footer.js” type=”text/javascript”></script>

</head>

<body>

<header></header>

<section></section>

<footer></footer>

</body>

</html>

Struktur Folder

app

--------- .html / .js

css

--------- .css

js

--------- .js

image

fonts

index.html

Keuntungan menggunakan konsep web components

1. Pengembangan nya seirama dengan standar HTML
2. Konsep modul, artinya memecah keseluruhan bagian web menjadi bagian bagian kecil sehingga jauh lebih memudahkan Frontend Dev dalam penanganan project.
3. Mengurangi masalah render blocking.
4. Memungkinkan website / aplikasi web bekerja lebih ringan karena setiap bagian website / aplikasi web hanya memanggil yang diperlukan.
5. …

Dukungan browser

Chrome

Opera

Safari

Firefox

Edge / IE9+

Android Browser

Apple Mobile Safari (+iOS8)

INTEGRASI PROJECT

STRUKTUR DEVELOPMENT FOLDER

project

--------- dist

--------- src

------------------- css

------------------- sass

----------------------------- style.sass

----------------------------- components

--------------------------------------- \_file.sass

------------------- js

----------------------------- file.js

------------------- img

------------------- app

----------------------------- file.js / file.html

------------------- fonts

------------------- index.html

**TASK-RUNNER**

APA ITU TASK RUNNER ?

Tool pembantu mengerjakan banyak hal secara otomatis

TUJUAN PENGGUNAAN TASK RUNNER

1. node.js & npm

library packager

instalasi via terminal :

linux ubuntu :

* + 1. curl –sL [https://deb.nodesource.com/setup\_\*.x](https://deb.nodesource.com/setup_*.x) | sudo –E bash –
    2. sudo apt-get install –y nodejs

windows:

…

mac :

* + 1. brew install node

Node JS source :

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

<https://nodejs.org/en/download/package-manager/>

1. setelah instalasi buka terminal menuju ke folder project
2. ketik perintah

npm init

1. isi sesuai kebutuhan pengguna
2. akhiri dengan jawaban “yes”

1. sass

css pre processor

instalasi :

linux :

* + 1. sudo apt-get install ruby
    2. sudo su –c “gem install sass”

windows :

* + 1. install ruby package hasil download dari <http://rubyinstaller.org/>
    2. buka terminal ketik

gem install sass

mac :

1. gem install sass

doc :

<http://sass-lang.com/install>

1. gulp

instalasi

npm install gulp –save-dev

doc :

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

tugas :

(?) (1)…………………………………………………………

dev :

* gulp-header

instalasi :

npm install gulp-header –save-dev

doc:

<https://www.npmjs.com/package/gulp-header>

tugas :

(?) (2)…………………………………………………………

* gulp-sourcemaps

instalasi :

npm install gulp-sourcemaps –save-dev

doc:

<https://www.npmjs.com/package/gulp-sourcemaps>

tugas :

(?) (3)…………………………………………………………

* gulp-sass

instalasi :

npm install gulp-sass –save-dev

doc:

<https://www.npmjs.com/package/gulp-sass>

tugas :

(?) (4)…………………………………………………………

* gulp-cssnano

instalasi :

npm install gulp-cssnano –save-dev

doc:

<https://www.npmjs.com/package/gulp-cssnano>

tugas :

(?) (5)…………………………………………………………

* gulp-autoprefixer

instalasi :

npm install gulp-autoprefixer –save-dev

doc:

<https://www.npmjs.com/package/gulp-autoprefixer>

tugas :

(?) (6)…………………………………………………………

* gulp-concat

instalasi :

npm install gulp-concat –save-dev

doc:

<https://www.npmjs.com/package/gulp-concat>

tugas :

(?) (7)…………………………………………………………

* gulp-jshint

instalasi :

npm install gulp-jshint –save-dev

doc:

<https://www.npmjs.com/package/gulp-jshint>

tugas :

(?) (8)…………………………………………………………

* gulp-uglify

instalasi :

npm install gulp-uglify –save-dev

doc:

<https://www.npmjs.com/package/gulp-uglify>

tugas :

(?) (9)…………………………………………………………

* gulp-watch

instalasi :

npm install gulp-watch –save-dev

doc:

<https://www.npmjs.com/package/gulp-watch>

tugas :

(?) (10)…………………………………………………………

* gulp-copy

instalasi :

npm install gulp-copy –save-dev

doc:

<https://www.npmjs.com/package/gulp-copy>

tugas :

(?) (11)…………………………………………………………

* gulp-zip

instalasi :

npm install gulp-zip –save-dev

doc:

<https://www.npmjs.com/package/gulp-zip>

tugas :

(?) (12)…………………………………………………………

* browser-sync

instalasi :

npm install browser-sync –save-dev

doc:

<https://www.npmjs.com/package/browser-sync>

tugas :

(?) (13)…………………………………………………………

* jshint

instalasi :

npm install jshint –save-dev

doc :

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

tugas :

(?) (14)…………………………………………………………

KONFIGURASI TASK RUNNER

1. edit package.json

{

"name": "materi\_01",

"version": "1.0.0",

"description": "materi ongis school kelas frontend dev",

"main": "gulpfile.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [

"task",

"runner",

"custom",

"elements",

"shadow",

"DOM",

"html",

"import",

"html",

"template"

],

"author": "Mifan Twan Ardana",

"license": "ISC",

"devDependencies": {

"browser-sync": "^2.18.12",

"gulp": "^3.9.1",

"gulp-autoprefixer": "^4.0.0",

"gulp-concat": "^2.6.1",

"gulp-copy": "^1.0.0",

"gulp-cssnano": "^2.1.2",

"gulp-header": "^1.8.8",

"gulp-jshint": "^2.0.4",

"gulp-sass": "^3.1.0",

"gulp-sourcemaps": "^2.6.0",

"gulp-uglify": "^3.0.0",

"gulp-watch": "^4.3.11",

"gulp-zip": "^4.0.0",

"jshint": "^2.9.4"

}

}

menjadi

{

"name": "materi\_01",

"version": "1.0.0",

"description": "materi ongis school kelas frontend dev",

"main": "gulpfile.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [

"task",

"runner",

"custom",

"elements",

"shadow",

"DOM",

"html",

"import",

"html",

"template"

],

"author": {

"name": "Mifan Twan Ardana",

"homepage": "http://vidiproject.com/"

},

"license": "ISC",

"devDependencies": {

"browser-sync": "^2.18.12",

"gulp": "^3.9.1",

"gulp-autoprefixer": "^4.0.0",

"gulp-concat": "^2.6.1",

"gulp-copy": "^1.0.0",

"gulp-cssnano": "^2.1.2",

"gulp-header": "^1.8.8",

"gulp-jshint": "^2.0.4",

"gulp-sass": "^3.1.0",

"gulp-sourcemaps": "^2.6.0",

"gulp-uglify": "^3.0.0",

"gulp-watch": "^4.3.11",

"gulp-zip": "^4.0.0",

"jshint": "^2.9.4"

}

}

1. create file baru dengan nama gulpfile.js
2. isi gulpfile.js

'use strict';

//global variable

var sources = require('./package.json'),

gulp = require('gulp'),

browserSync = require('browser-sync').create(),

header = require('gulp-header'),

sourcemaps = require('gulp-sourcemaps'),

banner = [

'/\*\*',

' \* <%= sources.name %> - <%= sources.description %>',

' \* @author <%= sources.author.name %>',

' \* @version v<%= sources.version %>',

' \* @link <%= sources.author.homepage %>',

' \* @license <%= sources.license %>',

' \*/',

''

].join('\n');

var browser\_support = [

'last 2 versions',

'> 5%',

'Firefox ESR',

"ie >= 10",

"ie\_mob >= 10",

"ff >= 30",

"chrome >= 34",

"safari >= 7",

"opera >= 23",

"ios >= 7",

"android >= 4.4",

"bb >= 10"

],

date = new Date().toISOString().slice(0,10);

//css-task variable

var sass = require('gulp-sass'),

cssnano = require('gulp-cssnano'),

autoprefixer = require('gulp-autoprefixer'),

//js-task variable

concat = require('gulp-concat'),

jshint = require('gulp-jshint'),

uglify = require('gulp-uglify'),

//staging & delivery variable

copy = require('gulp-copy'),

zip = require('gulp-zip');

// css task

gulp.task('sass', function() {

return gulp.src("./src/sass/\*\*/\*.sass")

.pipe(sourcemaps.init())

.pipe(sass({outputStyle: 'compressed', errLogToConsole: true}).on('error', sass.logError))

.pipe(concat(sources.name + '.min.css'))

.pipe(cssnano({autoprefixer: {browsers: browser\_support, add: true} }))

.pipe(sourcemaps.write('../maps'))

.pipe(header(banner, {sources : sources}))

.pipe(gulp.dest("./src/css/"))

.pipe(browserSync.stream())

});

// javascript task

gulp.task('javascript', function() {

return gulp.src('./src/js/\*\*/\*.js')

.pipe(jshint())

.pipe(jshint.reporter('default'))

.pipe(jshint.reporter('fail'))

.pipe(sourcemaps.init())

.pipe(concat(sources.name + '.min.js'))

.pipe(uglify())

.pipe(sourcemaps.write('../maps'))

.pipe(header(banner, {sources : sources}))

.pipe(gulp.dest('./src/app/'))

.pipe(browserSync.stream())

});

// html task

gulp.task('html', function() {

return gulp.src('./src/\*.html')

.pipe(gulp.dest('./src'))

.pipe(browserSync.stream())

});

// static server & task watch

gulp.task('default', function() {

browserSync.init({

server: "./src"

});

gulp.watch('src/sass/\*\*/\*.sass', function (event) {

console.log(event);

gulp.start('sass');

});

gulp.watch('src/js/\*\*/\*.js', function (event) {

console.log(event);

gulp.start('javascript');

});

gulp.watch('src/\*\*/\*.html', function (event) {

console.log(event);

gulp.start('html').on('change', browserSync.reload);

});

});

// delivery & compress ( integrated with web & apps )

gulp.task('dist', function(){

return gulp.src('./src/\*\*/\*')

.pipe(zip( 'prod\_'+ sources.name + '\_' + date +'.zip'))

.pipe(gulp.dest('./dist'))

});

1. buat file global.js di folder src/js
2. isi dengan :

console.log(‘test global js’);

1. buat file style.sass di folder src/sass
2. isi dengan :

@import components/global

1. buat folder baru dengan nama components di folder src/sass
2. buat file \_global.sass di folder src/sass/components tadi
3. isi dengan :

\*

padding: 0

margin: 0

1. ketik perintah :

gulp

1. perhatikan aktifitas terminal setiap kali terjadi event “add” , “change” atau “unlink” pada saat pekerjaan dev berlangsung.
2. keluar gulp dengan keyboard key :

CTRL + C

1. ketik perintah :

gulp dist

1. (?) (15) alur task runner …………………………………………………………………………………………………….

NEXT ??? CUSTOM ELEMENTS

CREDIT

<http://webcomponents.org>

<http://nodejs.org>

<http://npmjs.com>

<http://jshint.com>

<http://sass-lang.com>

DOC :

GLOSARIUM:

DOM ( DOCUMENT OBJECT MODEL ) adalah kumpulan aturan yang diperlukan javascript untuk memanipulasi elemen apapun yang tampil dalam halaman website / aplikasi web.

SESI 2 : CUSTOM ELEMENTS

CUSTOM ELEMENTS

Mengijinkan Frontend Developer untuk menetapkan type baru untuk tag html

Contoh tag html standar :

<h1></h1>

contoh custom element :

<f></frontend>

component tidak akan pernah ada tanpa ada fitur yang kita buat sendiri sebagai syarat menggunakan custom element, yaitu :

menetapkan element HTML / DOM baru

membuat element HTML / DOM baru berdasarkan tag yang sudah ada

membuat satu paket logic fungsi / perintah ke dalam satu tag

Mendaftarkan elements baru

Custom elements dibuat dengan document.registerElement();

var FeHead = document.registerElement(‘fe-head’);

document.body.appenchild(new FeHead());

argument pertama dari document.registerElement() adalah tag name yang kita buat

namanya harus mengandung dash (-)

contoh :

<fe-head></fe-head>

tidak disarangkan

<fehead></fehead> atau <fe\_head></fe\_head>

argument pertama adalah standar penulisan yang memungkinkan parser untuk membedakan element khusus dari element biasa, juga memastikan kompatibilitas ke depan saat tag baru ditambahkan ke HTML.

argument kedua adalah object (opsi) untuk menggambarkan prototype element. Ini adalah tempat untuk menambahkan fungsionalitas khusus ( seperti public properties atau methods ) ke tag html buatan.

Secara default, custom element mewarisi dari dasar HTMLElement. Jadi contoh sebelumnya setara dengan :

var FeHead = document.registerElement(‘fe-head’, {

prototype: Object.create(HTMLElement.prototype)

});

Argumen ke document.registerElement ('FeHead') mengajarkan browser tentang elemen baru, dan mengembalikan konstruktor yang dapat Anda gunakan untuk membuat contoh <fe-head>. Sebagai alternatif, Anda dapat menggunakan teknik instantiating elemen lain jika Anda tidak ingin menggunakan konstruktor.

Extending elements

Custom elements memungkinkan untuk menambahkan fungsi element HTML pada tag asli yang ada dan element khusus lainnya. Untuk menambahkan fungsi, perlu menambahkan nama pada argument registerElement() dan protote element yang akan diwarisi

Contoh

Extending native elements

Kalau kita tidak puas dengan fungsi tag html

<button></button>

kita boleh menambahkan sendiri fungsi pada tag html. Untuk menambahkan fungsi pada tag <button></button>, buat element baru yang mewarisi prototype HTMLButtonElement dan menambahkan fungsinya.

var FeButton = document.registerElement(‘fe-button’, {

prototype: Object.create(HTMLButtonElement.prototype),

extends: ‘button’

});

panggil

<button is=”fe-button”></button>

custom elements yang mewarisi dari elemen standar disebut extension custom elements, bisa kita baca dengan elemen button adalah element fe-button

Lifecycle callback methods

Elements dapat menentukan methods khusus untuk memanfaatkan masa keberadaan elements itu sendiri. Ada 4 jenis argument pada methods ini : masing masing punya nama dan tujuan tertentu

createdCallback

contoh dari element yang dibuat

attachedCallback

contoh dari element yang ditambahkan ke dalam document

detachedCallback

contoh dari element yang dihilangkan dari dalam document

attributeChangedCallback(attrName, oldVal, newVal)

attribute ketika menambahkan, menghapus, atau mengupdate

contoh :

var VarElement = Object.create(HTMLElement.prototype);

VarElement.createdCallback = function(){

This.innerHTML = “<p>hallo, kenalin saya element baru</p>”;

};

var VarEl = document.registerElement(‘var-el’, {

prototype: VarElement

});