wantsome
The friendly IT Academy

Saptamana 14

Partea 1

Programare Front-End

1. Large scale apps development

Developing production ready apps common issues

- Dezvoltarea de aplicatii web care sunt gata a fi folosite de catre utilizatorii finali poate intampina diverse probleme in momentul in care acestea implica un numar mare de functionalitati iar codul care sta la baza acestora devine destul de amplu:
  - multipli programatori cu stiluri diferite de a scrie codul si de a organiza fisierele lucreaza la functionalitati diferite, motiv pentru care structura aplicatie poate deveni haotica in lipsa unui *styleguide* si a unor metodologii comune
  - o mare parte din lucrurile necesare dezvoltarii devin repetitive - programatorii au nevoie sa se focuseze pe dezvoltarea noului cod in locul acestora
  - volumul de date necesare spre a fi transferate intre server si client devine destul de mare, lucru care afecteaza incarcarea rapida a paginilor
  - livrarea solutiilor finale in mediul de productie trebuie facuta manual de fiecare data la sfarsitul unei iteratii
  - ...

Developing production ready apps common issues

Ce solutii avem ?

- din motivele anterior prezentate, a fost necesara dezvoltarea unor solutii care implica programe, unelte si metodologii pentru a usura munca programatorilor si a optimiza procesul de dezvoltare al aplicatiilor

- astfel de solutii ofera suport pentru:

  - instalarea de pachete / librarii externe ce ofera solutii optime deja dezvoltate pentru anumiti algoritmi

  - reincarcarea si recompilarea automata a noului cod scris in momentul in care fiserele modificate se salveaza de catre programator

  - optimizarea resurselor ce trebuie transferate la nivel de client de catre server prin micsorarea marimii acestora pe baza unor diferite tehnici (*minification* for css, js, *images optimization, bundling*)

Developing production ready apps common issues

- adaugarea automata a prefix-urilor pentru proprietatile CSS astfel incat acestea sa poata fi interpretate de catre toate browsere-le indiferent de tip si versiune

- verificarea automata a codului pentru greseli de sintaxa si structura

- crearea automata de fisiere necesare pentru noi functionalitati

- executarea automata a testelor unitare

- executarea automata a unor unelte de analiza a calitatii codului

1.1 Workflow automation and optimization tools

Tools - Package managers, Task runners & production build systems

- **npm**
- **yalc**
- Gulp
- Grunt
- **Webpack**
- Rollup
- Parcel
- Browserify
- …

tip: check google and official docs for most popular ones

Tools comparison & how to

Case1: **Grunt**

- *Command line tool* pentru *front-end developers*, destinat sa execute taskuri predefinite, implementate de catre programatori: minifcare fisiere JS sau CSS, refresh automat al *browser*-ului in urma schimbarilor din cod, etc...

```
concat: {
    options: {
      separator: ';'
    },
    dist: {
      src: ['src/**/*.js'],
      dest: 'dist/<%= pkg.name %>.js'
    }
}
```

- PROs:
  - Ecosistem foarte mare de *plugins* (peste 6000)
- CONs:
  - devine complicat pe masura ce nevoia de configurari multiple creste
  - lipsa de flexibilitate in adaugarea de taskuri care nu sunt comune
  - tendinta de folosire este in scadere (performanta scazuta)

Tools comparison & how to

Case2: **Gulp**

- **Gulp**-ul a aparut la diferenta de ~1.5 ani de **Grunt**
- Este tot un *task-runner* pentru definirea si folosirea de task-uri repetitive
- Are un approach diferit fata de grunt care este configurat prin intermediul obiectelor - gulp functioneaza prin configurari de functii care sunt inlantuite intr-un *pipeline*

```
gulp.task('sass', function () {
  return gulp.src('./sass/**/*.scss')
    .pipe(sass( { errLogToConsole: true } ))
    .pipe(gulp.dest('./public/css'));
});
```

- *PROs*:

  - claritate si control asupra flow-ului de taskuri

  - ecosistem mare de plugin-uri si fiecare sunt separate

  - performanta mai buna decat grunt deoarece foloseste mai putina memorie

  - mai putin cod de scris pentru taskuri.

- *CONs*:

  - mai greu de inteles cum functioneaza *stream*-urile si *promise*-urile, dar mai usor de folosit dupa ce sunt intelese

Tools comparison & how to

Case3: **Webpack**

- are un approach putin diferit fata de gulp si grunt, e definit ca fiind un *module bundler*, dar e folosit pentru aceleasi scop
- Comparand *webpack*-ul cu primele doua - acesta ofera o mai mare flexibilitate si functionalitati mai avansate pentru proiectele bazate pe React, Angular, frameworks

```
module.exports = {
  entry: './main.js',
  output: {
    filename: 'bundle.js'
  },
  module: {
    rules:[
      {
        test: /\.css$/,
        use: [ 'style-loader', 'css-loader' ]
      },
    ]
  }
};
```

- PROs:
- mai usor de facut *split* la cod si dependente
- *Webpack-dev-server*, *hot reloading*

- CONs:

  - mai dificil de inteles si configurat

	Browserify	Grunt	Gulp	Webpack
Description	Browser-side require() the node way	The Java Script Task Runner	The streaming build system	Module bundler
Created	7 years ago (Feb, 2011)	7 years ago (Jan, 2012)	5 years ago (Jul, 2013)	6 years ago (Mar, 2012)
Version Average	every 6 days	every a month	every a month	every 4 days
Maintainers	40	4	64	520
Dependencies	48	17	13	25
Monthly downloads	2,261,042	2,038,947	3,387,902	14,960,064
Open issues	322	135	20	491
Open pull requests	30	20	0	30
GitHub stars	12,115	11,837	29,982	42,790
Subscribers	318	570	1,239	1,542
Forks	1,059	1,544	4,353	5,407

1. Frameworks

What's a framework ?

- colectie de unelte, conventii si librarii JavaScript care pune la dispozitia programatorilor o anumita modalitate de organizare la nivel arhitectural si cod pre-scris pentru a fi folosit in dezvoltarea de zi cu a zi a *task*-urilor si functionalitatilor asociate unei aplicatii web

- in general, fiecare *framework* JavaScript dispune de un anumit *tool* care creaza structura initiala a unei aplicatii web care urmeaza fi dezvoltata pe baza acestuia (organizarea fisierelor, librarii pre-instalate, cod pre-scris, unelte pentru *local development* si *workflow automation*, *task*-uri si comenzi pentru optimizare si crearea *build*-ului pentru *production*, etc…)

  Exemple de framework-uri populare:

  - **Angular**
  - **React**
  - **Vue**

PRACTICE: **Bundlers & Frameworks**
React.js