webdesign, javascript, node.js, npm, grunt, modernizr, windows

Les objectifs

Il n'est pas question ici de tutos CSS, php ou `js`... Je n'ai pas le niveau pour
Il s'agit seulement de ma découverte des outils de développement et la manière de les mettre en place pour en tirer le meilleur parti en simplifiant autant que possible le développement d'un CSS, au minimum, propre (comprenez “avec les préfixes vendeurs qu'il faut” et optimisé (comprenez “minifié”[1]).



La mise en place des outils présentés ci-dessous prends pas mal de temps et peut coûter quelques cheveux blancs donc il faut bien peser le pour et le contre avant de se lancer. **De mon point de vue**, si on peut éviter de perdre du temps parce qu'on a raté tel ou tel préfixe, ça vaut le coup mais c'est une question de choix.

Les préfixes vendeurs, vous connaissez ?

Le principe est le suivant : les éditeurs de navigateurs web intègrent souvent les nouvelles normes CSS en adaptant le nom à leur sauce (les fameux préfixes) et parfois en modifiant la syntaxe.

Les principaux préfixes sont :

- -o- pour Opera
- -moz- pour Gecko (Mozilla)
- -webkit- pour Webkit (Chrome, Safari, Android...)
- -ms- pour Microsoft (Internet Explorer)

Et cela donne par exemple :

```
.boite {
  -moz-border-radius:5px;
  -webkit-border-radius:5px;
  border-radius:5px;
}
```

OK, là c'est pas trop difficile mais en voici un plus casse pieds :

```
.main-sidebar {
  -webkit-box-flex: 1;      /* OLD - iOS 6-, Safari 3.1-6 */
  -moz-box-flex: 1;         /* OLD - Firefox 19- */
  width: 20%;               /* For old syntax, otherwise collapses. */
```

```
  -webkit-flex: 1;           /* Chrome */
  -ms-flex: 1;               /* IE 10 */
  flex: 1;                   /* NEW, Spec - Opera 12.1, Firefox 20+ */
}
```

Donc non seulement il faut apprendre à maîtriser les CSS mais en plus il faut connaître les bons préfixes et les petites subtilités de syntaxes.

Mais c'est pas tout... Heureusement et malheureusement, ces préfixes évoluent et les éditeurs de navigateurs tendent à s'approcher au fur et à mesure de la norme telle qu'elle a été proposée, et chacun à son rythme. Autrement dit un préfixe indispensable à un instant T peut ne plus l'être un an plus tard.

Et c'est là que j'ai découvert un outil magique : Autoprefixer. On lui donne du CSS de base (càd. ne contenant que des règles de style “aux normes”) et il recrache du CSS avec les préfixes en cours. Cerise sur le gâteau: si on lui donne une règle avec un préfixe obsolète (parce que le moteur de navigation supporte maintenant la syntaxe standard), il la supprime.

> Attention à ne pas vous précipiter sur votre moteur de recherche préféré en quête d'Autoprefixer car il est tombé en désuétude au profit de PostCSS, un projet plus large incluant, entre autres, Autoprefixer.

Les fioritures

En terme de web, il me semble que si il y a un moyen simple d'économiser quelques kilos octets de données à télécharger en “*minifiant*” le CSS (càd en supprimant les commentaires, les espaces et les sauts de lignes inutiles), il faut le faire. Utiliser un service en ligne pour le faire c'est faisable une fois de temps en temps mais pas pour un projet en cours de développement. Installer un outil spécialisé n'est pas rentable sauf si on installe déjà par ailleurs un outil comme `Autoprefixer`...

Le CSS est une langue vivante mais les recommandations sont souvent très en avance sur les éditeurs de navigateurs. Donc il est risqué d'utiliser du code CSS trop frais... Sauf si un outil est capable de détecter ces syntaxes pour les normaliser comme `cssnext`.

Il peut être très intéressant de détecter les fonctionnalités du navigateur web des visiteurs d'un site web afin d'adapter le CSS. Le plus simple pour cela est d'utiliser la librairie Modernizr. Ce module Javascript teste les fonctionnalités choisies et ajoute des classes au tag `<html>`.

Et ainsi de suite.

L'environnement de travail

Avant de se lancer, il vaut mieux prendre quelques minutes pour penser aux éléments dont on va avoir besoin :

- un moteur de site web dédié au développement (on peut être amené à bidouiller les options et on ne fait pas ce genre de choses sur un serveur de production)
- un site web lui aussi dédié au développement (on teste d'abord)
- un traitement de texte digne de ce nom (avec au minimum la numérotation des lignes et une mise en couleurs de la syntaxe des langages que l'on va utiliser)
- un ordinateur disposant d'un OS sur lequel on pourra utiliser tous les outils dont on a besoin (si on doit passer sa vie à transférer des fichiers entre plusieurs machines on ne va pas “tenir la distance”. 😳

On peut très bien choisir de travailler par exemple sur une machine Windows alors que le site web de travail est situé sur un serveur Linux. Le tout est de savoir où l'on va.



Pour ma part, j'ai opté pour une machine Windows hébergeant un site web local (`http://localhost`) dont le chemin physique est `D:\www.dev`. Attention : il est tentant d'utiliser une simple lettre de lecteur plutôt qu'un chemin comme disons `T:\`, qu'elle pointe vers un support local ou un partage distant, mais d'après mon expérience, le moteur Apache (et ce n'est sans doute pas le seul) n'aime pas qu'il y ait un “/” ou un “\” directement à la fin d'un chemin donc placer son site web directement à la racine d'un chemin n'est pas une bonne idée car dans l'exemple il sera racourci à `T:` qui ne mène nulle part.

Un serveur web dédié

Une des particularités de l'univers du web, c'est que ça bouge vite, très vite même.

La majorité des serveurs web de “production”, qu'il s'agisse d'un serveur hébergé ou d'un serveur à domicile, tournent sur des machines utilisant des solution éprouvées et stables (par exemple une Debian avec Apache et PHP5) ce qui n'est pas l'idéal pour développer (sinon, le jour où l'on va avoir besoin par exemple de passer à PHP7, on risque fort d'entrer tout droit dans un mur).

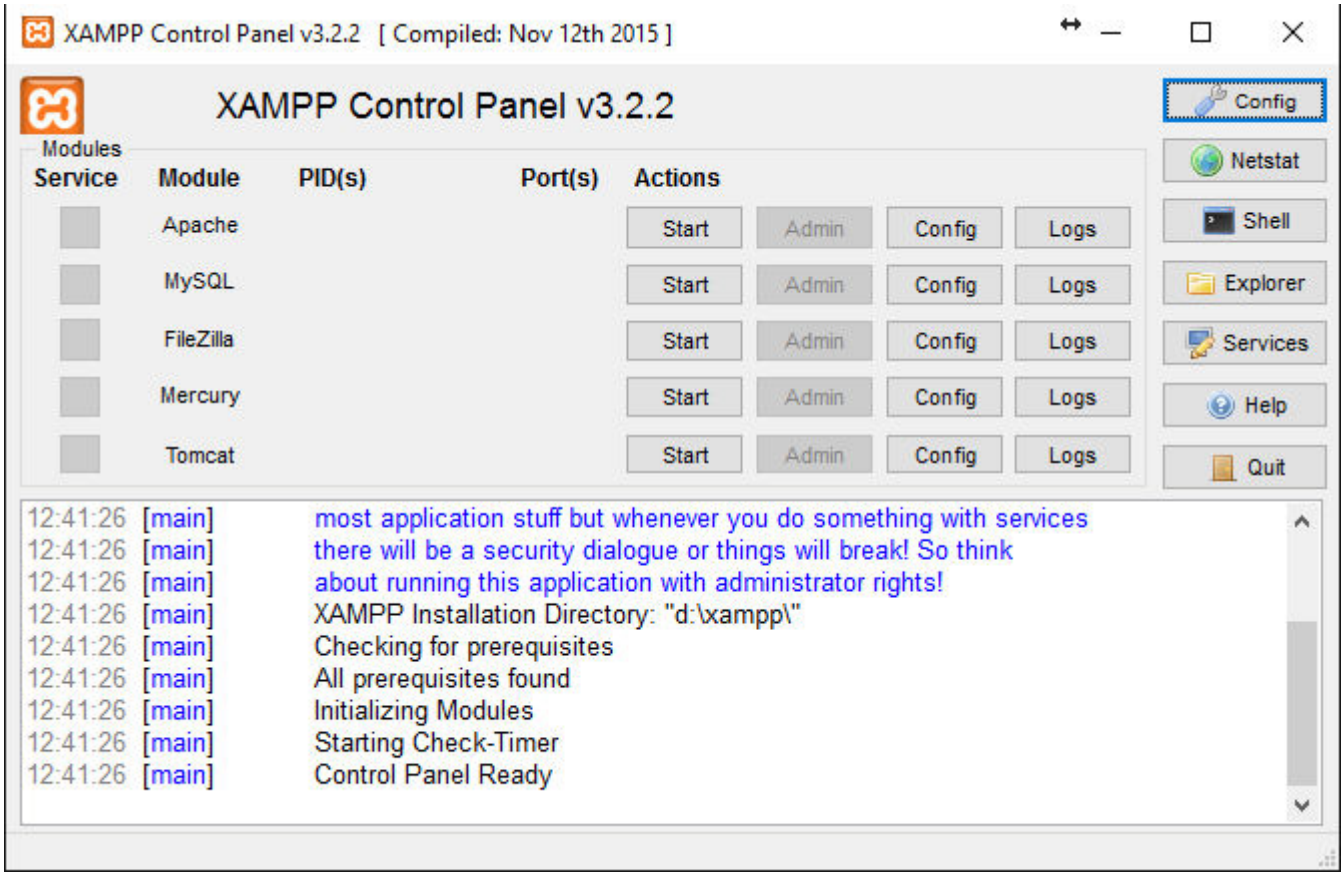
Donc pour avoir un environnement à la pointe des dernières technologies et développer en toute sérénité et de manière pro-active, on peut déjà écarter la solution hébergée (je doute qu'il en existe qui proposent un serveur web tournant avec les dernières versions, voire les versions beta, des outils indispensables). On peut évidement choisir de monter une machine locale maintenue à jour mais cela demande un boulot absolument vraiment énorme.

Le plus simple consiste selon moi à installer les outils nécessaires sur un ordinateur de bureau. J'avais besoin, pour la future version de Dokuwiki, d'un environnement PHP7 et puisqu'il existe une version de XAMPP avec cette version de PHP et tournant sous Windows, c'est celle que j'ai choisie.

XAMPP

C'est un pack de logiciels regroupant tout ce qu'il faut pour un site web: Apache, MySQL, ...

Voici à quoi cela ressemble lorsque l'on lance le ***Control Panel*** :

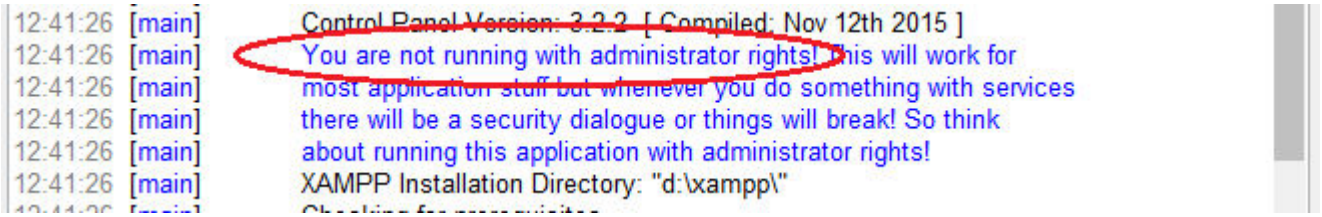


On a donc un accès rapide et très simple aux différents logiciel du pack (surtout avec l'icône *XAMPP* qui se loge dans la barre de tâches Windows au lancement).

Avant de commencer à jouer

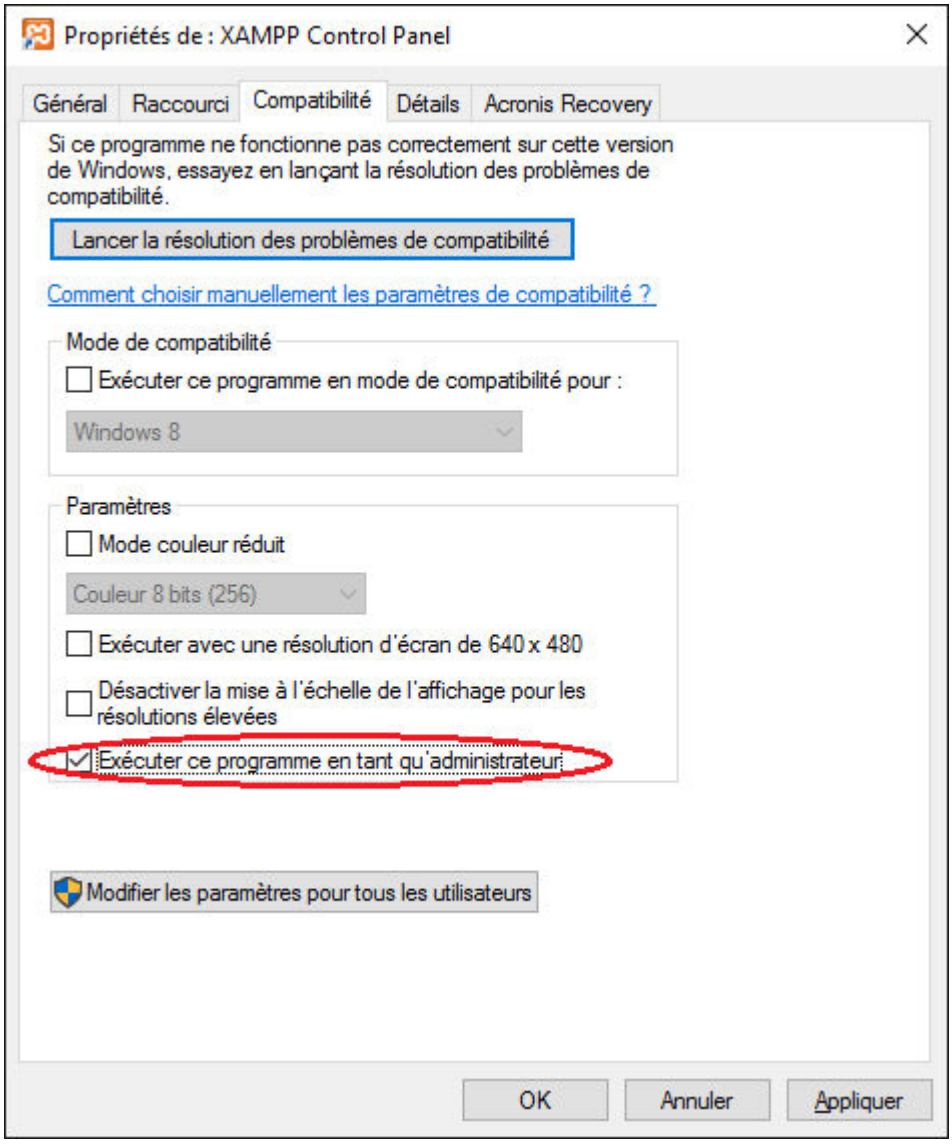
Lancer systématiquement XAMPP en tant qu'administrateur

Si l'on est attentif aux messages affichés par XAMPP, voici ce que l'on peut voir :



Le développement web est assez compliqué en lui-même pour ne pas se mettre de bâtons dans les roues avec un Windows 8 ou 10 qui bloque certaines actions. Donc la première chose à faire est de créer un raccourcis *XAMPP* qui sera toujours “lancé en tant qu'administrateur”.

Il suffit pour cela d'aller dans les l'onglet [Compatibilité] des propriétés du raccourci pour cocher la case [Exécuter ce programme en tant qu'administrateur] :



Il ne reste plus qu'à valider puis relancer *XAMPP*.

Adapter XAMPP à vos besoin

Ce n'est pas à vous de vous adapter à *XAMPP* (entendez par là utiliser le dossier qu'il propose pour votre site web local tout frais tout neuf (à savoir “C:/xampp/htdocs” si vous l'avez installé dans son dossier par défaut) mais l'inverse.

On peut évidement abandonner le site web inclut par défaut mais il est pratique pour certaines petites choses (comme l'accès très simple à `phpinfo()`) donc il me semble préferrable d'ajouter à *XAMPP* un second site et ce n'est pas aussi simple qu'on pourrait le penser :

- arrêter le service Apache de *XAMPP*

- créer l'arborescence de travail, dans mon cas `D:\www.dev\xampp` et `D:\www.dev\mon_site`
- s'assurer que l'on possède un accès complet à l'arborescence
- coller le contenu du dossier *xampp\htdocs* dans `D:\www.dev\xampp` et un fichier *index.html* minimaliste dans `D:\www.dev\mon_site`
- ouvrir le fichier *xampp\apache\conf\httpd.conf* :
  - y ajouter après la ligne `Listen 80` une autre ligne indiquant à Apache d'écouter sur un second port (pour le second site web), par exemple : `Listen 8880`
  - remplacer la valeur `DocumentRoot` et la directive `Directory` qui se trouve juste après pour pointer sur le chemin tel qu'il est vu par la machine *XAMPP*, par exemple :

```
DocumentRoot "D:/www.dev"
<Directory "D:/www.dev/xampp">
```

- après la ligne `</Directory>` qui signale la fin de la section éditée ci-dessus, ajouter ceci pour le second site :

```
<Directory "D:/www.dev/mon_site">
    Options Indexes FollowSymLinks Includes ExecCGI
    AllowOverride All
    Require all granted
</Directory>
```

- éditer ensuite le fichier *xampp\apache\conf\extra\httpd-vhosts.conf* pour qu'il ressemble à ceci :

```
NameVirtualHost *:80
NameVirtualHost *:8880

<VirtualHost *:80>
    ServerAdmin webmaster@mon_site.local
    DocumentRoot "D:/www.dev/mon_site"
</VirtualHost>
<VirtualHost *:8880>
    ServerAdmin webmaster@xampp.local
    DocumentRoot "D:/www.dev/xampp"
</VirtualHost>
```



Je vais plus souvent consulter `mon_site` que le site `xampp` donc autant me faciliter la vie en lui attribuant le port 80 (et donc éviter de devoir inclure le port dans l'adresse).



Attention à bien utiliser le “/” et pas “\” pour les chemins dans les fichiers *XAMPP* édités ci-dessus (même sous Windows).

Il ne reste plus qu'à redémarrer le service Apache de *XAMPP* et l'on a accès à `mon_site` à l'adresse http://localhost et au site XAMPP à l'adresse http://localhost:8880.



Notez qu'à ce stade, si l'un ou l'autre des services XAMPP né démarre pas, il y a fort à parier que c'est parce qu'il est configuré pour utilisé un port déjà oxxupé par un autre logiciel.

Node.js et ses amis

Je ne vais pas me risquer à entrer dans le détail car je ne maîtrise pas du tout (loin de là)... 

Tous les outils dont il est question sur cette page sont développés en Javascript, un langage qui n'est pas capable de dialoguer directement avec un OS et pour lequel il faut utiliser un moteur Javascript. Par ailleurs, plutôt que de développer à chaque fois un logiciel complet, les développeurs créent des packages qui utilisent les fonctions de packages d'autres développeurs et ainsi de suite. C'est ce qui explique que pour utiliser tous ces outils dont il est question ici, il faut créer une plateforme Javascript, j'ai même vu plusieurs fois le terme “écosystème”.

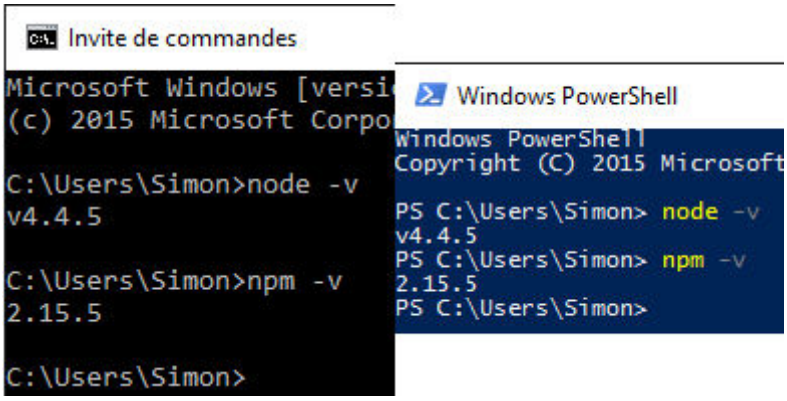
A chaque niveau de cet écosystème, il faut choisir quel outil utiliser... Mais, lorsque l'on n'a pas les connaissances nécessaire pour décider, il suffit de se laisser guider par les piste laissées par chaque développeur pour utiliser tel ou tel package.

Voici donc les différents éléments de la plateforme Javascript que j'ai mise en place: Node.js, npm et GRUNT.

Node.js et npm

J'ai déjà expérimenté l'installation sous Debian et ce n'était pas de la tarte, mais sous Windows, aucun soucis : on télécharge le logiciel d'installation directement depuis la la page d'accueil de Node.js et il fait tout tout seul (y compris installer `npm` dans la foulée et modifier automatiquement la variable PATH de l'OS).

Que l'on soit sous Windows ou Linux, `Node.js` et `npm` sont des outils en ligne de commande (sous Windows, on peut aussi bien utiliser la ligne de commande classique que le PowerShell (le second me semble le meilleur choix pour la colorisation syntaxique qui est toujours bonne à prendre) :



L'idéal est de ce créer un raccourci vers l'interface de commandes choisie en modifiant le dossier par défaut pour que l'on arrive directement dans le bon dossier à l'ouverture (le dossier contenant notre site web de travail).

GRUNT

L'installation est très simple puisqu'elle est gérée directement depuis la ligne de commande via `npm` ou `powershell` :

<cli>PS C:\> cd .\ProgramData\ PS C:\ProgramData> npm install -g grunt-cli C:\Users\sdela\AppData\Roaming\npm\grunt → C:\Users\sdela\AppData\Roaming\npm\node_modules\grunt-cli\bin\grunt + grunt-cli@1.3.2 added 150 packages from 121 contributors in 15.23s PS C:\ProgramData></cli>



Notez que les packages s'installent dans le dossier `…AppData\Roaming…` de l'utilisateur ce qui peut devenir problématique avec un disque SSD de petite taille.



L'instruction `npm install -g grunt-cli` (en se plaçant en powershell dans le dossier `c:\ProgramData`) permet la mise à jour de Grunt.

Le projet

Préparer un fichier package.json :

```
{
  "name": "namespaced",
  "version": "1.0.0",
  "description": "Namespaced DokuWiki template CSS processing",
```

```
  "keywords": [
    "DokuWiki",
    "template",
    "Namespaced"
  ],
  "author": "Simon Delage",
  "license": "GPL-3.0",
  "main": "Gruntfile.js",
  "scripts": {
  },
  "repository": {
    "type": "git",
    "url":
"git+https://github.com/geekitude/dokuwiki-template-namespaced.git"
  },
  "bugs": {
    "url":
"https://github.com/geekitude/dokuwiki-template-namespaced/issues"
  },
  "homepage":
"https://github.com/geekitude/dokuwiki-template-namespaced#readme",
  "devDependencies": {
  },
  "dependencies": {}
}
```

Chaque groupe d'actions (par exemple : ouvrir les fichiers de style et appliquer telle ou telle modification) constitue un projet représenté par un fichier descriptif `package.json` et un script de tâche(s) `Gruntfile.js`, tous deux à placer à la racine du projet en question (par exemple à la racine du site web concerné) et toutes les commandes qui suivent et qui concerne le projet doivent donc être lancées dans le PowerShell à cet emplacement (en terme de répertoire actif).

Taper successivement ces commandes Powershell : <cli>Windows PowerShell Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\Users\sdela> cd G:\www.dev\dokuwiki\lib\tpl\spacious PS G:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt –save-dev npm notice created a lockfile as package-lock.json. You should commit this file. + grunt@1.0.4 added 97 packages from 63 contributors and audited 179 packages in 15.004s found 0 vulnerabilities

PS G:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt-contrib-watch –save-dev + grunt-contrib-watch@1.1.0 added 23 packages from 28 contributors and audited 223 packages in 9.083s found 0 vulnerabilities

PS G:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt-contrib-cssmin –save-dev + grunt-contrib-cssmin@3.0.0 added 14 packages from 41 contributors and audited 249 packages in 8.568s found 0 vulnerabilities

PS G:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt-contrib-uglify –save-dev + grunt-contrib-uglify@4.0.1 added 4 packages from 4 contributors and audited 277 packages in 1.902s found 0 vulnerabilities

PS G:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt-autoprefixer –save-dev npm WARN deprecated browserslist@0.4.0: Browserslist 2 could fail on reading Browserslist >3.0 config used in other tools. + grunt-autoprefixer@3.0.4 added 17 packages from 41 contributors and audited 303 packages in 3.231s found 0 vulnerabilities

Windows PowerShell Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\Users\sdela> cd C:\www.dev\dokuwiki\lib\tpl\spacious PS C:\www.dev\dokuwiki\lib\tpl\spacious> npm init This utility will walk you through creating a package.json file. It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and save it as a dependency in the package.json file.

Press ^C at any time to quit. package name: (spacious) version: (1.0.0) description: Spacious DokuWiki template CSS processing entry point: (Gruntfile - Copie.js) Gruntfile.js test command: git repository: (https://github.com/geekitude/dokuwiki-template-spacious.git) keywords: DokuWiki template Spacious author: Simon DELAGE license: (ISC) GPL-3.0 About to write to C:\www.dev\dokuwiki\lib\tpl\spacious\package.json:

{

```
"name": "spacious",
"version": "1.0.0",
"description": "Spacious DokuWiki template CSS processing",
"main": "Gruntfile.js",
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
},
"repository": {
  "type": "git",
  "url": "git+https://github.com/geekitude/dokuwiki-template-spacious.git"
},
"keywords": [
  "DokuWiki",
  "template",
  "Spacious"
],
"author": "Simon DELAGE",
"license": "GPL-3.0",
"bugs": {
  "url": "https://github.com/geekitude/dokuwiki-template-spacious/issues"
},
"homepage": "https://github.com/geekitude/dokuwiki-template-spacious#readme"
```

}

Is this OK? (yes) PS C:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt –save-dev npm notice created a lockfile as package-lock.json. You should commit this file. + grunt@1.0.4 added 97 packages from 63 contributors and audited 179 packages in 11.84s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt-contrib-watch –save-dev + grunt-contrib-watch@1.1.0 added 23 packages from 28 contributors and audited 223 packages in 3.751s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt-postcss autoprefixer postcss-rtl –save-dev + postcss-rtl@1.5.0 + grunt-postcss@0.9.0 + autoprefixer@9.6.5 added 22 packages from 50 contributors and audited 297 packages in 5.022s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\spacious> grunt

> Local Npm module “grunt-autoprefixer” not found. Is it installed?

Running “watch” task Waiting… PS C:\www.dev\dokuwiki\lib\tpl\spacious> npm install grunt-autoprefixer –save-dev npm WARN deprecated browserslist@0.4.0: Browserslist 2 could fail on reading Browserslist >3.0 config used in other tools. + grunt-autoprefixer@3.0.4 added 18 packages from 40 contributors and audited 323 packages in 4.434s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\spacious> grunt Running “watch” task Waiting…

> File “css\src\spacious.theme.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 2.908s at Mon Oct 14 2019 16:42:32 GMT+0200 (GMT+02:00) - Waiting…

> File “css\src\spacious.theme.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 2.967s at Mon Oct 14 2019 16:44:31 GMT+0200 (GMT+02:00) - Waiting…

> File “css\src\spacious.theme.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 3.010s at Mon Oct 14 2019 16:45:52 GMT+0200 (GMT+02:00) - Waiting…

> File “css\src\spacious.plugins.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 2.910s at Mon Oct 14 2019 16:46:01 GMT+0200 (GMT+02:00) - Waiting...

> File “css\src\spacious.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 2.704s at Mon Oct 14 2019 16:46:16 GMT+0200 (GMT+02:00) - Waiting...

> File “css\src\spacious.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 2.951s at Mon Oct 14 2019 16:46:31 GMT+0200 (GMT+02:00) - Waiting...

> File “css\src\spacious.plugins.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 2.954s at Mon Oct 14 2019 16:46:38 GMT+0200 (GMT+02:00) - Waiting...

> File “css\src\spacious.theme.less” changed.

Running “postcss:dist” (postcss) task

> 3 processed stylesheets created.

Done. Completed in 2.984s at Mon Oct 14 2019 16:46:45 GMT+0200 (GMT+02:00) - Waiting...

Windows PowerShell Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\Users\sdela> cd C:\www.dev\dokuwiki\lib\tpl\namespaced PS C:\www.dev\dokuwiki\lib\tpl\namespaced> npm install grunt –save-dev npm notice created a lockfile as package-lock.json. You should commit this file. + grunt@1.0.4 added 97 packages from 63 contributors and audited 179 packages in 24.66s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\namespaced> npm install grunt-contrib-watch –save-dev + grunt-contrib-watch@1.1.0 added 23 packages from 28 contributors and audited 223 packages in 9.119s

found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\namespaced> npm install grunt-postcss –save-dev + grunt-postcss@0.9.0 added 4 packages from 38 contributors and audited 243 packages in 2.748s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\namespaced> npm install autoprefixer –save-dev + autoprefixer@9.7.3 added 11 packages from 8 contributors and audited 271 packages in 9.129s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\namespaced> npm install postcss-rtl –save-dev + postcss-rtl@1.5.0 added 6 packages from 6 contributors and audited 297 packages in 5.052s found 0 vulnerabilities

PS C:\www.dev\dokuwiki\lib\tpl\namespaced> </cli>

L'opération génère ces fichiers supplémentaires : `package.json` et `package-lock.json`

Il ne reste plus qu'à préparer le fichier `Gruntfile.js`

On peut ensuite lancer au même endroit un script grunt.ps1 contenant seulement ceci : grunt

et c'est parti : grunt surveille tous les fichiers du dossier src.

Dès qu'il détecte une modification, il exécute les tâches décrites dans le `Gruntfile.js`, c'est à dire un premier processus par PostCSS qui crée des fichiers dans dist puis ces fichiers sont minifiés vers le répertoire CSS.

1)

ce néologisme fait certainement hurler les linguistes mais moi je l'adore 🤣

From:
https://wiki.geekitude.fr/ - **Geekitude**

Permanent link:
https://wiki.geekitude.fr/info/webdesign/accueil?rev=1633927274

Last update: **2021/10/11 06:41**