À propos de l'éditeur

Cet article est publié par le magazine Le train de 13h37.
Le train de 13h37 est un magazine en ligne francophone dédié à la conception Web qui publie régulièrement des articles exclusifs rédigés par des auteurs invités et rémunérés.

Nous avons fait le choix d'une publication gratuite sur notre site, sous licence Creative Commons BY-NC-SA et non financée par la publicité.

Nous éditons aussi des livres papiers et numériques, avec les Éditions En Voiture Simone.

Magazine édité par la société WAGON 42 SAS, au capital de 5 000€ - SIRET : 752 205 856 000 12
25 Impasse Ernest Feuillet - 84000 Avignon // 09.81.61.08.07 – contact@wagon42.fr

Un Web orienté composants

Si vous avez eu l'occasion dans votre carrière d'utiliser Flash ou — soyons fous — des applets Java, vous êtes déjà familier avec la notion de composant. Il s'agit d'un élément de base pouvant être réutilisé, assemblé, combiné, étendu dans le but de faciliter un développement. Dans le cadre des *Web Components*, il s'agit d'éléments `HTML` embarquant leurs propres `CSS` et `JavaScript` afin d'avoir **nativement** un rendu et un comportement uniformes lors de leurs utilisations dans une page ou une application web. Un composant peut aller d'un simple bouton à un *widget* complet en passant par un élément non-visuel (comme une connexion *AJAX*) mais nous allons détailler tout cela par la suite.
Attention : les *Web Components* sont encore en cours de spécification et les *polyfills* comme *X-Tag* ou *Polymer* dont il est question dans la suite de l'article évoluent très vite. Il est pour l'instant déconseillé d'utiliser ces technologies en production car elles sont relativement instables, on vous aura prévenu(e)s !

Des concepts en cours de standardisation…

Poussés par l'équipe Chrome de Google, les *Web Components* sont en cours de standardisation via le processus classique du W3C, à l'état de brouillon. Le modèle de composant se découpe en 5 parties :

- **Les templates** : permettent de définir des fragments de `HTML` pour une utilisation future sans qu'ils soient interprétés ;
- **Les décorateurs** : pour styler les *templates* sans que les styles `CSS` s'appliquent aux autres éléments de la page ;
- **Les custom elements** : pour permettre aux développeurs de définir leurs propres éléments avec leurs propres tags `HTML` ;
- **Le Shadow DOM** : qui encapsule un élément complet à la manière des widgets natifs des navigateurs ;
- **Les imports** : afin de pouvoir charger des *composants* depuis une adresse distante.

Note : d'autres innovations sont en cours d'exploration comme les *Pointers Events*, les *Web Animations* ou les *Model Driven View* mais ça commence à faire beaucoup pour un seul article !
À ces spécifications s'ajoute le fait de pouvoir observer un objet `JavaScript` grâce à `Object.observe` (ou *Mutation Observers* si vous voulez briller en société) et modifier le DOM à la volée.

…mais des fonctionnalités déjà utilisables !

Ces spécifications ne sont bien évidement pas encore implémentées dans les navigateurs — même très récents — c'est pourquoi des équipes de développeurs courageux codent en `JavaScript` des compléments appelés *polyfills* qui miment le comportement que devrait avoir chaque navigateur s'il implémentait les spécifications des *Web Components*. Ces ajouts deviendront inutiles lorsque les différents moteurs de rendus auront travaillé sur ces spécifications.
Il existe 2 projets pour utiliser dès maintenant les *Web Components* : X-Tag par Mozilla et Polymer par Google. Une compatibilité partielle est prévue entre ces deux projets et nous allons nous concentrer sur Polymer dans la suite de cet article.

Une nouvelle façon de penser ses pages

Si vous connaissez les concepts sous-jacents des CSS Orientées Objets, sachez que les *Web Components* vont encore plus loin en proposant de véritables composants indépendants, réutilisables et pleinement fonctionnels. Ils ne se limitent pas au style mais intègrent aussi le comportement, autrement dit la logique qui régit le cycle de vie du composant et ses interactions avec l'utilisateur.
Si ces dernières années ont fait la part belle aux *frameworks* `CSS` comme Bootstrap, Foundation ou Pure, **ces prochains mois vont voir l'émergence de *toolkits* proposant des briques de composants aux styles unifiés et aux comportement extensibles**. Certains projets comme Brick ou Quetzal ont d'ailleurs vu le jour dans ce sens. Mais assez de théorie et voyons voir ce à quoi ressemble concrètement un composant web.

Un exemple d'application

Nous allons créer un composant permettant d'afficher de manière synthétique les informations relatives à un dépôt Github sur ces propres pages. C'est ce que fait déjà plus ou moins le plugin GitHub jQuery Repo Widget dont nous allons nous inspirer.
Notre objectif est de pouvoir intégrer facilement le composant dans nos pages de la manière suivante :

```
<github-repository repository="scopyleft/web-components"></github-repository>
```

Cette simple ligne devant afficher le composant relatif au dépôt passé en paramètre avec les styles dédiés et un chargement dynamique des informations en JavaScript. La première étape est de créer notre page `HTML 5`, appelons-la dans un excès d'originalité `index.html` :

```
<!DOCTYPE html>
<html>
  <head>
    <script src="polymer.min.js"></script>
    <link rel="import" href="github-repository.html">
  </head>
  <body>
    <github-repository repository="scopyleft/web-components"></github-repository>
  </body>
</html>
```

On commence par charger la bibliothèque `polymer.min.js` (ce fichier s'appelle actuellement `polymer-v0.0.20130912.min.js` si vous téléchargez l'archive directement sur le site) et on importe l'élément `github-repository` depuis une autre page web au nom éponyme (cela n'est pas obligatoire mais c'est une bonne pratique pour avoir des composants réutilisables). On peut maintenant lancer un serveur local, si vous avez Python il suffit de taper cette commande dans le dossier contenant le fichier `index.html` nouvellement créé :

```
$ python -m SimpleHTTPServer 8765
```

Si vous n'avez pas python, vous pouvez configurer un serveur Apache ou n'importe quelle solution permettant de servir des pages web depuis le dossier courant.
En accédant à `http://localhost:8765` vous devriez avoir une page blanche et la console devrait vous indiquer que l'URL `github-repository.html` vous renvoie une 404. Normal, nous ne l'avons pas encore créée ! Créons ce fichier localement en commençant notre composant

avec un attribut `repository` puisque l'on souhaite le rendre dynamique :

<polymer-element name="github-repository" attributes="repository">
Il est ensuite composé de 2 parties, la première contenant les styles et le html sous la balise `template` :
<template>
<style>
@host {
:scope
{font-family:helvetica,arial,sans-serif;font-size:13px;line-height:18px;background:#fafafa;border:1
px solid #ddd;color:#666;border-radius:3px}
:scope a{color:#4183c4;border:0;text-decoration:none}
:scope.github-box-title{position:relative;border-bottom:1px solid #ddd;border-radius:3px 3px 0
0;background:#fcfcfc;background:-moz-linear-gradient(#fcfcfc,#ebebeb);background:-webkit-linear
-gradient(#fcfcfc,#ebebeb);}
:scope.github-box-title
h3{font-family:helvetica,arial,sans-serif;font-weight:normal;font-size:16px;color:gray;margin:0;pad
ding:10px 10px 10px
30px;background:url('
AMAAAAx3e/WAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAyR
pVFh0WE1MOmNvbS5hZG9iZS54bXAAAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVz
VNME1wQ2VoaUh6cmVTek5UY3prYzlkIj8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6b
nM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUCBDb3JlIDUuMC1jMDYxIDY0LjE0MDk0OSwg
MjAxMC8xMi8wNy0xMDo1NzowMSAgICAgICAgIj4gPHJkZjpSREYgeG1sbnM6cmRmPSJod
HRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj4gPHJkZjpEZXNjcml
wdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vbnMuYWRvYmUuY29tL3hhc
C8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wL21tLyIgeG
1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS9SZXNvdXJjZ
VJlZiMiIHhtcDpcmVhdG9yVG9vbD0iQWRvYmUgUGhvdG9zaG9wIENTNS4xIE1hY2ludG9z
aCIgeG1wTU06SW5zdGFuY2VJRD0ieG1wLmlpZDpEQjIyNkJERkM0NjYxMUUxOEFDQzk3O
DcxRDkzRjhCRSIgeG1wTU06RG9jdW1lbnRJRD0ieG1wLmRpZDpEQjIyNkJFMEM0NjYxMU
UxOEFDQzk3ODcxRDkzRjhCRSI+IDx4bXBNTTpEZXJpdmVkRnJvbSBzdFJlZjppbnN0YW5jZ
UlEPSJ4bXAuaWlkOkRCMjI2QkREQzQ2NjExRTE4QUNDOTc4NzFEOTNGOEJFIiBzdFJlZjpk
b2N1bWVudElEPSJ4bXAuZGlkOkRCMjI2QkRFQzQ2NjExRTE4QUNDOTc4NzFEOTNGOEJFI
i8+IDwvcmRmOkRlc2NyaXB0aW9uPiA8L3JkZjpSREY+IDwveDp4bXBtZXRhPiA8P3hwYWNr
ZXQgZW5kPSJyIj8+dka2KgAAAEVQTFRFxMTEyMjI0tLSvb29vr6+zc3Ny8vLxcXFz8/P6enp3t
7ex8fH0dHR1NTUw8PDwMDAzs7OvLy8wcHBu7u7v7+/zMzM////budQFwAAABd0Uk5T/////////
//////////////////wDmQOZeAAAAcklEQVR42tSQSQ7DMAwD6chOukWs5eX/Ty2coo0T9wOdEzE
gdRBuzNmnDofgja52JDyz5TCqUp0O6kfrb4bzSXkRiTviEZZ6JKLMJ5VQ2v8iGbtbfEwXmjFM
G0VwdQo10hQNxYqtLMv9O6xvpZ/QeAkwAKjwHiJLaJc3AAAAAElFTkSuQmCC') 7px center
no-repeat}
:scope.github-box-title h3.repo{font-weight:bold}
:scope.github-box-title.github-stats{position:absolute;top:8px;right:10px;background:white;border:
1px solid #ddd;border-radius:3px;font-size:11px;font-weight:bold;line-height:21px;height:21px}
:scope.github-box-title.github-stats a{display:inline-block;height:21px;color:#666;padding:0 5px 0
18px;background:
url('

AAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAyRpVFh0WE1MOmN
vbS5hZG9iZS54bXAAAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh
6cmVTek5UY3prYzlkIj8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iIHg
6eG1wdGs9IkFkb2JlIFhNUCBDb3JlIDUuMC1jMDYxIDY0LjE0MDk0OSwgMjAxMC8xMi8wN
y0xMDo1NzowMSAgICAgICAgICAgIj4gPHJkZjpSREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3Lncz
Lm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6Y
WJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuc
zp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wL21tLyIgeG1sbnM6c3RSZWY9I
mh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS9SZXNvdXJjZVJlZiMiIHhtcDpDcm
VhdG9yVG9vbD0iQWRvYmUgUGhvdG9zaG9wIENTNS4xIE1hY2ludG9zaCIgeG1wTU06SW5
zdGFuY2VJRD0ieG1wLmlpZDpEQjIyNkJEQkM0NjYxMUUxOEFDQzk3ODcxRDkzRjhCRSIge
G1wTU06RG9jdW1lbnRJRD0ieG1wLmRpZDpEQjIyNkJEQ0M0NjYxMUUxOEFDQzk3ODcxR
DkzRjhCRSI+IDx4bXBNTTpEZXJpdmVkRnJvbSBzdFJlZjppbnN0YW5jZUlEPSJ4bXAuaWlkO
kRCMjI2QkQ5QzQ2NjExRTE4QUNDOTc4NzFEOTNGOEJFIiBzdFJlZjpkb2N1bWVudElEPSJ4
bXAuZGlkOkRCMjI2QkRBQzQ2NjExRTE4QUNDOTc4NzFEOTNGOEJFIi8+IDwvcmRmOkRl
c2NyaXB0aW9uPiA8L3JkZjpSREY+IDwveDp4bXBtZXRhPiA8P3hwYWNrZXQgZW5kPSJyIj8
+h1kA9gAAAK5QTFRF+fn5sbGx8fHx09PTmpqa2dnZ/f3919fX9PT00NDQ1dXVpKSk+vr6+/v7
vb298vLycnJ8/PztLS0zc3N6enp/v7+q6ur2NjY9/f3srKy/Pz8p6en7u7uoaGhnJyc4eHhtbW1pqam6
Ojo9fX17e3toqKirKys1NTUzs7Ox8fHwcHBwMDA5eXlnZ2dpaWl0dHR9vb25ubm4uLi3d3dqqq
qwsLCv7+/oKCgmZmZ////8yEsbwAAAMBJREFUeNrE0tcOgjAUBuDSliUoMhTEvfdef9//xUQjg
aLX0Ium/ZLT/
+SkRPxZpGykvuf5VMJogy5jY9yjDHcWFhqlcRuHc4o6B1QK0BDg+hcZgNDh3NWTwzItH/bRr
hvT+g3zSxZkNGCZpoWGIbU0a3Y6zV5VA6keyeDxiw62P0gUqEW0FbDim4nVikFJbU2zZXyb
UEaxhCqOQqyh5/G0wpWICUwthyqwD4InOMuXJ7/gs7WkoPdVg1vykF8CDACEFanKO3aSYw
AAAABJRU5ErkJggg==') no-repeat}
 :scope.github-box-title.github-stats.watchers{border-right:1px solid #ddd}
 :scope.github-box-title.github-stats.forks{background-position:-4px -21px;padding-left:15px}
 :scope.github-box-content{padding:10px;font-weight:300}
 :scope.github-box-content p{margin:0}
 :scope.github-box-content.link{font-weight:bold}
 :scope.github-box-download{position:relative;border-top:1px solid
#ddd;background:white;border-radius:0 0 3px 3px;padding:10px;height:24px}
 :scope.github-box-download.updated{margin:0;font-size:11px;color:#666;line-height:24px;font-we
ight:300}
 :scope.github-box-download.updated strong{font-weight:bold;color:#000}
 :scope.github-box-download.download{position:absolute;display:block;top:10px;right:10px;height
:24px;line-height:24px;font-size:12px;color:#666;font-weight:bold;text-shadow:0 1px 0
rgba(255,255,255,0.9);padding:0 10px;border:1px solid
#ddd;border-bottom-color:#bbb;border-radius:3px;background:#f5f5f5;background:-moz-linear-gra
dient(#f5f5f5,#e5e5e5);background:-webkit-linear-gradient(#f5f5f5,#e5e5e5);}
 :scope.github-box-download.download:hover{color:#527894;border-color:#cfe3ed;border-bottom-
color:#9fc7db;background:#f1f7fa;background:-moz-linear-gradient(#f1f7fa,#dbeaf1);background:-
webkit-linear-gradient(#f1f7fa,#dbeaf1);}
 }
 </style>

```
    <div class="github-box-title">
      <h3>
        <a class="owner" href="{{ ownerUrl }}">{{ login }}</a>
        /
        <a class="repo" href="{{ repoUrl }}">{{ name }}</a>
      </h3>
      <div class="github-stats">
        <a class="watchers" href="{{ watchersUrl }}">{{ watchers }}</a>
        <a class="forks" href="{{ forksUrl }}">{{ forks }}</a>
      </div>
    </div>
    <div class="github-box-content">
      <p class="description">{{ description }} &mdash; <a href="{{ readmeUrl }}">Read
More</a></p>
      <p class="link"><a href="{{ homepage }}">{{ homepage }}</a></p>
    </div>
    <div class="github-box-download">
      <p class="updated">Latest commit to the <strong>master</strong> branch on
{{ pushedAt }}</p>
      <a class="download" href="{{ downloadUrl }}">Download as zip</a>
    </div>
  </div>
</template>
```

Cela fait beaucoup d'information mais il ne faut pas s'affoler, les `CSS` sont assez verbeuses car elles intègrent les images mais sont juste là pour définir les styles propres à Github. Il s'agit de déclaration classiques qui peuvent être *inline* comme ici ou plus aérées. La seule particularité est relative à la combinaison du `@host` et du `:scope` qui permettent de restreindre les styles au composant, pratique lorsque l'on souhaite intégrer des composants de plusieurs provenances ! Il est possible de remplacer le `:scope` par un `id` sur la `div` principale mais cela est moins élégant (notez que la syntaxe pourrait changer prochainement).
La partie `HTML` est plus intéressante avec l'utilisation de variables écrites entre `{{ }}`, syntaxe avec laquelle vous êtes peut-être déjà familier. Celles-ci seront complétées directement avec le `JavaScript` qui est inséré à la suite du fichier dans une balise `<script>` :

```
<script>
  Polymer('github-repository', {
    width: '500px',
    created: function() {
      this.vendorName = this.repository.split('/')[0];
      this.repoName = this.repository.split('/')[1];
    },
    enteredView: function() {
      var url = 'https://api.github.com/repos/' + this.repository,
          httpRequest = new XMLHttpRequest(),
          self = this;
      httpRequest.open('GET', url, true);
      httpRequest.send();
```

```
    httpRequest.onreadystatechange = function() {
      if (this.status !== 200 || this.readyState !== 4) {
        return;
      }
      var repo = JSON.parse(this.responseText);
      self.login = repo.owner.login;
      self.ownerUrl = repo.owner.html_url;
      self.name = repo.name;
      self.repoUrl = repo.html_url
      self.watchers = repo.watchers;
      self.watchersUrl = self.repoUrl + "/watchers";
      self.forks = repo.forks;
      self.forksUrl = self.repoUrl + "/forks";
      self.description = repo.description;
      self.homepage = repo.homepage;
      if (repo.has_wiki) {
        self.readmeUrl = self.repoUrl + "#readme";
      };
      if (repo.pushed_at) {
        date = new Date(repo.pushed_at);
        function pad(n){ return n<10 ? '0'+n : n };
        self.pushedAt = date.getFullYear() + '-' + pad(date.getMonth() + 1) + '-' +
pad(date.getDate());
      }
      self.downloadUrl = self.repoUrl + "/zipball/master";
    };
  }
 });
</script>
```

On commence par définir la largeur par défaut du composant, avec une valeur par défaut de 500 pixels (le lecteur bidouilleur pourra tenter de rendre cette valeur dynamique en la passant en attribut). La fonction `created` est appelée à l'instanciation et permet de définir le nom d'utilisateur et le dépôt concernés.
La fonction `enteredView` est la plus intéressante car elle permet d'effectuer la requête `AJAX` récupérant les informations du dépôt et de remplacer *automagiquement* celles-ci dans le `template` précédemment défini.

On n'oublie bien évidemment pas de refermer la balise `</polymer-element>` à la fin de notre fichier. En rafraichissant la page, vous devriez apercevoir votre composant apparaître avec les informations relatives au dépôt que vous avez renseigné en attribut du composant (ici `scopyleft/web-components`).

Et la suite ?

Deux vidéos en anglais d'interventions réalisées dans le cadre de Google I/O font référence à ce jour et permettent de voir les exemples en pratique si vous n'êtes pas encore prêt(e) à mettre les mains dans le code :

- Web Components: A Tectonic Shift for Web Development
- Web Components in Action

Miško Hevery, l'un des co-créateurs d'AngularJS, s'est déjà prononcé en faveur d'une intégration partielle et d'une compatibilité avec les *Web Components* développé avec Polymer pour la version 2.0 du framework sur la liste de diffusion du développement de Polymer.
En ce qui concerne EmberJS, c'est plus compliqué en raison des problématiques de performances que cela entraîne mais l'objectif à long terme est d'être compatible également.

Autant d'encouragements pour développer (et partager) dès à présent votre *toolkit* de composants web afin de passer d'un développement atomique à un développement moléculaire ! Les *Web Components* ont certainement leur carte à jouer dans l'avenir du Web et de ses interactions.

À propos de l'auteur

David Larlet est un artisan, un geek et un citoyen.
Artisan, son métier est une passion qui lui permet de s'épanouir tout en véhiculant des valeurs qui lui sont chères. Il a co-créé scopyleft — une coopérative accompagnatrice de projets web — pour mettre à profit ses compétences et son expérience au service d'une aventure humaine, éthique et sociale.

Geek, il aime explorer des sujets et partager ces découvertes avec ses pairs. Il est intéressé par les outils et leurs usages, il participe à leur évolution et leur diffusion. Sa culture biologique lui permet de mettre en perspective certaines évolutions techniques et sociétales.

Citoyen, il prend soin de sa cité en lui accordant de l'attention. À une échelle plus large, il a conscience de son environnement et de son instabilité. Il est persuadé que la politique aurait un rôle à jouer dans l'individuation des personnes si elle était employée au service d'une vision partagée et consentie.

- Son site personnel
- @davidbgk

Magazine édité par la société WAGON 42 SAS, au capital de 5 000€ - SIRET : 752 205 856 000 12
25 Impasse Ernest Feuillet - 84000 Avignon // 09.81.61.08.07 – contact@wagon42.fr