**eBook Gratuit**

# APPRENEZ polymer

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#polymer

# Table des matières


À propos ..... 1

Chapitre 1: Commencer avec le polymère ..... 2

Remarques ..... 2

Versions ..... 2

Examples ..... 3

Bonjour le monde ..... 3

Utilisation d'éléments du catalogue de polymères ..... 4

Télécharger l'élément ..... 4

Tonnelle ..... 4

fichier zip ..... 4

Importer l'élément dans votre application ..... 4

Utilisez l'élément ..... 4

Structure élémentaire ..... 5

Configurer votre première application polymère à partir d'un modèle ..... 5

Installation de l'interface de ligne de commande du polymère ..... 6

Initialiser votre application à partir d'un modèle d'application ..... 6

Servez votre application ..... 6

Chapitre 2: Création d'une application à l'aide du kit de démarrage Polymer ..... 7

Introduction ..... 7

Syntaxe ..... 7

Examples ..... 7

Installation du kit de démarrage en polymère ..... 7

Chapitre 3: En boucle, le modèle dom-repeat. ..... 8

Examples ..... 8

Une liste de base ..... 8

Chapitre 4: Exemple de toggleAttribute de polymère ..... 9

Syntaxe ..... 9

Paramètres ..... 9

Remarques ..... 9

Examples ... 9
Exemple de base ... 9
Chapitre 5: Feuille de triche en polymère ... 11
Introduction ... 11
Examples ... 11
Définir un élément ... 11
Extension d'un élément ... 11
Définir un mixin ... 12
Méthodes de cycle de vie ... 13
Liaison de données ... 13
Observateurs ... 14
Chapitre 6: Gestion des événements ... 16
Remarques ... 16
Examples ... 16
Evénement à l'écoute à l'aide de l'écouteur Object ... 16
Auditeur annoté ... 17
Auditeur impératif ... 17
Événements personnalisés ... 18
Événement de changement de propriété ... 19
Reciblage des événements ... 20
Chapitre 7: Google Map Mark avec cache intégré ... 21
Syntaxe ... 21
Paramètres ... 21
Remarques ... 21
Examples ... 21
A - Commencer ... 21
Importer des dépendances ... 22
Remarque ... 22
Enregistrement de notre élément avec polymère ... 22
Ajouter un Blueprint à notre élément personnalisé - via des templates ... 23
B - Rendu du google map de base ... 23
Préparez votre navigateur - Polyfill it ... 23

Remarque: 23
Remarque: 25
Remarque: 26
C - Ajout d'une capacité de recherche à notre élément personnalisé 26
Remarque: 27
Accepter une entrée utilisateur en tant que requête de recherche 28
Ajouter un formulaire de recherche 28
Remarque: 30
Afficher les résultats de la recherche 31
D - Résultats de la recherche en cache 34
Remarque: 35
Remarque: 36
Le Polymère Google Map personnalisé complet avec cache intégré 37
Chapitre 8: Le débogage 39
Examples 39
Désactiver la mise en cache de l'importation HTML 39
Chapitre 9: Modaux réutilisables avec polymère 40
Syntaxe 40
Remarques 40
Examples 40
Modals 40
Quels sont nos objectifs? 41
Modals, Avec Polymère? 44
Remarque 44
Barre d'outils réutilisable avec modal | Partager | Icônes de fourche - Codage de l'élémen 45
Enregistrer l'élément personnalisé de la barre d'outils avec Modal | Partager | Icônes fou 46
Note: Icônes de fer de polymère 46
Remarque: utilisation de l'icône de fer et grammaire 47
Remarque: Rendre l'identifiant modal unique 48
Remarque: défilement du dialogue papier 51
L'image complète Élément personnalisé 52

Utiliser l'élément personnalisé avec modal | Partager | Icônes fourchette ... 53
Chapitre 10: SUPER-optimisation pour la production ... 57
Introduction ... 57
Syntaxe ... 57
Examples ... 57
Installer tous les outils requis ... 57
Utilisation ... 57
Mettre tous ensemble ... 57
En cours d'exécution vulcaniser et plus croustillant ... 57
Minifier les fichiers ... 58
import build.min.html ... 58
Chapitre 11: table de données de fer ... 59
Examples ... 59
Bonjour le monde ... 59
Import CSS ... 60
Détails de la ligne ... 61
Modifier les détails de la ligne ... 62
Modifier les détails de la ligne à l'aide d'un sous-élément ... 63
Remarque ... 65
Sélectionnez la ligne, empêchez la désélection ... 65
Remarque ... 65
Chapitre 12: Test d'unité ... 68
Remarques ... 68
Examples ... 68
Exemple simple utilisant Web-component-tester ... 68
installer ... 68
mise en place ... 68
fonctionnement ... 68
test / index.html ... 68
src / utils.html ... 69
Chapitre 13: Utilisation de bibliothèques Javascript externes avec Polymer ... 70

Introduction . . . 70
Paramètres . . . 70
Remarques . . . 70
Examples . . . 70
Importer un fichier HTML statique . . . 70
Chargement paresseux . . . 71
Crédits . . . 72

# À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: polymer

It is an unofficial and free polymer ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official polymer.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapitre 1: Commencer avec le polymère

## Remarques

Le projet Polymer comprend:

- Bibliothèque de polymères : Polymer est une bibliothèque légère qui vous permet de tirer pleinement parti des composants Web. Avec Web Components, vous pouvez créer des éléments personnalisés réutilisables qui interagissent de manière transparente avec les éléments intégrés du navigateur ou décomposez votre application en composants de taille appropriée, ce qui rend votre code plus propre et moins coûteux à gérer.
- WebComponents Polyfill : WebComponents Polyfill est une future bibliothèque ciblée destinée à répondre aux spécifications des composants Web du W3C. Les navigateurs qui implémentent complètement la spécification n'ont pas besoin de webcomponents.js. Cependant, certaines parties de la spécification manquent toujours à la plupart des navigateurs, donc cela dépendra pendant un certain temps.
- Polymer App Toolbox : Polymer App Toolbox vous aide à créer et à fournir des applications Web progressives de pointe avec un minimum de charge et de charge, en exploitant de puissantes fonctionnalités de plate-forme Web telles que Web Components, Service Worker et HTTP / 2. La boîte à outils fournit une architecture basée sur des composants, des mises en page réactives, un routeur modulaire, une prise en charge de la localisation, une prise en charge clé en main du stockage local et de la mise en cache hors ligne et une distribution efficace des ressources d'applications dégroupées. Adoptez ces fonctionnalités individuellement ou utilisez-les ensemble pour créer une application Web progressive complète. Les polymères saupoudrent un peu de sucre sur les API standard des composants Web, vous permettant ainsi d'obtenir de meilleurs résultats. La bibliothèque Polymer fournit un ensemble de fonctionnalités permettant de créer des éléments personnalisés. Ces fonctionnalités sont conçues pour faciliter et accélérer la création d'éléments personnalisés fonctionnant comme des éléments DOM standard. Semblable aux éléments DOM standard, les éléments Polymer peuvent être:

## Versions

| Version | Date de sortie |
|---|---|
| v2.0.0 | 2017-05-15 |
| v1.7.0 | 2016-09-28 |
| v1.6.1 | 2016-08-01 |
| v1.6.0 | 2016-06-29 |
| v1.5.0 | 2016-05-31 |

| Version | Date de sortie |
| --- | --- |
| v1.4.0 | 2016-05-18 |
| v1.0.0 | 2015-05-27 |

# Examples

## Bonjour le monde

Cet exemple crée un élément Polymer nommé `x-foo` , dont le modèle est lié à une propriété de chaîne nommée "message". Le HTML de l'élément est importé dans le document principal, ce qui permet d'utiliser les balises `<x-foo>` dans `<body>` .

*x-foo.html*

```
<dom-module id="x-foo">
  <template>
    <span>{{message}}</span>
  </template>

  <script>
    Polymer({
      is: 'x-foo',
      properties : {
        message: {
          type: String,
          value: "Hello world!"
        }
      }
    });
  </script>
</dom-module>
```

*index.html*

```
<head>
  <!-- PolyGit used here as CDN for demo purposes only. In production,
       it's recommended to import Polymer and Web Components locally
       from Bower. -->
  <base href="https://polygit.org/polymer+1.6.0/components/">

  <script src="webcomponentsjs/webcomponents-lite.min.js"></script>
  <link rel="import" href="polymer/polymer.html">

  <link rel="import" href="x-foo.html">
</head>
<body>
  <x-foo></x-foo>
</body>
```

Voir la démo dans CodePen

### Utilisation d'éléments du catalogue de polymères

Polymer produit de nombreux éléments bien conçus pour votre application.

Parcourez-les dans leur catalogue d'éléments .

Passons en revue le flux de travail de l'utilisation d'un élément en incluant `paper-input` ( Documentation )

# Télécharger l'élément

Pour télécharger un élément, il existe deux manières:

## Tonnelle

Le moyen le plus pratique consiste à utiliser la ligne de commande à l'aide de la commande d' `install` bower:

```
bower install --save PolymerElements/paper-input
```

**Remarque:** `--save` ajoute l'élément comme dépendance à `bower.json` de votre application.

## fichier zip

L'autre solution consiste à ajouter l'élément sélectionné ( `paper-input` dans ce cas) à votre collection (dans le catalogue Polymer) en utilisant "Ajouter à la collection" dans la navigation et à télécharger votre collection en utilisant l'icône en forme d'étoile dans le coin supérieur droit.

Cela générera un fichier .zip contenant l'élément et toutes ses dépendances. Vous pouvez ensuite copier le dossier `bower_components` dans le fichier .zip / dans le répertoire racine de votre application.

# Importer l'élément dans votre application

Pour importer l'élément que vous venez d'installer, importez le fichier `.html` correspondant:

```
<link rel="import" href="bower_components/paper-input/paper-input.html">
```

# Utilisez l'élément

Vous pouvez maintenant utiliser `paper-input` dans le document que vous avez importé pour:

```
<paper-input></paper-input>
```

## Structure élémentaire

Nous avons obtenu l'élément de base suivant, `my-element` enregistré sous le nom `src/my-element.html`

```
<link rel="import" href="bower_components/polymer/polymer.html">

<dom-module id="my-element">

  <template>
    <style>
      /* local styles go here */
      :host {
        display: block;
      }
    </style>
    <!-- local DOM goes here -->
    <content></content>
  </template>

  <script>
    Polymer({
      /* this is the element's prototype */
      is: 'my-element'
    });
  </script>

</dom-module>
```

- Le `<link>` inclut la bibliothèque Polymer utilisant une importation HTML.
- Le `<dom-module>` est le wrapper DOM local de l'élément (dans ce cas, `my-element` ).
- Le `<template>` est la définition DOM locale réelle.
- Le `<style>` à l'intérieur du `<template>` vous permet de définir des styles qui sont étendus à cet élément et à son DOM local et n'affecteront rien d'autre dans le document.
- Le `<content>` contiendra tout ce que vous placez dans votre élément.
- La pseudo-classe `:host` correspond à l'élément personnalisé ( `my-element` ).
- L'appel `Polymer` enregistre l'élément.
- L' `is` de la propriété est le nom de l'élément (il **doit** correspondre au `<dom-module>` de `id` )

Vous pouvez l'importer dans votre application en utilisant:

```
<link rel="import" href="src/my-element.html">
```

Et l'utiliser comme une balise:

```
<my-element>Content</my-element>
```

## Configurer votre première application polymère à partir d'un modèle

Préparez-vous à créer votre propre application Web progressive avec Polymer!

Avant de pouvoir installer Polymer, vous avez besoin des éléments suivants:

- Node.js - consultez la documentation de StackOverflow sur l' installation de Node.js
- Bower - vous pouvez installer Bower à l'aide du gestionnaire de paquet de nœuds installé avec Node.js:

```
npm install -g bower
```

# Installation de l'interface de ligne de commande du polymère

Le CLI Polymer vous fournit tous les outils nécessaires aux projets de polymères:

```
npm install -g polymer-cli
```

# Initialiser votre application à partir d'un modèle d'application

Utilisez `polymer init` pour initialiser votre application à partir d'un modèle d'application .

Un modèle cool est le modèle `--app-drawer-template` . Utilisons cela:

```
polymer init app-drawer-template
```

# Servez votre application

Il n'y a aucun bâtiment nécessaire pour servir votre première application Polymer. Juste le `serve` :

```
polymer serve --open
```

Cela ouvrira l'application dans votre navigateur par défaut sur `http://localhost:8080` .

Lire Commencer avec le polymère en ligne:
https://riptutorial.com/fr/polymer/topic/949/commencer-avec-le-polymere

# Chapitre 2: Création d'une application à l'aide du kit de démarrage Polymer

## Introduction

Polymer Starter Kit est une excellente solution pour commencer à créer des applications Web progressives . Il offre une interface utilisateur de conception matérielle réactive, de routage de pages, de service hors ligne.

## Syntaxe

- Polymère: Commande pour initier un polymère
- Init polymère: présente une liste de modèles à choisir pour créer votre élément / application
- npm: lance l'interface CLI du gestionnaire de packages Node.js
- npm install: installe un paquet
- npm install -g: installe un package globalement sur votre système. Utile pour les outils CLI

## Examples

### Installation du kit de démarrage en polymère

1. Assurez-vous que Polymer CLI est installé dans votre système globalement. Si ce n'est pas le cas, ouvrez l'interface de ligne de commande / terminal et exécutez cette commande: `npm install -g polymer-cli`

   Note: Si vous êtes un utilisateur Ubuntu, vous devrez peut-être préfixer le code ci-dessus avec le mot-clé `sudo` .

2. Après l'installation de Polymer CLI, exécutez cette commande

   cd [votre annuaire dans lequel vous voulez commencer votre projet]

```
polymer init
```

3. Vous aurez 4 options. Utilisez les touches fléchées pour aller au 4ème, c'est **-à-** dire le **kit de démarrage** . Appuyez sur la touche `Enter` .

   Tous les fichiers requis seront téléchargés dans votre répertoire sélectionné.

Lire Création d'une application à l'aide du kit de démarrage Polymer en ligne: https://riptutorial.com/fr/polymer/topic/9330/creation-d-une-application-a-l-aide-du-kit-de-demarrage-polymer

# Chapitre 3: En boucle, le modèle dom-repeat.

## Examples

### Une liste de base

Ceci est un élément polymère de base qui affiche une liste de noms.

```
<link rel="import" href="../bower_components/polymer/polymer.html">

<dom-module id="basic-list">
  <template>
    <style>
    </style>

    <div>Name's list</div>
    <template is="dom-repeat" items="{{list}}">
        <div>{{item.lastName}}, {{item.firstName}}</div>
    </template >

  </template>

  <script>
    Polymer({
      is: 'basic-list',

      properties:{
        list:{
          type: Array,
          value: function(){
            let list = [
              {firstName: "Alice", lastName: "Boarque"},
              {firstNName: "Carlos, lastName: "Dutra"}
            ]

            return list
          },
        },
      },
    });
  </script>
</dom-module>
```

Lire En boucle, le modèle dom-repeat. en ligne: https://riptutorial.com/fr/polymer/topic/6160/en-boucle--le-modele-dom-repeat-

# Chapitre 4: Exemple de toggleAttribute de polymère

## Syntaxe

1. toggleAttribute (name, bool, node)

## Paramètres

| prénom | Détails |
|---|---|
| prénom | String: nom de l'attribut HTML à modifier |
| bool | booléen: booléen pour activer ou désactiver l'attribut. Lorsqu'il n'est pas spécifié, l'état de l'attribut sera inversé. |
| nœud | HTMLElement: nom du noeud qui contient l'attribut HTML. Par défaut à cette |

## Remarques

Un bon exemple de ceci sera form, où le bouton submit ne devrait être actif que si tous les champs obligatoires ont été saisis.

## Examples

### Exemple de base

```
<script src='bower_components/webcomponentsjs/webcomponents-lite.min.js'></script>
<link rel='import' href='bower_components/polymer/polymer.html'>
<link rel='import' href='bower_components/paper-button/paper-button.html'>
<link rel='import' href='bower_components/paper-input/paper-input.html'>
<dom-module id='toggle-attribute'>
  <template>
    <style>
    </style>
    <paper-input id='input'></paper-input>
    <paper-button on-tap='_toggle'>Tap me</paper-button>
  </template>
</dom-module>
<script>
  Polymer({
    is:'toggle-attribute',
    properties:{
      isTrue:{
        type:Boolean,
        value:false
      }
```

```
    },
    _toggle:function(){
      this.isTrue = !this.isTrue;
      this.toggleAttribute('disabled',this.isTrue,this.$.input);
    }
  })
</script>
```

Voici un lanceur de course

Lire Exemple de toggleAttribute de polymère en ligne:
https://riptutorial.com/fr/polymer/topic/5978/exemple-de-toggleattribute-de-polymere

# Chapitre 5: Feuille de triche en polymère

## Introduction

Ceci est un aide-mémoire pour la bibliothèque Polymer 2.x. Fourche de poste de Monica Dinculescu.

## Examples

### Définir un élément

Docs: 1.x -> Guide de mise à niveau 2.x , enregistrement d'un élément , modules de style partagé .

```
<link rel="import" href="bower_components/polymer/polymer-element.html">
<dom-module id="element-name">
  <template>
    <!-- Use one of these style declarations, but not both -->
    <!-- Use this if you don't want to include a shared style -->
    <style></style>
    <!-- Use this if you want to include a shared style -->
    <style include="some-style-module-name"></style>
  </template>
  <script>
    class MyElement extends Polymer.Element {
      static get is() { return 'element-name'; }
      // All of these are optional. Only keep the ones you need.
      static get properties() { ... }
      static get observers() { ... }
    }

    // Associate the new class with an element name
    customElements.define(MyElement.is, MyElement);
  </script>
</dom-module>
```

Pour obtenir la définition de classe pour une balise personnalisée particulière, vous pouvez utiliser `customElements.get('element-name')` .

### Extension d'un élément

Docs: extension d'éléments , modèles hérités .

Au lieu de `Polymer.Element` , un élément personnalisé peut étendre un autre élément:

```
class ParentElement extends Polymer.Element {
  /* ... */
}
class ChildElement extends ParentElement {
  /* ... */
}
```

Pour modifier ou ajouter au modèle du parent, remplacez le getter de `template` :

```
<dom-module id="child-element">
  <template>
    <style> /* ... */ </style>
    <span>bonus!</span>
   </template>
  <script>
    var childTemplate;
    var childTemplate = Polymer.DomModule.import('child-element', 'template');
    var parentTemplate = ParentElement.template.cloneNode(true);
    // Or however you want to assemble these.
    childTemplate.content.insertBefore(parentTemplate.firstChild, parentTemplate);

    class ChildElement extends ParentElement {
      static get is() { return 'child-element'; }
      // Note: the more work you do here, the slower your element is to
      // boot up. You should probably do the template assembling once, in a
      // static method outside your class (like above).
      static get template() {
        return childTemplate;
      }
    }
    customElements.define(ChildElement.is, ChildElement);
  </script>
</dom-module>
```

Si vous ne connaissez pas la classe parente, vous pouvez également utiliser:

```
class ChildElement extends customElements.get('parent-element') {
  /* ... */
}
```

## Définir un mixin

Docs: mixins , éléments hybrides .

Définition d'une expression de classe mixin pour partager l'implémentation entre différents éléments:

```
<script>
  MyMixin = function(superClass) {
    return class extends superClass {
      // Code that you want common to elements.
      // If you're going to override a lifecycle method, remember that a) you
      // might need to call super but b) it might not exist.
      connectedCallback() {
        if (super.connectedCallback) {
          super.connectedCallback();
        }
        /* ... */
      }
    }
  }
</script>
```

Utiliser le mixin dans une définition d'élément:

```
<dom-module id="element-name">
  <template><!-- ... --></template>
  <script>
    // This could also be a sequence:
    // class MyElement extends AnotherMixin(MyMixin(Polymer.Element)) { … }
    class MyElement extends MyMixin(Polymer.Element) {
      static get is() { return 'element-name' }
      /* ... */
    }
    customElements.define(MyElement.is, MyElement);
  </script>
</dom-module>
```

Utiliser des comportements hybrides (définis dans la syntaxe 1.x) comme mixins:

```
<dom-module id="element-name">
  <template><!-- ... --></template>
  <script>
    class MyElement extends Polymer.mixinBehaviors([MyBehavior, MyBehavior2], Polymer.Element)
{
     static get is() { return 'element-name' }
     /* ... */
    }
    customElements.define('element-name', MyElement);
  </script>
</dom-module>
```

## Méthodes de cycle de vie

Docs: rappels de cycle de vie , prêts .

```
class MyElement extends Polymer.Element {
 constructor() { super(); /* ... */ }
 ready() { super.ready(); /* ... */ }
 connectedCallback() { super.connectedCallback(); /* ... */ }
 disconnectedCallback() { super.disconnectedCallback(); /* ... */ }
 attributeChangedCallback() { super.attributeChangedCallback(); /* ... */ }
}
```

## Liaison de données

Docs: liaison de données, liaison d' attributs , liaison aux éléments du tableau , liaisons calculées .

N'oubliez pas: les propriétés Polymer camel-cases , donc si vous utilisez en JavaScript `myProperty` , en HTML, vous utiliserez `my-property` .

Liaison à **sens unique** : lorsque `myProperty` change, leur `theirProperty` est mise à jour:

```
<some-element their-property="[[myProperty]]"></some-element>
```

Liaison **bidirectionnelle** : lorsque `myProperty` change, sa `theirProperty` est mise à jour et

inversement:

```
<some-element their-property="{{myProperty}}"></some-element>
```

**Liaison d'attribut** : lorsque `myProperty` `true` , l'élément est masqué; quand c'est `false` , l'élément est visible. La différence entre la liaison d'attribut et de propriété est que la liaison de propriété est équivalente à `someElement.someProp = value` , alors que la liaison d'attribut est équivalente à: `someElement.setAttribute(someProp, value)`

```
<some-element hidden$="[[myProperty]]"></some-element>
```

**Liaison calculée** : la liaison à l'attribut `class` recompile les styles lorsque `myProperty` change:

```
<some-element class$="[[_computeSomething(myProperty)]]"></some-element>
<script>
_computeSomething: function(prop) {
  return prop ? 'a-class-name' : 'another-class-name';
}
</script>
```

## Observateurs

Docs: observateurs , observateurs multi-propriétés , mutations d' observation , ajout dynamique d'observateurs .

L'ajout d'un `observer` dans le bloc de `properties` permet d'observer les modifications de la valeur d'une propriété:

```
static get properties() {
  return {
    myProperty: {
      observer: '_myPropertyChanged'
    }
  }
}

// The second argument is optional, and gives you the
// previous value of the property, before the update:
_myPropertyChanged(value, /*oldValue */) { /* ... */ }
```

Dans le bloc d' `observers` :

```
static get observers() {
  return [
    '_doSomething(myProperty)',
    '_multiPropertyObserver(myProperty, anotherProperty)',
    '_observerForASubProperty(user.name)',
    // Below, items can be an array or an object:
    '_observerForABunchOfSubPaths(items.*)'
  ]
}
```

Ajouter un observateur dynamiquement pour une propriété `otherProperty` :

```
// Define a method.
_otherPropertyChanged(value) { /* ... */ }
// Call it when `otherPropety` changes.
this._createPropertyObserver('otherProperty', '_otherPropertyChanged', true);
```

Lire Feuille de triche en polymère en ligne: https://riptutorial.com/fr/polymer/topic/10710/feuille-de-triche-en-polymere

# Chapitre 6: Gestion des événements

## Remarques

- Voici un plunker pour tous les exemples
- Le nom de l'attribut est insensible à la casse et sera toujours converti en minuscule, par exemple si un attribut `on-myListener` est défini sur l'événement `mylistener` .
- Semblable à `listen` vous pouvez également utiliser la méthode `unlisten` pour supprimer tout auditeur.
- Le changement de population déclenche un événement par le nom de `property-changed` par exemple si la propriété est `myProperty` événement sera `my-propert-changed` (camel `my-propert-changed` à '-'). Vous pouvez les écouter soit en utilisant l'attribut `on-event` de l' `listener` Object.
- La nouvelle valeur est toujours stockée dans `e.detail.value` pour les événements de modification de propriété.

## Examples

### Evénement à l'écoute à l'aide de l'écouteur Object

```
<dom-module id="using-listeners-obj">
  <template>
    <style>
      :host{
        width: 220px;
        height: 100px;
        border: 1px solid black;
        display: block;
      }

      #inner{
        width: calc(100% - 10px);
        height: 50px;
        border: 1px solid blue;
        margin: auto;
        margin-top: 15px;
      }
    </style>
    <div>Tap me for alert message</div>
    <div id="inner">Tap me for different alert</div>
  </template>
</dom-module>
<script>
  Polymer({
    is:'using-listeners-obj',

    //Listener Object
    listeners:{
      //tap a special gesture event in Polymer. Its like click event the main difference being
it is more mobile deivce friendly
      'tap':'tapped',  //this will get executed on host of the element
      'inner.tap':'divTapped' //this tap will get executed only on the mentioned node (inner
in this case)
```

```
    },

    tapped:function(){
      alert('you have tapped host element');
    },

    divTapped:function(e){
      event.stopPropagation();  //this is only to stop bubbling of event. If you comment this
then tap event will bubble and parent's(host) listener will also get executed.
      console.log(e);
      alert('you have tapped inner div');
    }
  })
</script>
```

## Auditeur annoté

Une autre façon d'ajouter un écouteur d'événement consiste à utiliser l'annotation `on-event` dans DOM. Vous trouverez ci-dessous un exemple d'utilisation `up` événement qui est déclenché lorsque le doigt / la souris monte

```
<dom-module id="annotated-listener">
  <template>
    <style>
      .inner{
        width: calc(200px);
        height: 50px;
        border: 1px solid blue;
        margin-top: 15px;
      }
    </style>
    <!-- As up is the name of the event annotation will be on-up -->
    <div class="inner" on-up='upEventOccurs'>Tap me for different alert</div>
  </template>
</dom-module>
<script>
  Polymer({
    is:'annotated-listener',
    upEventOccurs:function(e){
      //detail Object in event contains x and y co-ordinate of event
      alert('up event occurs at x:'+e.detail.x+' y:'+e.detail.y);
    }
  })
</script>
```

## Auditeur impératif

Vous pouvez également ajouter / supprimer l'écouteur impérativement en utilisant la méthode listen et unlisten de Polymer

```
    <dom-module id="imparative-listener">
      <template>
        <style>
          #inner{
            width: 200px;
            height: 50px;
```

```
            border: 1px solid blue;
            margin-top: 15px;
          }
        </style>
        <div id="inner">I've imparative listener attached</div>
      </template>
    </dom-module>
    <script>
      Polymer({
        is:'imparative-listener',

        //keeping it in attached to make sure elements with their own shadow root are attached
        attached:function(){
          this.listen(this.$.inner,'track','trackDetails'); // For more details on method
check https://www.polymer-project.org/1.0/docs/api/Polymer.Base#method-listen
        },

        //all the listener functions have event as first parameter and detail (event.detail)
as second parameter to the function
        trackDetails:function(e,detail){
          if(detail.state=='end'){
            alert('Track distance x:'+detail.dx+' y:'+detail.dy);// For more details on track
event check https://www.polymer-project.org/1.0/docs/devguide/gesture-events
      }
    }
  })
</script>
```

## Événements personnalisés

Vous pouvez également déclencher vos propres événements, puis les écouter depuis l'un des éléments Polymer de la page HTML.

Cet élément déclenche l'événement personnalisé

```
    <dom-module id="custom-event">
  <template>
    <style>
      #inner{
        width: 200px;
        height: 50px;
        border: 1px solid blue;
        margin-top: 15px;
      }
    </style>
    <div id="inner" on-tap="firing">I'll fire a custom event</div>
  </template>
</dom-module>
<script>
  Polymer({
    is:'custom-event',
    firing:function(){
      this.fire('my-event',{value:"Yeah! i'm being listened"}) //fire is the method which is
use to fire custom events, second parameter can also be null if no data is required
    }
  })
</script>
```

Voici un élément qui écoute cet événement personnalisé

```
  <link rel="import" href="custom-event.html">
 <dom-module id="custom-event-listener">
  <template>
    <style></style>
    <custom-event on-my-event="_myListen"></custom-event>
  </template>
</dom-module>
<script>
  Polymer({
    is:'custom-event-listener',
/*    listeners:{ //you can also use listener object instead of on-event attribute
      'my-event':'listen'
    },*/
    _myListen:function(e,detail){
      alert(detail.value+"from Polymer Element");
    },
  })
</script>
```

Écouter à partir de HTML

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Events</title>
    <script src='bower_components/webcomponentsjs/webcomponents-lite.min.js'></script>
    <link rel="import" href="./bower_components/polymer/polymer.html">
    <link rel="import" href="custom-event-listener.html">
  </head>
  <body>
    <!--As event bubbles by default we can also listen to event from custom-event-listener
instead of custom-event-->
    <custom-event-listener></custom-event-listener>
  </body>
  <script>
    document.querySelector('custom-event-listener').addEventListener('my-event',function(e){
      console.log(e);
      alert(e.detail.value+"from HTML");
    })
  </script>
</html>
```

## Événement de changement de propriété

Propriétés avec `notify:true` déclenche également un événement

```
<link rel="import" href="../bower_components/paper-input/paper-input.html">
<dom-module id="property-change-event">
  <template>
    <style></style>
    <paper-input id="input" label="type here to fire value change event" on-value-
changed='changeFired'></paper-input>
  </template>
</dom-module>
<script>
```

```
  Polymer({
    is:'property-change-event',
    changeFired:function(e){
      if(e.detail.value!="")
        alert("new value is: "+e.detail.value);
    }
  })
</script>
```

## Reciblage des événements

Il est possible de recibler un événement dans Polymer, c'est-à-dire que vous pouvez modifier les détails de l'événement comme `path` en masquant ainsi les détails réels d'un événement ou d'un élément.

Par exemple, si un élément `div` dans l'élément `event-retargeting` déclenche l'événement, mais que le développeur ne veut pas que l'utilisateur sache qu'il peut recibler l'événement à l'élément `event-retargeting` en utilisant le code suivant.

```
var targetEl = document.querySelector('event-retargeting');
var normalizedEvent = Polymer.dom(event);
normalizedEvent.rootTarget = targetEl;
normalizedEvent.localTarget =targetEl
normalizedEvent.path = [];
normalizedEvent.path.push(targetEl);
normalizedEvent.path.push(document.querySelector('body'));
normalizedEvent.path.push(document.querySelector('html'));
```

Pour voir l'exemple de travail, veuillez vous reporter à la section des `remarks` de `plunker` .

Lire Gestion des événements en ligne: https://riptutorial.com/fr/polymer/topic/4778/gestion-des-evenements

# Chapitre 7: Google Map Mark avec cache intégré

## Syntaxe

- <g-map

```
api-key="AIzaSyBLc_T17p91u6ujSpThe2H3nh_8nG2p6FQ"

locations='["Boston","NewYork","California","Pennsylvania"]'></g-map>
```

## Paramètres

| clé API | clé api javascript de google |
|---|---|
| **Emplacements** | Liste des emplacements que vous souhaitez marquer sur la carte google |
| ---------- | ---------- |

## Remarques

Pour l'ensemble du code, accédez au référentiel

Pour voir la carte google en action, regardez ici

## Examples

### A - Commencer

En bref, notre élément personnalisé devrait

- Avoir une fonctionnalité de recherche intégrée qui recherche et marque les lieux sur la carte.
- Accepter Un attribut appelé location-array, qui est une liste de lieux
- Avoir un auditeur pour écouter un événement appelé «google-map-ready». Cet événement est déclenché lorsque l'élément est chargé.
    - Cet écouteur doit parcourir le tableau de localisation et assigner l'élément suivant de votre tableau de localisation comme «requête de recherche» actuelle.
- Avoir une méthode appelée cache
    - Cette méthode, met en cache la latitude et la longitude de chaque endroit du tableau de localisation, dès que la recherche renvoie un résultat
- Avoir un modèle de répétition Dom, qui parcourt les résultats mis en cache et dépose un marqueur à chaque emplacement

Ainsi, le but de notre élément personnalisé signifie essentiellement, si vous passez un tableau de sites comme celui-ci:

```
<g-map location-array='["Norway","Sweden"]'></g-map>
```

L'élément personnalisé

- Ne devrait pas seulement marquer la Norvège et la Suède sur la carte, mais,
- Doit également marquer toutes les recherches ultérieures sur la carte - par exemple, si vous recherchez Boston, après que la carte a marqué la Norvège et la Suède, Boston devrait également avoir une épingle sur la carte

## Importer des dépendances

Premièrement, importons tout ce dont nous avons besoin dans notre élément. Cet élément est sous

elements / g-map.html

```
<link rel=import href="../bower_components/google-map/google-map.html">
<link rel=import href="../bower_components/google-map/google-map-marker.html">
<link rel=import href="../bower_components/google-map/google-map-search.html">
<link rel=import href="../bower_components/google-map/google-map-directions.html">
```

### Remarque

Certaines des importations ci-dessus, ne sont pas nécessaires, mais il est bon d'avoir, si vous souhaitez ajouter des fonctionnalités sur vos marqueurs.

Nous avons nécessairement besoin,

- Google Map
- google-map-marker
- google-map-search

En outre, il est présumé, vous savez comment installer les différents éléments polymères via Bower

## Enregistrement de notre élément avec polymère

Ensuite, nous enregistrons notre élément personnalisé comme “g-map”

```
Polymer({
       is:"g-map"
});
```

Alors! notre élément personnalisé est enregistré avec Polymer.

Nous ajouterons plus tard des propriétés spécifiques à Google Map Search et toutes les autres propriétés dont nous avons besoin.

## Ajouter un Blueprint à notre élément personnalisé - via des templates

Ensuite, nous ajoutons le modèle pour notre élément personnalisé

Ouvrez elements / g-map.html et ajoutez un template qui lie les éléments personnalisés DOM

```
<template id="google-map-with-search">
</template>
```

Maintenant, enveloppez l'intégralité du HTML et du javascript que vous avez écrits pour notre élément personnalisé, à l'intérieur d'un module dom.

```
<dom-module id="g-map">

    <template id="google-map-with-search">

    </template>
    <script>
    Polymer({
        is:"g-map",
        properties: {}
 });
     </script>
</dom-module>
```

Très agréable! Nous avons un plan de base

### B - Rendu du google map de base

## Préparez votre navigateur - Polyfill it

Créez une nouvelle page de destination pour notre élément, à index.html.

Incluez des polyfills pour qu'un élément personnalisé fonctionne. Importez également l'élément g-map personnalisé, que nous avons enregistré avec Polymer.

### Remarque:

- Les composants Web sont le polyfill nécessaire pour la prise en charge du navigateur.
- Polymer, est inclus, afin que nous puissions utiliser la bibliothèque pour créer notre élément personnalisé

Ouvrez donc index.html et incluez les polyfills et les polymères nécessaires. En outre, **importez**

l'élément personnalisé que nous avons enregistré

```
<head>
    <title>Google Map</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
     <script src="bower_components/webcomponentsjs/webcomponents.min.js"></script>
     <link rel="stylesheet" href="bower_components/bootstrap/dist/css/bootstrap.min.css">
     <link rel="import" href="bower_components/polymer/polymer.html">
     <link rel="import" href="elements/g-map.html">

</head>
```

Donc, avec le navigateur Polyfilled,

Invoquons l'élément polymère google-map dans notre élément personnalisé.

```
<template id="google-map-with-search">

    <google-map></google-map>

</template>
```

et ensuite, appelez l'élément personnalisé dans index.html, notre page de destination.

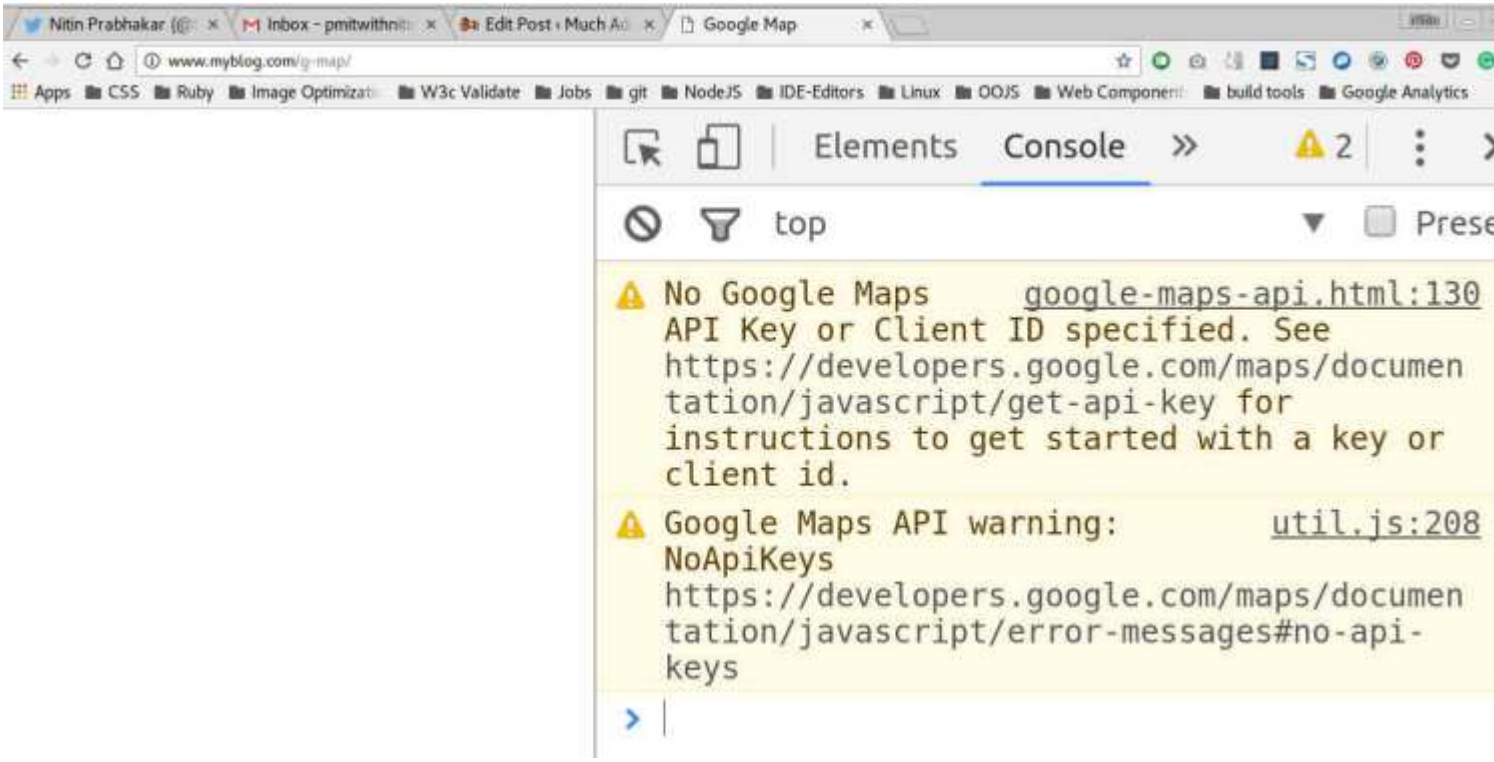
Donc, cela va dans l'index.html

```
<body>

    <div class="container">

        <div class="row">

            <div class="col-xs-12">
                <g-map></g-map>
            </div>

        </div>

    </div>

</body>
```

J'ai essayé de rendre ma page avec l'élément personnalisé g-map à ce stade et j'ai trouvé une page EMPTY!

POURQUOI??

Quand je vois le journal de la console, je le vois




Ah!

Donc, nous avons besoin d'une clé api pour rendre une carte à partir de Google.

## Remarque:

Obtenir une clé API de l'API javascript de Google est assez simple.

Suivez simplement le lien ci-dessous pour générer votre clé personnelle par projet.

Clé API Google

Maintenant, puisque la clé API doit être utilisée, nous avons deux choix.

- Laissez l'utilisateur le transmettre en tant qu'attribut
- Avoir une clé API par défaut configurée

Maintenant, j'irais pour le premier, simplement parce que toute clé API a ses limites imposées. Par exemple, une clé configurée et livrée avec l'élément que nous développons pourrait bientôt être épuisée.

Quiconque veut utiliser l'élément personnalisé peut générer une nouvelle clé et la transmettre en tant qu'attribut.

ainsi:

```
<g-map api-key="AIzaSyBLc_T17p91u6ujSpThe2H3nh_8nG2p6FQ"></g-map>
```

Nous passons donc la clé api en tant qu'attribut et dans notre élément personnalisé, nous

l'associons à l'attribut api-key de l'élément Polymer.

ainsi:

```
<google-map api-key=[[apiKey]]></google-map>
```

Donc, une fois que nous avons passé la clé API, et que nous l’actualisons, nous ne voyons toujours pas la carte!

## Remarque:

Google Maps, besoin de CSS pour rendre. Nous avons besoin de la hauteur de la carte définie explicitement.

Donc, faites un fichier css dans le répertoire css, dites app.css et ajoutez ces lignes

```
google-map{

    min-height:30vmax;
}
```

Et maintenant, après rafraîchissement, nous voyons cela


Yay! Nous venons de rendre une carte avec juste un balisage minimal! Nous venons d'écrire cela!

```
<google-map api-key="[[apiKey]]"></google-map>
```

### C - Ajout d'une capacité de recherche à notre élément personnalisé

Pour rechercher un lieu, nous utilisons l'élément puissant que Polymer fournit, appelé google-map-search.

Tout ce que vous devez faire, c'est passer

- un objet carte et
- une chaîne de requête à l'élément comme ceci:

```
<google-map-search map=[[map]] query=[[query]]></google-map-search>
```

Comment passons-nous l'objet carte? D'où venons-nous cela?

Simple!

Lorsque vous appelez l'élément google-map, liez la mappe de propriétés de votre élément personnalisé à la carte de propriétés de l'élément.

Donc, nous invoquons l'élément google-map comme ceci:

```
<google-map api-key="whatever key you generated for google's javascript API"
map={{map}}></google-map>
```

## Remarque:

Nous faisons une liaison de données bidirectionnelle pour la carte ci-dessus, lorsque vous appelez le polymère Google-map.

Il est représenté par les accolades {{}} au lieu d'une liaison unidirectionnelle indiquée par des accolades carrées [[]].

Lorsque nous faisons une liaison bidirectionnelle,

- Lorsque la propriété de carte de notre élément personnalisé g-map change, l'élément google-map est notifié

et,

- Chaque fois que la propriété de carte de google-map change, nous sommes avertis dans notre élément personnalisé g-map.

Ainsi, avec la liaison de données bidirectionnelle en place, chaque fois que l'objet de carte change dans google-map, notre élément personnalisé est mis à jour avec les modifications.

et nous transmettons l'objet de carte en tant qu'attribut à l'élément google-map-search, de sorte que tous les résultats de recherche soient marqués dans la carte en cours de rendu.

Comment le DOM de notre élément personnalisé regarde-t-il ce moment?

```
<template id="google-map-custom-element">

<google-map-search map=[[map]] query=[[query]]></google-map-search>
<google-map api-key=[[apiKey]] map={{map}}></google-map>
</template>
```

Nous n'avons jamais ajouté les propriétés de la carte et de la requête pour notre élément

personnalisé!

Ajouter les propriétés dans l'appel d'enregistrement à Polymer

```
Polymer({

    is:"g-map",

    properties:{

        map:{

            type: Object

        },

        query:{

            type:String,

            value:""

        }
    }

});
```

Maintenant que nous avons enregistré les propriétés, nous notons que nous

- Obtient la carte actuellement rendue en tant qu'objet - via la liaison de données et son stockage en tant que propriété locale également appelée carte,
- Avoir une requête de propriété, passée à google-map-search

# Accepter une entrée utilisateur en tant que requête de recherche

Mais qui nous donne la requête à rechercher? Se tromper! maintenant? PERSONNE.

C'est précisément ce que nous faisons ensuite.

Ajoutons un formulaire de recherche que l’utilisateur qui rend la carte utilisera pour saisir une requête. Nous avons besoin d'entrées pour le champ de recherche, et nous utilisons les données de fer de Polymer.

# Ajouter un formulaire de recherche

Inclure les éléments de fer nécessaires au début du fichier g-map.html

```
<link rel=import href=../bower_components/iron-input/iron-input.html>
<link rel=import href=../bower_components/iron-icon/iron-icon.html>
<link rel=import href=../bower_components/iron-icons/iron-icons.html>
<link rel=import href=../bower_components/iron-icons/maps-icons.html>
```

Nous voulons le formulaire de recherche, dans le coin supérieur gauche, nous ajoutons donc une entrée de fer avec une position absolue, et un peu de css pour l'épingler.

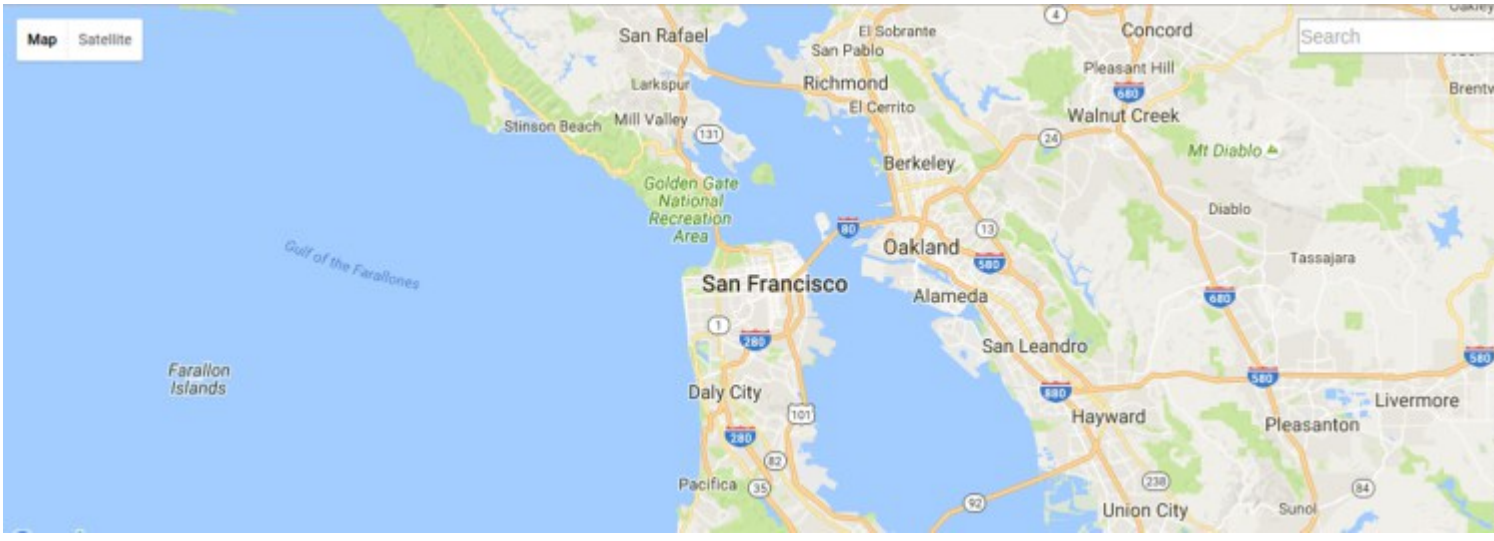
Donc, enveloppez le DOM de notre élément personnalisé, dans un div appelé «map-container», et ajoutez un div enfant appelé «search_form».

```
<template id="google-map-custom-element">
    <div class="map-container">
         <google-map-search map="[[map]]" query="[[query]]"></google-map-search>
         <google-map api-key="[[apiKey]]"></google-map>
         <div class="search_form">
             <iron-icon icon="icons:search" on-tap=__search></iron-icon>
             <input is="iron-input" placeholder="Search Places" type=text name="search" bind-
value="{{query}}">
         </div>
     </div>
 </template>
```

Ajouter un css de base pour épingler le formulaire de recherche que nous avons écrit en haut à droite de la carte

```
.search_form{
    position: absolute;
    top:10px;
    right:10px;
}
.search_form iron-icon{
    position: absolute;
    right: 1px;
    top:1px;
    cursor: pointer;
}
```

Et puis, quand on rafraîchit, on a ça. Le formulaire de recherche en haut à droite




Il ne nous reste plus qu'à fournir une contribution à l'élément google-map-search, via la propriété query de notre élément personnalisé.

Ainsi, nous lions notre «requête» de propriété à l'entrée de fer dans le formulaire de recherche.

ainsi:

```
<input is="iron-input" placeholder="Search" type=text name="search" bind-value="{{query}}">
```

## Remarque:

Nous lions la requête de propriété de notre élément personnalisé à l'entrée de recherche en utilisant,

Attribut «bind-value» d'une entrée de fer.

Donc, bind-value = {{query}} dans l'entrée de fer ci-dessus, s'assure que toute modification du texte à l'entrée de fer est notifiée à la propriété query.

Avec notre configuration actuelle,

Pour chaque alphabet saisi par un utilisateur dans la zone de recherche, une recherche est effectuée, ce qui ralentit la carte et consomme la limite de notre clé API.

Pourquoi?

Parce que la requête de propriété de notre élément personnalisé est liée à l'entrée de fer et que chaque lettre saisie dans l'entrée de fer change la valeur de «requête», ce qui affecte alors le code HTML dans

```
<google-map-search map="[[map]]" query="[[query]]">
```

provoquant une recherche

Notre solution est simple! Liez simplement l'attribut "query" de l'élément google-map-search à une propriété distincte.

Disons que nous créons une propriété «searchQuery» pour notre élément personnalisé.

Ajoutez ceci, sous la propriété "query" dans l'appel Polymer

```
searchQuery :{

    type:String

}
```

Ensuite, modifiez le lien dans l'élément élément google-map-search dans le DOM de notre élément personnalisé

ainsi:

```
<google-map-search map="[[map]]" query="[[searchQuery]]">
```

Enfin, notez que nous avons ajouté une icône de fer (icône de recherche) à notre formulaire de recherche?

```
<iron-icon icon="icons:search" on-tap=search></iron-icon>
```

Ajoutons cette fonction de recherche à notre élément personnalisé. Il est appelé en appuyant sur l'icône de recherche.

Dans notre méthode de recherche, nous affectons tout ce qui existe dans le champ de recherche à l'attribut «query» de google-map-search!

Ajoutez cette fonction, après l'objet «properties» dans l'appel du polymère pour l'enregistrement.

```
//paste this after, the properties object

search: function() {
this.query = this.searchQuery;
}
```

Très agréable! Donc, maintenant, une recherche se produit uniquement en appuyant sur l'icône de recherche.

## Afficher les résultats de la recherche

Nous avons

- Rendu une carte
- Épinglé un formulaire de recherche sur la carte
- A écrit la fonctionnalité pour rechercher par pression sur l'icône de recherche

Maintenant, nous devons afficher les résultats.

Pour cela, nous devons associer les résultats de l'élément google-map-search à une propriété locale.

Alors, laissez-nous créer une propriété pour notre élément personnalisé g-map, et nommez-le résultat duh!

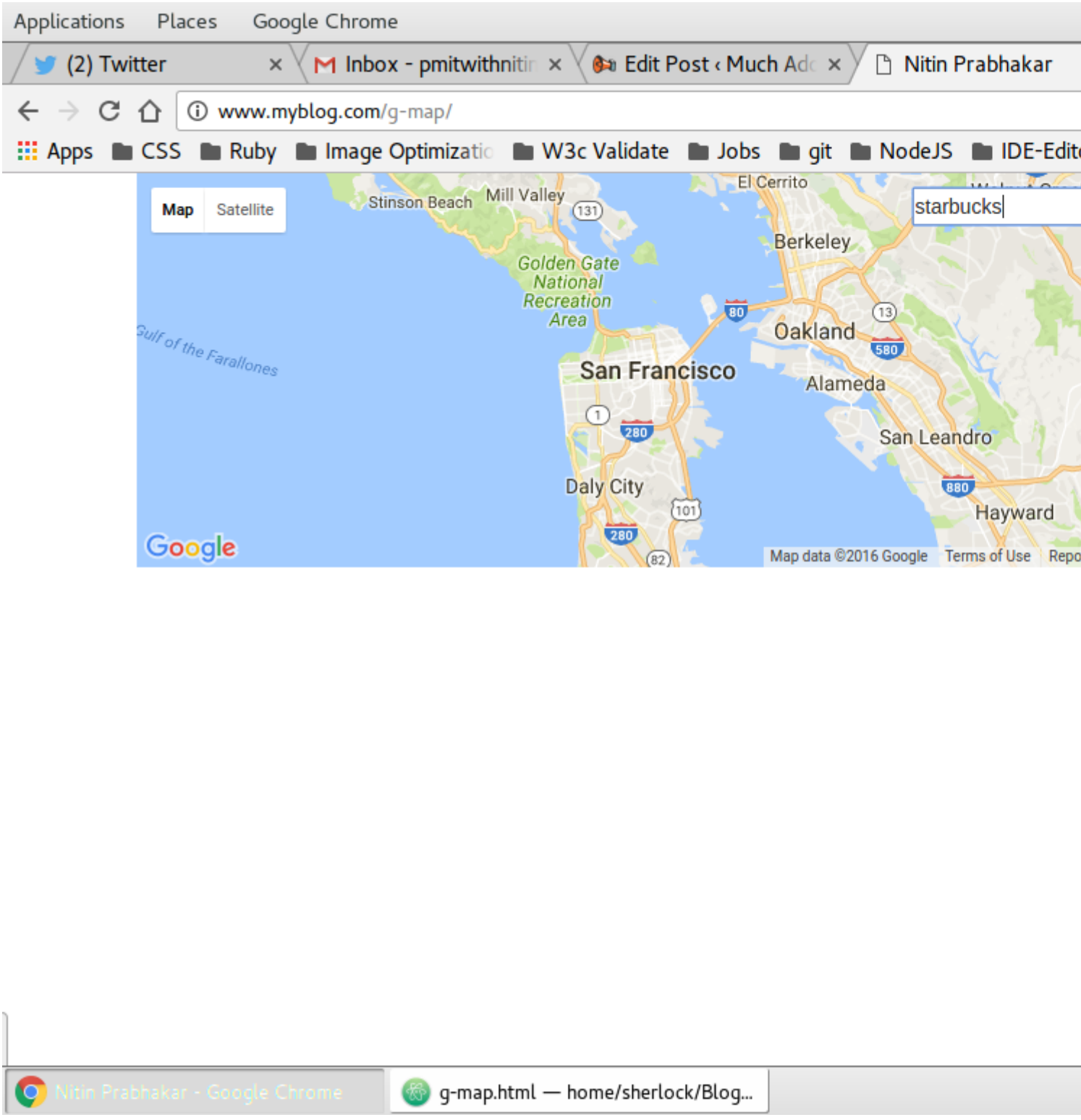
Dans l'appel d'enregistrement Polymer, ajoutez les résultats de propriété de type object dans la déclaration de propriétés

```
results:{

type:Object

}
```

Et liez la propriété results de google-map-search à la propriété de résultats locale que vous avez définie ci-dessus.

```
<google-map-search results={{results}} map=[[map]] query=[[query]]></google-map-search>
```

Avec cela en place, laissez-nous rafraichir la page et rechercher des **étoiles** sur la carte.




Rien ne s'est passé!

Bien Duh! nous avons simplement lié les résultats de google-map-search, à la propriété results. Avons-nous écrit quelque chose pour les afficher sur la carte? NON!

Nous devons le faire.

Nous savons donc que l'élément Polymer google-map-search renvoie un tableau de marqueurs en tant que résultats.

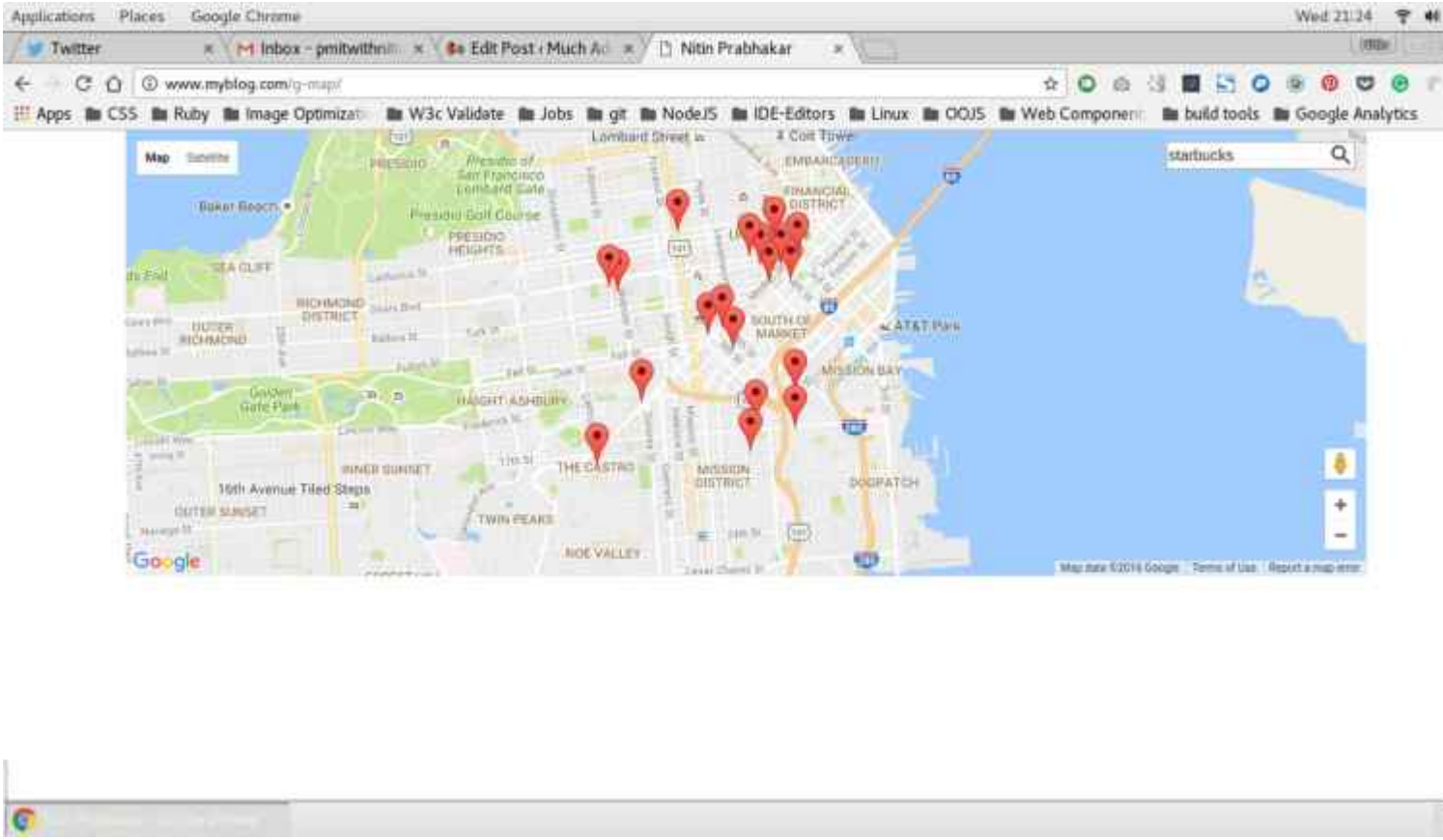
Nous devons parcourir le tableau et placer des marqueurs sur la carte.

Donc, nous écrivons un modèle de répétition de dom, dans notre élément, pour afficher les marqueurs pour chaque résultat

ainsi:

```
<google-map api-key=[[apiKey]] map=[[map]]>

    <template is="dom-repeat" items="{{results}}" as="marker">
        <google-map-marker latitude="{{marker.latitude}}" longitude="{{marker.longitude}}">
            <h2>{{marker.name}}</h2>
            <span>{{marker.formatted_address}}</span>
        </google-map-marker>
    </template>

</google-map>
```

Et maintenant sur rafraîchir, nous voyons cela




Très agréable!

Nous avons ajouté avec succès la fonctionnalité de recherche à notre élément personnalisé.

Tout ce que nous devons faire est,

écris ceci dans notre index.html

```
<g-map api-key="Whatever API key you generated"></g-map>
```

Et vous obtenez une carte sur laquelle vous pouvez rechercher!

## D - Résultats de la recherche en cache

Notre objectif était cependant de

- faire une liste de lieux
- tous les marquer sur la carte et,
- marquez tout autre lieu de notre choix en utilisant le champ de recherche

Nous avons un champ de recherche, où l'utilisateur peut rechercher une requête à la fois. Afin de gérer plusieurs requêtes, nous devons mettre en cache les résultats pour chaque requête avant d'afficher les marqueurs et de les épingler sur la carte.

Alors ajoutons

- Une propriété dire des emplacements qui accepte un tableau d'emplacements
- Un cache de propriétés qui stocke les résultats de chaque requête dans le tableau d'emplacement transmis à notre élément personnalisé

```
cache:{

type:Object

},

locations:{

type:Object

}
```

Ensuite, nous écrivons une fonction pour mettre en cache chaque résultat de recherche, résultant d'une recherche pour chaque emplacement du tableau de localisation.

C'est assez simple!

Nous rassemblons déjà les résultats dans une propriété appelée «résultats», que nous parcourons ensuite pour afficher les marqueurs.

Tout ce que nous devons maintenant faire, c'est pousser chaque résultat de recherche dans un tableau, et notre cache est terminé!

Nous devons utiliser un événement pour déclencher la mise en cache. Cet événement, appelé «on-google-map-search-result», est déclenché après que l'élément google-map-search a terminé une recherche.

Donc, «on-google-map-search-result», nous appelons une fonction dire «cacher», pour mettre en cache les résultats de la recherche actuelle dans une propriété appelée cache.

```
<google-map-search map=[[map]] query=[[searchQuery]] on-google-map-search-results=cacher>
```

## Remarque:

Normalement, pour pousser un élément dans un tableau, nous utilisons la méthode push vanilla

Par exemple: J'ai un nouvel endroit à ajouter à ma liste de lieux que je veux marquer, disons NewYork.

Si le tableau auquel je veux l'ajouter s'appelle des emplacements, j'écrirais

```
locations.push('New York');
```

Cependant, dans notre cas, nous devons pousser n'importe quel résultat de Google-map-search dans un tableau appelé cache. Ensuite, nous devons parcourir le cache avec les derniers résultats inclus. Pas le cache existant.

Maintenant, nous utilisons le modèle de répétition dom pour parcourir les résultats et placer des marqueurs sur la carte.

Cependant, si nous passons un tableau de localisation comme ceci:

```
<g-map locations=["Iceland","Argentina","London"]></g-map>
```

Ensuite, la propriété results de notre élément personnalisé sera remplacée pour chaque élément de ce tableau.

Par conséquent, nous appelons la méthode cacher sur l'évènement google-map-search-results et poussons le résultat actuel dans la propriété cache.

Cela ne peut pas être une poussée de vanille comme dans la note ci-dessus.

Nous devons laisser savoir au répéteur que notre cache a été écrasé à chaque nouvelle recherche.

Donc, nous utilisons une poussée spéciale que Polymer expédie, comme ceci:

```
cacher: function() {
this.push('cache', this.results);
}
```

Ajoutez cette fonction, après la fonction de recherche que nous avons écrite précédemment.

La syntaxe implique que le cache de propriété doit être muté, en ajoutant la valeur actuelle des

résultats de la propriété et que tous les répéteurs domotiques utilisant la propriété du cache pour effectuer une boucle doivent être notifiés de la mutation.

Enfin, nous modifions le modèle dom-repeat pour qu'il passe en boucle dans le cache, au lieu des résultats.

## Remarque:

Le cache est un tableau de tableaux.

chaque élément du tableau cache est un tableau de résultats

Donc, notre répéteur sera comme ça:

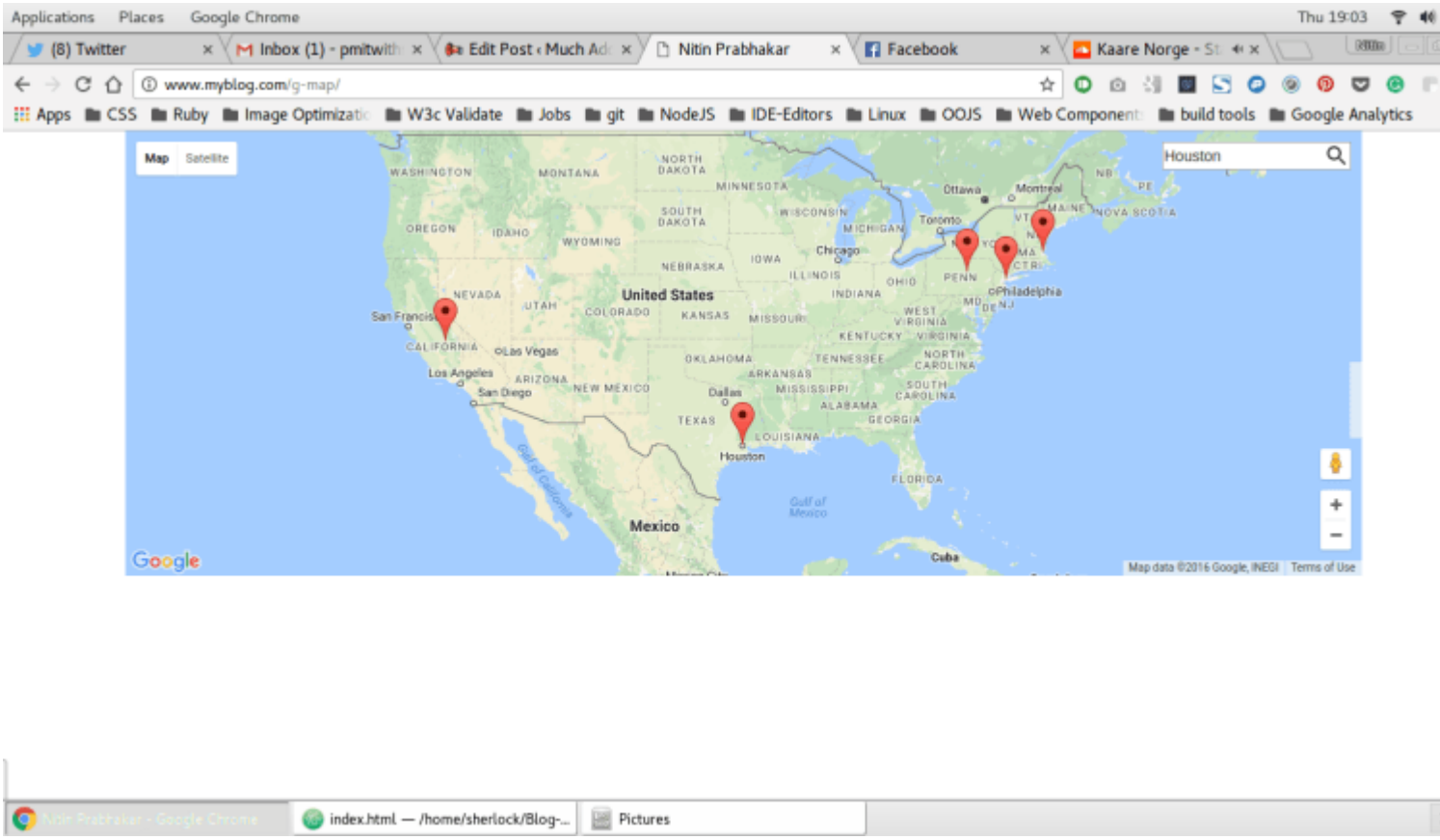
```
<template id="resultList" is="dom-repeat" items="[[cache]]">

    <template is="dom-repeat" items="[[item]]" as="marker">

        <google-map-marker latitude="{{marker.latitude}}" longitude="{{marker.longitude}}">
            <h2>{{marker.name}}</h2>
            <span>{{marker.formatted_address}}</span>
        </google-map-marker>
    </template>
</template>
```

Très agréable! Nous avons l'élément personnalisé tout codé et prêt à être utilisé!

Laissez-nous le tester en passant un tableau d'emplacements à partir de index.html doit-on?

```
<g-map

api-key="AIzaSyBLc_T17p91u6ujSpThe2H3nh_8nG2p6FQ"

locations='["Boston","NewYork","California","Pennsylvania"]'>
</g-map>
```

et c'est parti!




## Le Polymère Google Map personnalisé complet avec cache intégré

```
<link rel=import href="../bower_components/google-map/google-map.html">
<link rel=import href="../bower_components/google-map/google-map-marker.html">
<link rel=import href="../bower_components/google-map/google-map-search.html">
<link rel=import href="../bower_components/google-map/google-map-directions.html">
<link rel=import href="../bower_components/iron-input/iron-input.html">
<link rel=import href="../bower_components/iron-icon/iron-icon.html">
<link rel=import href="../bower_components/iron-icons/iron-icons.html">
<link rel=import href="../bower_components/iron-icons/maps-icons.html">
<dom-module id="g-map">
    <template id="google-map">
        <div class="map-container">
            <google-map-search map="[[map]]" query="[[query]]" results="{{results}}" on-
google-map-search-results=cacher>
            </google-map-search>
            <google-map api-key="[[apiKey]]" map="{{map}}" on-google-map-ready="_onMapResolve"
fit-to-markers disable-zoom single-info-window>
                <template id="resultList" is="dom-repeat" items="[[cache]]">

                    <template is="dom-repeat" items="[[item]]" as="marker">

                        <google-map-marker latitude="{{marker.latitude}}"
longitude="{{marker.longitude}}">
                            <h2>{{marker.name}}</h2>
                            <span>{{marker.formatted_address}}</span>
                        </google-map-marker>
                    </template>
                </template>
            </google-map>
            <div class="search_form">
                <p>
```

```
                    <iron-icon icon=icons:search on-tap=search></iron-icon>
                    <input is=iron-input placeholder="Search" type=text name=start bind-
value={{searchQuery}}>
                </p>
            </div>
        </div>
    </template>
    <script>
        Polymer({
            is: "g-map",
            properties: {
                apiKey: {
                    type: String,
                    value: "AIzaSyBLc_T17p91u6ujSpThe2H3nh_8nG2p6FQ"
                },
                map: {
                    type: Object
                },
                query: {
                    type: String,
                    value: ""
                },
                locations: {
                    type: Array,
                    value: []
                },
                cache: {
                    type: Array,
                    value: []
                },
                results: {
                    type: Array,
                    value: function() {
                            return [];
                        }
                    }
            },
            _onMapResolve: function() {
                var search = document.querySelector("google-map-search");
                this.locations.forEach(function(location) {
                    search.query = location;
                })
            },
            search: function() {
                this.query = this.searchQuery;
            },
            cacher: function() {
                this.push('cache', this.results);
            }
        })
    </script>
</dom-module>
```

Lire Google Map Mark avec cache intégré en ligne:
https://riptutorial.com/fr/polymer/topic/7487/google-map-mark-avec-cache-integre

# Chapitre 8: Le débogage

## Examples

### Désactiver la mise en cache de l'importation HTML

La mise en cache de l'importation HTML signifie parfois que les modifications apportées aux fichiers HTML importés ne sont pas répercutées lors de l'actualisation du navigateur. Prenez l'importation suivante comme exemple:

```
<link rel="import" href="./my-element.html">
```

Si une modification est apportée à `my-element.html` après avoir préalablement chargé la page, le fichier modifié peut ne pas être téléchargé et utilisé dans le document en cours lorsqu'il est actualisé (car il a été précédemment importé et mis en cache). Cela peut être bon pour une production, mais peut nuire au développement.

Pour désactiver cela dans Google Chrome:

- Ouvrez les DevTools de Google Chrome
- Sélectionnez le menu principal > Paramètres
- Aller à la section Réseau
- Sélectionnez "Désactiver le cache (pendant que DevTools est ouvert)"

Cela évitera la mise en cache des importations HTML, mais uniquement lorsque DevTools est ouvert.

Lire Le débogage en ligne: https://riptutorial.com/fr/polymer/topic/5864/le-debogage

# Chapitre 9: Modaux réutilisables avec polymère

## Syntaxe

- Déclaration d'élément: <tool-bar link2-share = "" link2-fork = "" modal-id = "" title = ""> </ tool-bar>

## Remarques

Remarque:

Le code à travers cet écrit n'est pas une copie de travail. Vous devez remplacer les remplisseurs pour les noms de hrefs, src et de projet.

Le code illustre uniquement une preuve de concept.

Pour voir l'élément personnalisé en action,

va ici

Et pour parcourir l'utilisation et la structure de l'élément personnalisé,

va ici

L' ***utilisation*** est dans “index.html”

L' ***élément*** est dans «elements / tool-bar.html»

## Examples

### Modals

Donc, vous souhaitez ajouter de la conception matérielle à votre portefeuille d’entreprise ou de carrière. Vous ne pouvez pas résister à l'utilisation d'un modal? qui clique sur un clic sur les cartes que vous avez l’intention de concevoir pour chacun de vos projets / produits!

Disons, pour chaque projet, vous avez fait, ou pour chaque produit que vous avez conçu, vous avez besoin d'une carte de matériel.Chaque carte devrait avoir

- Une icône qui affiche une boîte d’information qui répertorie toutes les fonctionnalités de
- votre produit ou projet, en détail. Une icône «partager ceci», qui partage
- votre produit ou projet sur les réseaux sociaux Icône représentant une fourchette, qui ouvre la page github de votre produit / projet

Avec html natif, et en supposant que vous utilisiez un bootstrap ou une autre disposition de type Flex Box, vous devrez

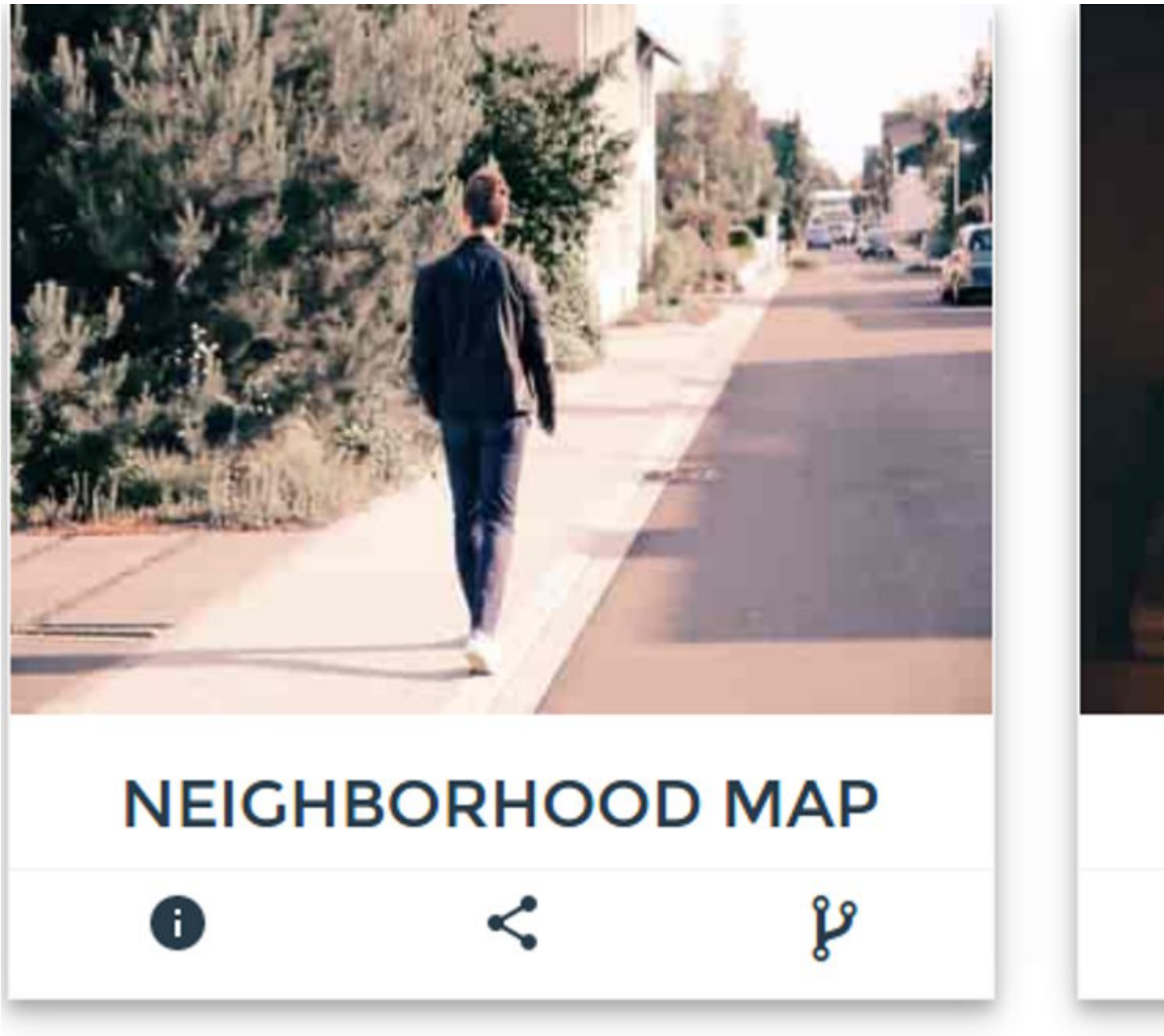
- Ecrire un div wrapper pour chaque nouveau projet avec une ligne de classe
- Enveloppez votre image de héros pour chaque projet dans une colonne de 12 cols de large
- Ecrire une autre ligne
- Enroulez chaque lien (info | share | fork) dans une colonne de 4 colonnes de large
- Injecter ces deux lignes dans un conteneur

Pour commencer. Une fois que vous avez fait ce qui précède pour tous vos projets, vous devez travailler sur la création de la fenêtre contextuelle requise pour chaque projet.

- Vous téléchargez des tonnes de javascript et de css (en personnalisant le bootstrap pour la fonctionnalité modale)
- Vous écrivez un modal avec un identifiant unique, pour différencier les informations de chaque projet (modal-P1 pour le projet 1, modal-P2 pour le projet 2 et ainsi de suite)
- Vous composez ensuite le fichier HTML requis, pour afficher les informations de chaque projet, dans son modal correspondant.

## Quels sont nos objectifs?

Juste pour répéter, visuellement, si vous avez deux projets dans votre portefeuille, le but est d'avoir des cartes comme ça:




Donc, nous avons deux préoccupations pour chaque projet

- L'image du héros
- La barre d'outils avec info (affiche un modal), des liens de partage et de fourche

Tout d'abord, laissez-nous nous attaquer à ce qui se produit lorsqu'un utilisateur clique sur le lien info dans la barre située sous chaque projet. Nous avons besoin d'un modal pour afficher plus d'informations sur le projet.

Alors, comment configurez-vous exactement un modal avec bootstrap?

- Vous écrivez ceci pour le déclencher, pour chaque projet

```
<button type="button" class="btn btn-info btn-lg" data-toggle="modal" data-
target="#myProject1"><img src="path-to-info-icon"></img></button>
```

- Vous écrivez ensuite le corps du modal comme suit:

```
<div id="myProject1" class="modal fade" role="dialog">
     <div class="modal-dialog">
         <div class="modal-content">
             <div class="modal-header">
                 <button type="button" class="close" data-dismiss="modal"></button>
                 <h4 class="modal-title">Project Title</h4>
             </div>
             <div class="modal-body">
                 <p>Some text in the modal.</p>
             </div>
             <div class="modal-footer">
                 <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
             </div>
         </div>
     </div>
</div>
```

et en fonction du contenu et de la longueur de votre description de projet, vous perdez un peu de temps à surveiller l'ouverture et la fermeture d'une soupe au-dessus dans le corps du modal!

De plus, ce balisage est carrément fastidieux pour composer tout en vous nécessitant un modal.

Phew! Et est-ce que ça s'est terminé là? Non, vous devez toujours écrire pour chaque projet,

- L'image de héros de la carte projet, affichée au-dessus de la barre d'outils.

Alors, comment pouvons-nous ajouter l'image de héros pour chaque projet? PLUS html!

Le héros a besoin d'image

- Une rangée à part
- Une colonne de 12 cols de large à l'intérieur de cette rangée

Comment le HTML complet chercherait-il à afficher un projet comme une carte ci-dessus?

```
<article id="project-neighbourhood">

    <div class="row">

        <div class="col-12">

            <img src="path-to-hero-image"></img>

        </div>

    </div>

    <div class="row">

        <div class="col-12">
            <div class="row">
                <div class="col-4">
                    <button type="button" class="btn btn-info btn-lg" data-toggle="modal"
data-target="#myProject1"><img src="path-to-info-icon"></img></button>
                </div>
                <div class="col-4">
```

```
                    <button class="btn btn-lg" id="share-on-g-plus"><img src="path-to-share-
icon"></img></button>
                </div>
                <div class="col-4">
                    <button class="btn btn-lg" id="fork"><img src="path-to-fork-
icon"></img></button>
                </div>
            </div>
        </div>
    </div>
</article>
```

Maintenant, disons que vous avez 10 projets à montrer! Vous devez réécrire ce balisage désordonné 10 fois! Plus 10 différents modaux!

Je n'ai pas la patience de réécrire ce HTML, et 10 Modals, et je sais que vous ne le faites pas aussi!

### Modals, Avec Polymère?

## Remarque

> Je suppose que tu sais
>
> Spécifications des composants Web
>
> Polymère de Google

Et si, on pouvait juste avoir un élément personnalisé, disons

```
<tool-bar></tool-bar>
```

et comme par magie, tout ce que le balisage Modal pouvait faire?

Tentant hein!

Comment spécifiez-vous quel Modal appartient à quel projet?

Simple! Ecrivez

```
<tool-bar modal-id="Project-cats"></tool-bar>
```

Donc, cela réserve le balisage avec un identifiant de «Projet-chats» pour le projet sur les chats par exemple.

Comment écrivez-vous ce qui entre dans le modal? simple! Écrivez simplement votre balisage normal, enveloppé, dans la balise personnalisée “”

Ainsi:

```
<tool-bar modal-id="Project-cats">

<div class="col-12 modal-desc">
```

```
    <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.</p>
</div>

</tool-bar>
```

N'est-ce pas assez simple?

Et si vous vous demandez comment se présente le balisage complet? Y compris la part et les liens de fourche, voir ci-dessous.

```
<div class=row>

    <div class="col-12"> <img src="path-to-hero-image"></img></div>

</div>

<div class="row">

    <tool-bar

    modal-id="Project-cats"

    link2-share="http://www.project-cats.com/kitten"

    link2-fork="https://github.com/myusername/my-project-cats">

        <div class="col-12 modal-desc">

            <p> yadda yadda blah blah</p>

        </div>

    </tool-bar>

</div>
```

Beaucoup mieux que de réécrire la div soupe complète hein!

Bien sûr, vous pouvez le raccourcir davantage et le rendre complètement compilable en faisant abstraction de l'implémentation de l'image du héros à l'intérieur de l'élément personnalisé.

## Barre d'outils réutilisable avec modal | Partager | Icônes de fourche - Codage de l'élément

Jusqu'à présent, nous avons défini la facilité avec laquelle il est possible d'écrire un élément personnalisé qui cache le HTML malpropre qui se trouve derrière et donne à l'utilisateur un balisage rapide et facile à écrire.

Il est temps de le coder!

Notre élément personnalisé, pour afficher la barre sous l'image du héros devrait

- Accepter un lien à partager
- Accepter un lien vers le dépôt pour être fourchu
- Acceptez un modalId pour différencier les autres modaux.
- Accepter un titre pour le modal

## Enregistrer l'élément personnalisé de la barre d'outils avec Modal | Partager | Icônes fourchette

Laissez-nous enregistrer notre élément personnalisé, qui accepte la liste de propriétés ci-dessus.

```
<script>
Polymer({
    is:"tool-bar",
    properties:{
        link2Share:{
            type:String,
            value:""
        },
        link2Fork:{
            type:String,
            value:""
        },
        modalId:{
            type:String,
            value:""
        },
       title:{
            type:String,
            value:"myModal"
        }
    }
});
</script>
```

Très agréable! Ensuite, ajoutez le code HTML dont nous avons besoin.

Tout d'abord, les icônes pour Modal / Share / Fork doivent être ajoutées. Une simple ul fera l'affaire. Nous utiliserons également les icônes de fer de Polymer pour afficher les icônes Modal / Share / Fork.

## Note: Icônes de fer de polymère

> Installez les jeux d'icônes comme dans la racine de votre projet.
>
> ```
> bash $ bower install --save PolymerElements/iron-icon
>
> bash $ bower install --save PolymerElements/iron-icons
> ```

Ensuite, incluez les éléments Polymer installés dans la déclaration d'élément personnalisée, comme ceci

Remarque:

À toutes fins utiles, notre élément personnalisé, sera logé à l'intérieur

***Project-root / elements***

```
<link rel="import" href="../bower_components/iron-icon/iron-icon.html">
<link rel="import" href="../bower_components/iron-icons/iron-icons.html">
<link rel="import" href="../bower_components/iron-icons/social-icons.html">
```

Très agréable! Si vous vous demandez comment utiliser les icônes de fer, consultez la note ci-dessous.

## Remarque: utilisation de l'icône de fer et grammaire

Si nous devons utiliser l'icône info pour notre Popup Modal, nous écrivons le balisage comme ceci:

```
<iron-icon icon="icons:info"></iron-icon>
```

Si nous voulons l'icône de partage de réseau social, alors l'attribut d'icône ci-dessus devient

icon = " **social: partager** "

Donc, la signification grammaticale, j'ai besoin du jeu d'icônes de «SOCIAL» et je veux l'icône SHARE de cet ensemble.

Vous pouvez aussi utiliser font-awesome, auquel cas vous pouvez utiliser,

choisissez ce que vous jugez le mieux adapté.

Une fois les inclusions effectuées, nous écrivons le modèle pour notre élément personnalisé. Nous utiliserons des liants unidirectionnels pour les données dans le balisage de notre élément personnalisé. Nous lions tout ce qui est passé en tant qu'attribut par l'utilisateur, aux propriétés correspondantes de notre élément, tout en entamant l'appel d'enregistrement de notre élément.

Le modèle HTML pour notre élément personnalisé est donc le suivant:

```
<dom-module id="tool-bar">

    <template id="tool-box">

        <ul class="flex-wrap toolbar">
            <li>
                <iron-icon icon="icons:info" id="anchor-for-[[modalId]]"
onclick="clickHandler(event)"></iron-icon>
            </li>
            <li>
                <a href="https://plus.google.com/share?url=[[link2Share]]" target="_blank"
><iron-icon icon="social:share"></iron-icon></a>
            </li>
```

```
            <li>
                <a href="[[link2Fork]]" target="_blank"><i class="fa fa-2x fa-code-fork" aria-
hidden=true></i></a>
            </li>
        </ul>

    </template>

</dom-module>
```

Cool! La barre d'outils a donc son modèle basé sur ul, qui répertorie les trois liens

- icône info - Popup modal
- partager cette icône - partager sur google
- Fork this icon - La fourche au repo

En outre, nous avons lié les attributs que l'utilisateur écrit, aux propriétés respectives.

# Remarque: Rendre l'identifiant modal unique

> Nous avons traité l'unicité de l'icône info en spécifiant l'attribut id comme ça
>
> ```
> id="anchor-for-[[modalId]]" onclick="clickHandler(event)"></iron-icon>
> ```
>
> Donc, chaque projet aura un identifiant d'ancrage unique.
>
> c'est à dire,
>
> Si l'utilisateur passe “project-cats” comme modal-id, dans l'appel à notre élément personnalisé, l'identifiant d'ancrage sera “anchor-for-project-cats”
>
> Si l'utilisateur ajoute un autre projet et passe «project-puppy» en tant qu'attribut modal-id, l'identifiant d'ancrage sera anchor-for-project-puppy
>
> etc.
>
> Mais comment pouvons-nous nous assurer que chaque icône d'information est associée à sa propre modale?
>
> L' **icône** « **fer à repasser** » pour les **informations** doit être associée à un attribut « ***dialogue de données*** », qui doit également être unique.
>
> L'attribut data-dialog est similaire à l'attribut data-target dans le modal bootstrap. Il doit être unique pour chaque nouveau projet.
>
> Nous laissons à l'utilisateur le soin de conserver son caractère unique. Ainsi, l'utilisateur ne peut pas avoir le même attribut modal-id pour différents appels à la `<tool-bar>` .

Comme indiqué dans la note ci-dessus, nous devons associer l'icône d'information de chaque projet à une boîte de dialogue de données.

Ajoutons cette fonctionnalité.

Nous utilisons la méthode ready.It est très similaire à celui de Jquery

```
$.ready();
```

Il est appelé lorsque le chargement de l'élément personnalisé est terminé.

Ajoutez cette fonction, après l'objet de propriétés, dans l'appel de polymère pour l'enregistrement.

```
ready: function() {
    document.querySelector("#anchor-for-" + this.modalId).setAttribute('data-dialog',
this.modalId);
}
```

Très sympa! Alors maintenant nous avons

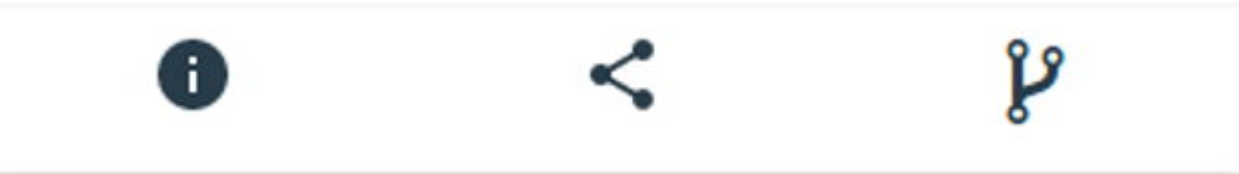
- Un élément personnalisé enregistré en tant qu'élément polymère
- Modal | Partager | Icônes de fourches ajoutées à l'élément personnalisé

Un modal est attaché à l'icône de fer info, avec l'identifiant «anchor-for- [quelque chose est passé comme modal-id]». C'est la courtoisie de la méthode prête de notre élément.

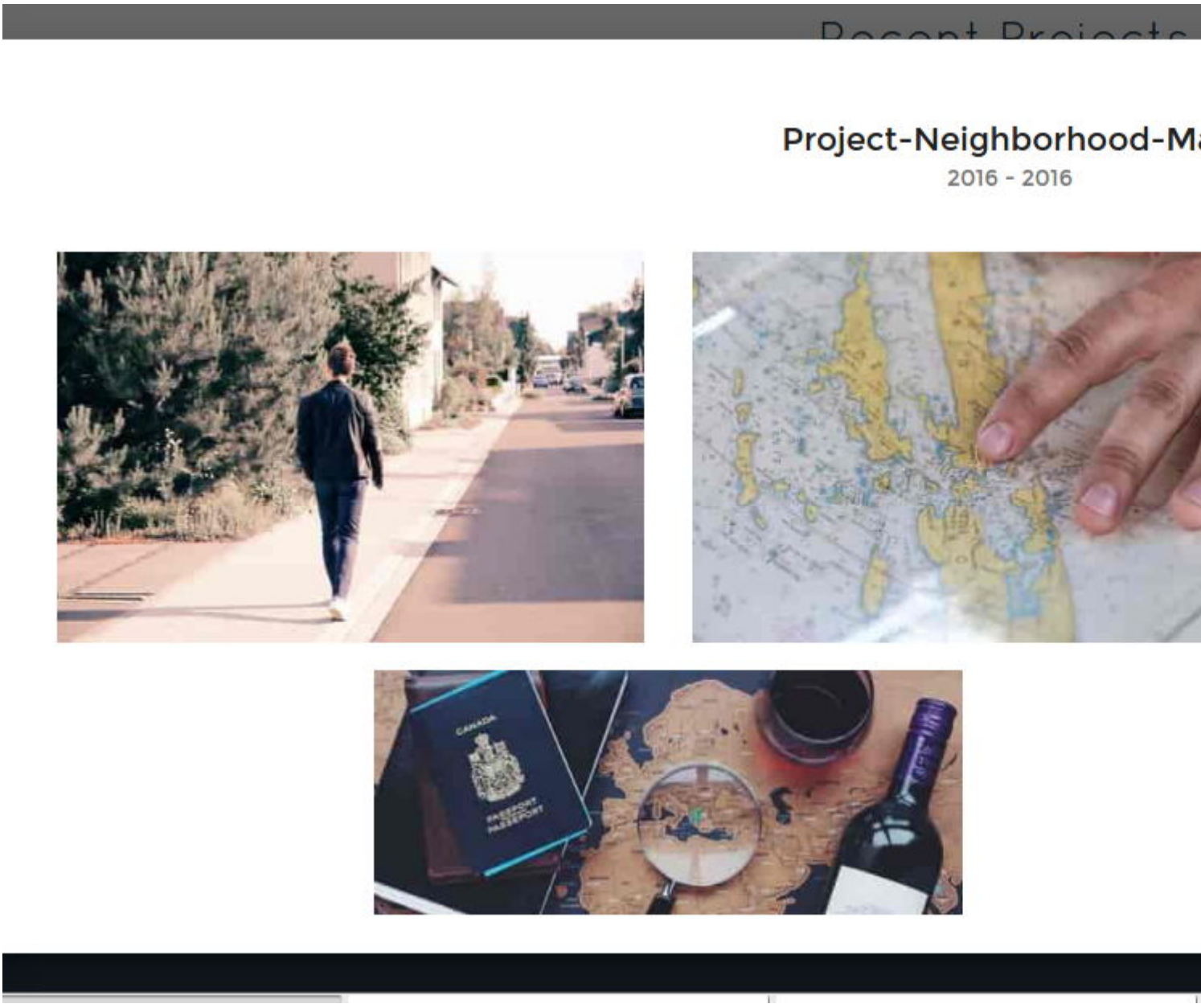
Donc, jusqu'à présent, l'écriture

```
<tool-bar
link2-share="http://www.myportfolio.com/project-neighbourhood-map"
link2-fork="https://github.com/me/project-neighbourhood-map"
modal-id="project-neighbourhood-map">
</tool-bar>
```

Produit ceci:



Mais lorsque l'utilisateur clique sur cette icône d'information, quelque chose comme ceci devrait se produire




et notre objectif était d'emballer cette description complète du projet dans la `<tool-bar>` de la page des utilisateurs.

Se souvenir de Shadow DOM? Nous avons besoin d'un conteneur dans notre élément personnalisé, qui fournit le DOM de l'ombre pour le contenu de notre Modal. Le contenu lui-même est placé dans une `<content>` à l'intérieur de shadowDOM de l'élément personnalisé.

Pour le conteneur qui héberge la description du projet, nous utilisons l'élément polymère « **paper-dialog** ». La boîte de dialogue papier est affectée comme identifiant, tout ce qui est passé comme identifiant modal dans la déclaration sur la page de l'utilisateur.

Donc, ajoutez ce balisage, après le ***ul*** dans le DOM de l'élément personnalisé de la barre d'outils

```
<paper-dialog id="[[modalId]]" modal>
    <div class="text-right modal-close">
        <iron-icon icon="icons:power-settings-new" dialog-dismiss></iron-icon>
    </div>
    <h2 class="text-center text-capitalize">[[title]]</h2>
```

```
    <paper-dialog-scrollable>
        <div class="container-fluid">
            <div class="row flex-wrap info">
                <content></content>
            </div>
        </div>
    </paper-dialog-scrollable>
</paper-dialog>
```

## Remarque: défilement du dialogue papier

> Nous utilisons ***Paper-dialog-scrollable*** , pour envelopper la description de notre projet, de sorte que toutes les descriptions longues ne débordent pas.

Le dernier bit consiste à ajouter la fonctionnalité onclick lorsque le modal doit s'ouvrir, en cliquant sur l'icône info.

Lorsqu'un clic se produit, comme c'est normal, un événement est déclenché, que nous transmettons à notre fonction clickHandler comme suit:

```
<iron-icon icon="icons:info" id="anchor-for-[[modalId]]" onclick="clickHandler(event)"></iron-icon>
```

Donc, une fois que la fonction clickHandler reçoit l'événement,

- Il faut rassembler, quel élément a déclenché le clic,
- Il faut rassembler l'attribut data-dialog de cet élément pour déterminer quel modal ouvrir (Rappelez-vous que chaque projet a son propre identifiant modal?)
- Il faut ensuite appeler la méthode ouverte, exactement pour ce modal

Donc, la fonction est comme ça:

Ajoutez ceci, avant l'appel d'enregistrement de l'élément. Cela ne fait pas partie de l'élément, mais est orienté dans l'élément.

```
function clickHandler(e) {
    var button = e.target;
    while (!button.hasAttribute('data-dialog') && button !== document.body) {
        button = button.parentElement;
    }

    if (!button.hasAttribute('data-dialog')) {
        return;
    }

    var id = button.getAttribute('data-dialog');
    var dialog = document.getElementById(id);
    if (dialog) {
        dialog.open();
    }
}
```

Donc, avec cela, nous avons terminé le codage de l'élément personnalisé, `<tool-bar>` .

## L'image complète Élément personnalisé

```
<link rel="import" href="../bower_components/iron-icon/iron-icon.html">
<link rel="import" href="../bower_components/iron-icons/iron-icons.html">
<link rel="import" href="../bower_components/iron-icons/social-icons.html">
<link rel="import" href="../bower_components/paper-dialog/paper-dialog.html">
<link rel="import" href="../bower_components/paper-button/paper-button.html">
<link rel="import" href="../bower_components/paper-dialog-scrollable/paper-dialog-
scrollable.html">
<dom-module id="tool-bar">
        <template id="tool-set">
            <ul class="flex-wrap toolbar">
                <li>
                    <iron-icon icon="icons:info" id="anchor-for-[[modalId]]"
onclick="clickHandler(event)"></iron-icon>
                </li>
                <li>
                    <a href="https://plus.google.com/share?url=[[link2Share]]"
target="_blank">
                        <iron-icon icon="social:share"></iron-icon>
                    </a>
                </li>
                <li><a href="[[link2Fork]]" target="_blank"><i class="fa fa-2x fa-code-fork"
aria-hidden=true></i></a>
                </li>
            </ul>
            <paper-dialog id="[[modalId]]" modal>
                <div class="text-right modal-close">
                    <iron-icon icon="icons:power-settings-new" dialog-dismiss>
</iron-icon>
                </div>
                <h2 class="text-center text-capitalize">[[title]]</h2>
                <paper-dialog-scrollable>
                    <div class="container-fluid">
                        <div class="row flex-wrap info">
                            <content></content>
                        </div>
                    </div>
            </paper-dialog-scrollable>
        </paper-dialog>
    </template>

    <script>
            function clickHandler(e)
            {
                 var button = e.target;
                 while (!button.hasAttribute('data-dialog') && button !== document.body) {
                    button = button.parentElement;
                 }

                 if (!button.hasAttribute('data-dialog'))
                 {
                   return;
                 }

                var id = button.getAttribute('data-dialog');
                var dialog = document.getElementById(id);
                if (dialog) {
                  dialog.open();
               }
```

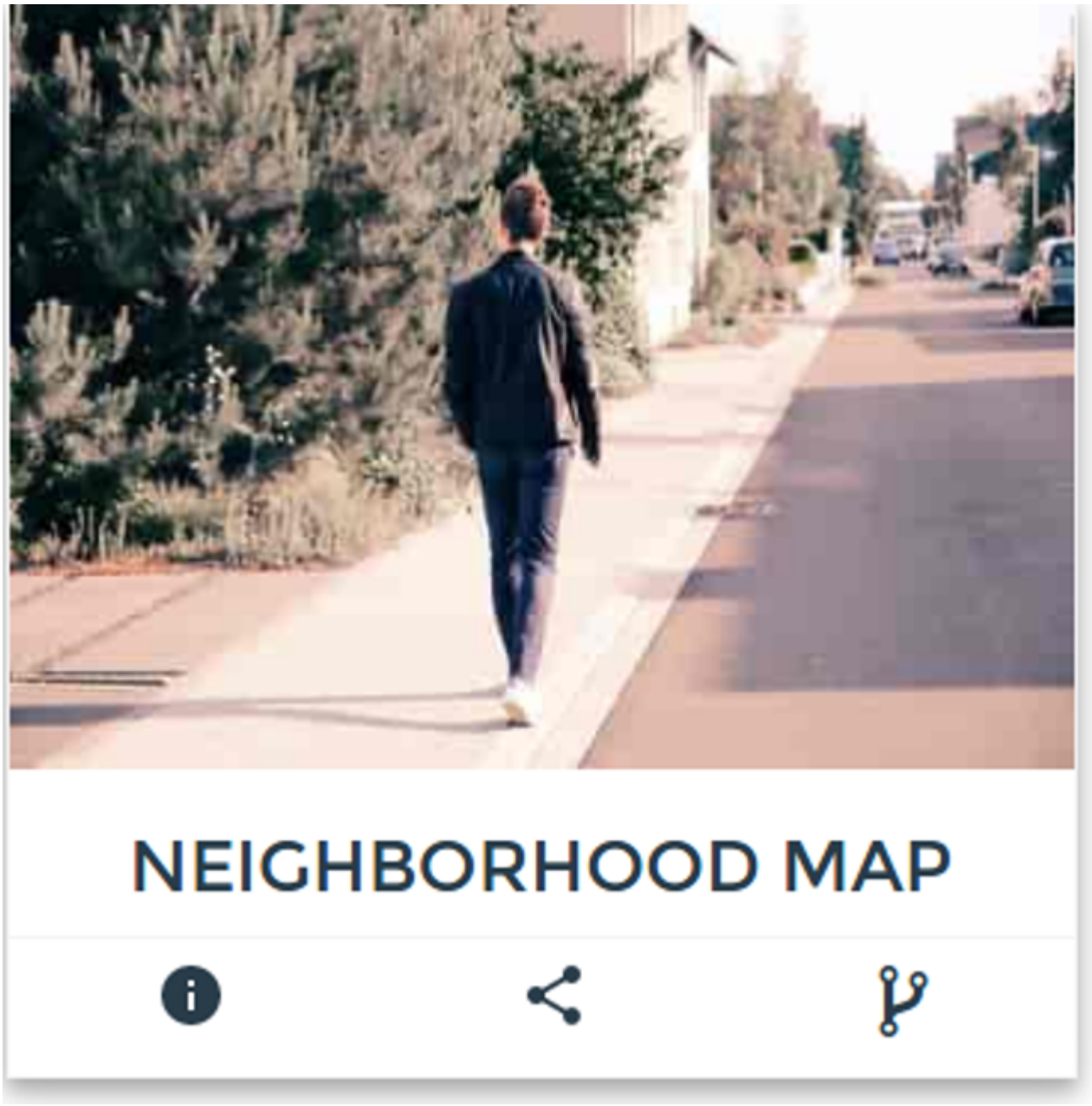
```
          }
      Polymer({
             is: "tool-bar",
            properties: {
                  link2Fork:{
                      type:String,
                      value:""
                   },
                 link2Share: {
                       type: String,
                       value: ""
                   },
                 title: {
                       type: String,
                       value: null
                 },
                modalId:{
                       type:String,
                       value:""
               }
             },
            ready: function() {
                  document.querySelector("#anchor-for-" + this.modalId).setAttribute('data-
dialog', this.modalId);
            }
         });
    </script>
</dom-module>
```

## Utiliser l'élément personnalisé avec modal | Partager | Icônes fourchette

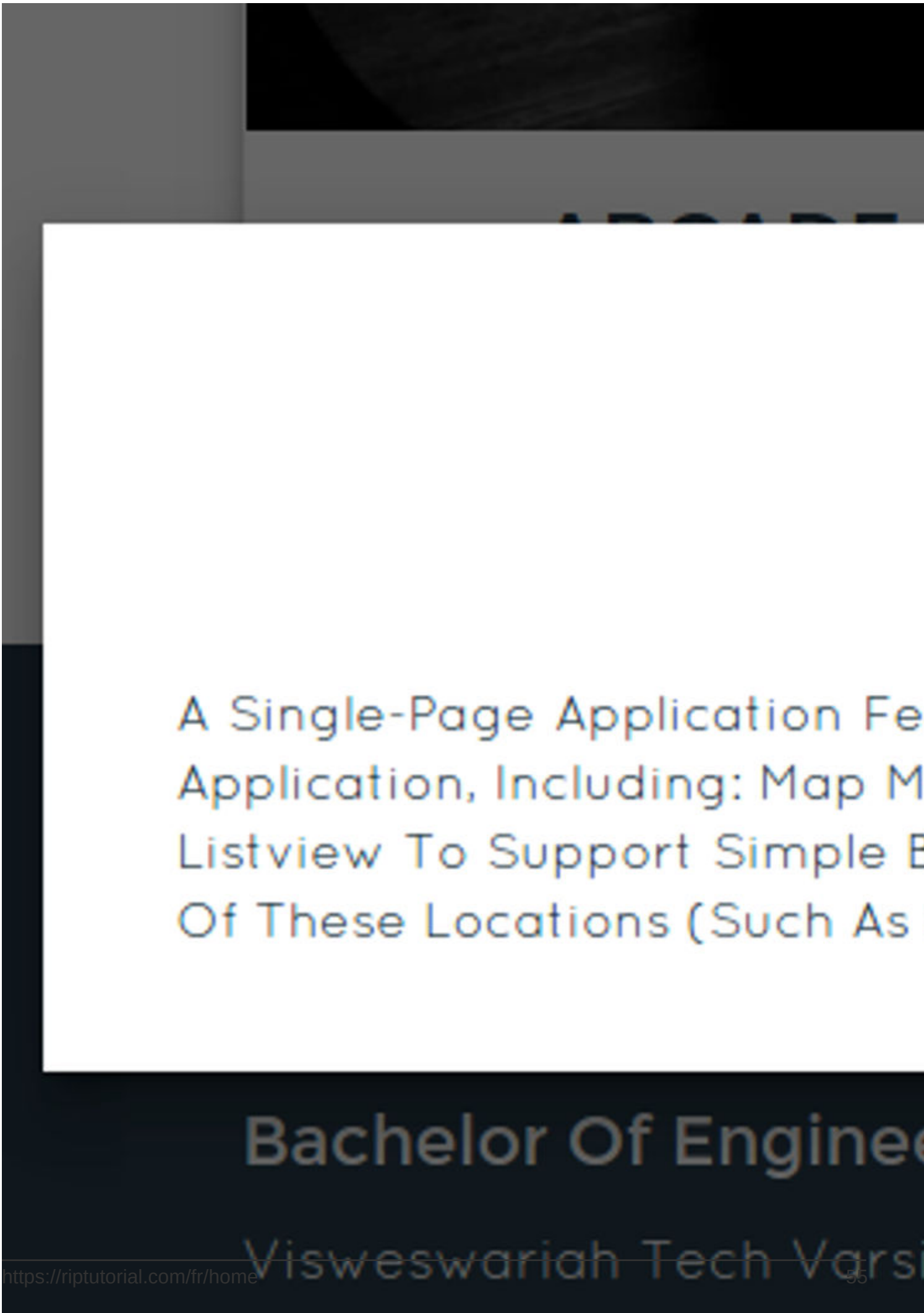
Sur n'importe quelle page sur laquelle vous souhaitez afficher votre portefeuille de produits / projets, appelez l'élément personnalisé comme suit:

```
<article id="project-neighbourhood">
    <div class="row">
        <div class="col-12 hero">
            <img src="path-to-hero-image">
        </div>
    </div>
    <tool-bar link2-share="http://www.mywebsite.com/neighbourhood" link2-
fork="https://github.com/myusername/neighbourhood" modal-id="project-neighbourhood-map"
title="project-neighbourhood-map">
                <div class="col-12">
                      <p> a single-page application featuring a map of my neighborhood or a
neighborhood I would like to visit. I will then add additional functionality to this
application, including: map markers to identify popular locations or places you'd like to
visit, a search function to easily discover these locations, and a listview to support simple
browsing of all locations. I will then research and implement third-party APIs that provide
additional information about each of these locations (such as Street View images, Wikipedia
articles, Yelp reviews, etc).</p>
               </div>
    </tool-bar>
</article>
```

et vous obtenez cela comme aperçu




Et quand vous cliquez sur info, vous obtenez ceci


A Single-Page Application Fe
Application, Including: Map M
Listview To Support Simple B
Of These Locations (Such As
Bachelor Of Engine
Visweswariah Tech Vars

https://riptutorial.com/fr/polymer/topic/7244/modaux-reutilisables-avec-polymere

# Chapitre 10: SUPER-optimisation pour la production

## Introduction

Une fois que votre projet est terminé, vous devez vous demander comment vous allez télécharger ces 100 importations HTML sur votre serveur Web et même si vous le faites, combien d'heures votre site va prendre pour charger un seul client. Dans cette rubrique, vous verrez comment convertir les problèmes de développement en fichiers HTML et js simples et raffinés.

## Syntaxe

- npm install: enregistre les packages à l'aide du gestionnaire de packages Node.js
- npm install -g: enregistre les paquets de npm en tant que global (utile pour les packages d'interface de ligne de commande)
- cd: définit le focus de la ligne de commande sur une bibliothèque spécifique dans laquelle les processus peuvent être exécutés
- vulcanize: Réduit toutes les importations HTML dans un seul fichier
- Bac à légumes: Convertit JS en ligne dans un fichier HTML en fichier JS externe

## Examples

### Installer tous les outils requis

Exécutez la commande suivante un par un:

```
npm install -g vulcanize crisper
```

## Utilisation

- Vulcanize: Crunches tous les fichiers d'importation HTML dans un seul fichier
- Bac à légumes: extrait les js en ligne dans son propre fichier

**Remarque** : les utilisateurs d'Ubuntu peuvent avoir besoin de préfixer la commande ci-dessus avec `sudo` .

### Mettre tous ensemble

Placez toutes les importations html dans vos fichiers dans un seul fichier `elements.html` . Ne vous inquiétez pas si un fichier est importé plusieurs fois, il sera réduit à une seule importation.

### En cours d'exécution vulcaniser et plus croustillant

sur votre fichier `elements.html` , exécutez les commandes suivantes:

```
cd PATH/TO/IMPORTFILE/
vulcanize elements.html -o elements.vulc.html --strip-comments --inline-css --inline-js
crisper --source elements.vulc.html --html build.html --js build.js
```

Vulcanize récupéré le code source de toutes les importations, puis remplacé les importations par leur code source.
Crisper a pris tous les js du fichier elements.vulc.html, l'a placé dans un seul fichier build.js, défini une balise de `script` référençant le fichier `build.js` dans le fichier `build.html`

## Minifier les fichiers

- Ouvrir un minificateur HTML
- Ouvrir `build.html`
- Copier tout son code
- Dans la première zone de texte du minfier HTML, collez le code que vous avez copié à partir de `build.html`
- Cliquez **sur le** bouton **Minify**
- Dans la deuxième zone de texte, le code minifié apparaîtra. Bien reçu
- Créez un fichier `build.min.html` et collez-y tout votre code copié
- Ouvrez JSCompress
- Sélectionnez 2ème onglet, c.-à-d. **Télécharger des fichiers JavaScript**
- Cliquez sur `Choose Files`
- Sélectionnez le fichier `build.js`
- Cliquez sur le bouton *Compresser*
- Téléchargez le fichier en tant que `build.js` dans le même répertoire que build.min.html

## import build.min.html

Supprimez toutes les importations précédentes des fichiers HTmL à partir desquels vous avez copié les importations. Remplacer les importations par

```
<link rel="import" href="PATH/TO/IMPORTFILE/build.min.html">
```

Lire SUPER-optimisation pour la production en ligne:
https://riptutorial.com/fr/polymer/topic/9336/super-optimisation-pour-la-production

# Chapitre 11: table de données de fer

## Examples

### Bonjour le monde

Point de départ pour `iron-data-table` sur le `iron-data-table` .

Jsbin de travail

```
<!DOCTYPE html>
<html>
  <head>
    <base href="https://polygit.org/polymer+:master/iron-data-
table+Saulis+:master/components/">
    <link rel="import" href="polymer/polymer.html">

    <script src="webcomponentsjs/webcomponents-lite.min.js"></script>

    <link rel="import" href="iron-ajax/iron-ajax.html">
    <link rel="import" href="paper-button/paper-button.html">

    <link rel="import" href="iron-data-table/iron-data-table.html">
  </head>
  <body>
    <dom-module id="x-foo">
      <template>
        <style>
        </style>
        <paper-button on-tap="msg">Click Me</paper-button>

        <iron-ajax auto
          url="https://saulis.github.io/iron-data-table/demo/users.json"
          last-response="{{users}}"
          >
        </iron-ajax>
        <iron-data-table selection-enabled items="[[users.results]]">
          <data-table-column name="Picture" width="50px" flex="0">
            <template>
              <img src="[[item.user.picture.thumbnail]]">
            </template>
          </data-table-column>
          <data-table-column name="First Name">
            <template>[[item.user.name.first]]</template>
          </data-table-column>
          <data-table-column name="Last Name">
            <template>[[item.user.name.last]]</template>
          </data-table-column>
          <data-table-column name="Email">
            <template>[[item.user.email]]</template>
          </data-table-column>
        </iron-data-table>

      </template>
      <script>
        (function(){
```

```
          'use strict';
          Polymer({
            is: 'x-foo',
            msg: function() {
              console.log('This proves Polymer is working!');
            },
          });
        })();
      </script>
    </dom-module>
    <x-foo></x-foo>
  </body>
</html>
```

## Import CSS

Importer une feuille de style externe.

Jsbin de travail

```
<!DOCTYPE html>
<html>
  <head>
    <base href="https://polygit.org/polymer+:master/iron-data-
table+Saulis+:master/components/">
    <link rel="import" href="polymer/polymer.html">

    <script src="webcomponentsjs/webcomponents-lite.min.js"></script>

    <link rel="import" href="iron-ajax/iron-ajax.html">
    <link rel="import" href="paper-button/paper-button.html">

    <link rel="import" href="iron-data-table/iron-data-table.html">
    <link rel="import" href="iron-data-table/default-styles.html">
  </head>
  <body>
    <dom-module id="x-foo">
      <template>
        <style>
        </style>
        <paper-button on-tap="msg">Click Me</paper-button>

        <iron-ajax auto
          url="https://saulis.github.io/iron-data-table/demo/users.json"
          last-response="{{users}}"
          >
        </iron-ajax>
        <iron-data-table selection-enabled items="[[users.results]]">
          <data-table-column name="Picture" width="50px" flex="0">
            <template>
              <img src="[[item.user.picture.thumbnail]]">
            </template>
          </data-table-column>
          <data-table-column name="First Name">
            <template>[[item.user.name.first]]</template>
          </data-table-column>
          <data-table-column name="Last Name">
            <template>[[item.user.name.last]]</template>
          </data-table-column>
```

```
          <data-table-column name="Email">
            <template>[[item.user.email]]</template>
          </data-table-column>
        </iron-data-table>

      </template>
      <script>
        (function(){
          'use strict';
          Polymer({
            is: 'x-foo',
            msg: function() {
              console.log('This proves Polymer is working!');
            },
          });
        })();
      </script>
    </dom-module>
    <x-foo></x-foo>
  </body>
</html>
```

## Détails de la ligne

Développez les détails de la ligne pour afficher des données supplémentaires.

Jsbin de travail

```
<!DOCTYPE html>
<html>
  <head>
    <base href="https://polygit.org/polymer+:master/iron-data-
table+Saulis+:master/components/">
    <link rel="import" href="polymer/polymer.html">

    <script src="webcomponentsjs/webcomponents-lite.min.js"></script>

    <link rel="import" href="iron-ajax/iron-ajax.html">
    <link rel="import" href="paper-button/paper-button.html">

    <link rel="import" href="iron-data-table/iron-data-table.html">
    <link rel="import" href="iron-data-table/default-styles.html">
  </head>
  <body>
    <dom-module id="x-foo">
      <template>
        <style>
          #grid1 data-table-row-detail {
            height: 100px;
          }
          #grid1 .detail {
            width: 100%;
            display: flex;
            justify-content: space-around;
            align-items: center;
            border: 2px solid #aaa;
          }
        </style>
        <paper-button on-tap="msg">Click Me</paper-button>
```

```
        <iron-ajax auto
          url="https://saulis.github.io/iron-data-table/demo/users.json"
          last-response="{{users}}"
          >
        </iron-ajax>
        <iron-data-table id="grid1" details-enabled items="[[users.results]]">
          <template is="row-detail">
            <div class="detail">
              <img src="[[item.user.picture.medium]]">
              <p>[[item.user.username]]</p>
              <p>[[item.user.email]]</p>
            </div>
          </template>
          <data-table-column name="First Name">
            <template>[[item.user.name.first]]</template>
          </data-table-column>
          <data-table-column name="Last Name">
            <template>[[item.user.name.last]]</template>
          </data-table-column>
        </iron-data-table>
      </template>
      <script>
        (function(){
          'use strict';
          Polymer({
            is: 'x-foo',
            msg: function() {
              console.log('This proves Polymer is working!');
            },
          });
        })();
      </script>
    </dom-module>
    <x-foo></x-foo>
  </body>
</html>
```

## Modifier les détails de la ligne

### Jsbin de travail

```
<!DOCTYPE html>
<html>
  <head>
    <base href="https://polygit.org/polymer+:master/iron-data-
table+Saulis+:master/components/">
    <link rel="import" href="polymer/polymer.html">

    <script src="webcomponentsjs/webcomponents-lite.min.js"></script>

    <link rel="import" href="iron-ajax/iron-ajax.html">
    <link rel="import" href="paper-button/paper-button.html">
    <link rel="import" href="paper-input/paper-input.html">

    <link rel="import" href="iron-data-table/iron-data-table.html">
    <link rel="import" href="iron-data-table/default-styles.html">
  </head>
  <body>
```

```
    <dom-module id="x-foo">
      <template>
        <style>
          #grid1 data-table-row-detail {
            height: 100px;
          }
          #grid1 .detail {
            width: 100%;
            display: flex;
            justify-content: space-around;
            align-items: center;
            border: 2px solid #aaa;
          }
        </style>
        <paper-button on-tap="msg">Click Me</paper-button>

        <iron-ajax auto
          url="https://saulis.github.io/iron-data-table/demo/users.json"
          last-response="{{users}}"
          >
        </iron-ajax>
        <iron-data-table id="grid1" details-enabled items="[[users.results]]">
          <template is="row-detail">
            <div class="detail">
              <img src="[[item.user.picture.medium]]">
              <p>[[item.user.username]]</p>
              <p>[[item.user.email]]</p>
              <paper-input></paper-input>
            </div>
          </template>
          <data-table-column name="First Name">
            <template>[[item.user.name.first]]</template>
          </data-table-column>
          <data-table-column name="Last Name">
            <template>[[item.user.name.last]]</template>
          </data-table-column>
        </iron-data-table>
      </template>
      <script>
        (function(){
          'use strict';
          Polymer({
            is: 'x-foo',
            msg: function() {
              console.log('This proves Polymer is working!');
            },
          });
        })();
      </script>
    </dom-module>
    <x-foo></x-foo>
  </body>
</html>
```

## Modifier les détails de la ligne à l'aide d'un sous-élément

Cet exemple utilise un élément distinct pour modifier les données liées au modèle de `row-detail` .

Jsbin de travail

```
<!DOCTYPE html>
<html>
  <head>
    <base href="https://polygit.org/polymer+:master/iron-data-
table+Saulis+:master/components/">
    <link rel="import" href="polymer/polymer.html">

    <script src="webcomponentsjs/webcomponents-lite.min.js"></script>

    <link rel="import" href="iron-ajax/iron-ajax.html">
    <link rel="import" href="paper-button/paper-button.html">
    <link rel="import" href="paper-input/paper-input.html">

    <link rel="import" href="iron-data-table/iron-data-table.html">
    <link rel="import" href="iron-data-table/default-styles.html">
  </head>
  <body>
    <dom-module id="row-detail">
      <template>
        <img src="[[item.user.picture.medium]]">
        <span>[[item.user.username]]</span>
        <span>[[item.user.email]]</span>
        <paper-input></paper-input>
      </template>
      <script>
        (function() {
          'use strict';
          Polymer({
            is: 'row-detail',
            properties: {
              item: Object,
            },
          });
        })();
      </script>
    </dom-module>
    <dom-module id="x-foo">
      <template>
        <style>
          #grid1 data-table-row-detail {
            height: 150px;
          }
          #grid1 .detail {
            width: 100%;
            display: flex;
            justify-content: space-around;
            align-items: center;
            border: 2px solid #aaa;
          }
        </style>
        <paper-button on-tap="msg">Click Me</paper-button>

        <iron-ajax auto
          url="https://saulis.github.io/iron-data-table/demo/users.json"
          last-response="{{users}}"
          >
        </iron-ajax>
        <iron-data-table id="grid1" details-enabled items="[[users.results]]">
          <template is="row-detail">
            <div class="detail">
              <row-detail item="{{item}}"></row-detail>
```

```
            </div>
          </template>
          <data-table-column name="First Name">
            <template>[[item.user.name.first]]</template>
          </data-table-column>
          <data-table-column name="Last Name">
            <template>[[item.user.name.last]]</template>
          </data-table-column>
        </iron-data-table>
      </template>
      <script>
        (function(){
          'use strict';
          Polymer({
            is: 'x-foo',
            msg: function() {
              console.log('This proves Polymer is working!');
            },
          });
        })();
      </script>
    </dom-module>
    <x-foo></x-foo>
  </body>
</html>
```

# Remarque

Il existe actuellement un problème décrit ici qui entraîne la réduction de la section des détails de la ligne s'il contient un sous-élément. Le correctif doit inclure `tabindex="0"` comme suit. ( Voir cette réponse Stack Overflow .)

x-foo.html

```
<template is="row-detail">
  <div class="detail">
    <row-detail item="{{item}}" tabindex="0"></row-detail>
  </div>
</template>
```

### Sélectionnez la ligne, empêchez la désélection

Le comportement par défaut consiste à désélectionner la ligne lorsque vous cliquez deux fois. Dans certains cas d'utilisation, vous pouvez désactiver ce comportement de désélection.

# Remarque

`table.deselectItem(item)` désélectionnera impérativement un élément. Cela fonctionne à la fois avec un `item` ou un `index` (lorsque vous utilisez un tableau d'éléments) comme argument.

Jsbin de travail

```
<!DOCTYPE html>
<html>
  <head>
    <base href="https://polygit.org/polymer+:master/iron-data-
table+Saulis+:master/components/">
    <link rel="import" href="polymer/polymer.html">

    <script src="webcomponentsjs/webcomponents-lite.min.js"></script>

    <link rel="import" href="iron-ajax/iron-ajax.html">
    <link rel="import" href="paper-button/paper-button.html">

    <link rel="import" href="iron-data-table/iron-data-table.html">
    <link rel="import" href="iron-data-table/default-styles.html">
  </head>
  <body>
    <x-foo></x-foo>
    <dom-module id="x-foo">
      <template>
        <style>
        </style>
        [[_computeSelectedStr(selectedItem)]]

        <iron-ajax
            auto
            url="https://saulis.github.io/iron-data-table/demo/users.json"
            last-response="{{users}}"
            >
        </iron-ajax>
        <iron-data-table id="grid"
                         selection-enabled
                         on-deselecting-item="_deselecting"
                         items="[[users.results]]"
                         selected-item="{{selectedItem}}"
                         >
          <data-table-column name="Picture" width="50px" flex="0">
            <template>
              <img src="[[item.user.picture.thumbnail]]">
            </template>
          </data-table-column>
          <data-table-column name="First Name">
            <template>[[item.user.name.first]]</template>
          </data-table-column>
          <data-table-column name="Last Name">
            <template>[[item.user.name.last]]</template>
          </data-table-column>
          <data-table-column name="Email">
            <template>[[item.user.email]]</template>
          </data-table-column>
        </iron-data-table>

      </template>
      <script>
        (function(){
              'use strict';
          Polymer({
            is: 'x-foo',
            observers: [
              '_selectedItemChanged(selectedItem)' ,
            ],
            _selectedItemChanged: function(ob) {
```

```
            console.log('selectedItem', ob);
          },
          _computeSelectedStr: function(ob) {
            return JSON.stringify(ob);
          },
          _deselecting: function(e) {
            e.preventDefault();
          }
        });
      })();
    </script>
  </dom-module>
 </body>
</html>
```

Lire table de données de fer en ligne: https://riptutorial.com/fr/polymer/topic/6447/table-de-donnees-de-fer

# Chapitre 12: Test d'unité

## Remarques

Web Component Tester - l'outil pour les applications de tests unitaires créées avec Polymer. Vous obtenez un environnement de test basé sur un navigateur, configuré avec des outils tels que moka , chai , async , lodash , sinon & sinon-chai , test-fixture , accessibilité-developer-tools . WCT exécutera vos tests sur tous les navigateurs installés localement ou à distance via Sauce Labs.

## Examples

### Exemple simple utilisant Web-component-tester

# installer

```
npm install web-component-tester --save-dev
```

# mise en place

wct.conf.js

```
module.exports = {
    verbose: true,
    plugins: {
        local: {
            browsers: ['chrome']
        }
    }
};
```

# fonctionnement

```
node node_modules/web-component-tester/bin/wct
```

# test / index.html

```
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-
scale=1">
```

```
        <script src="../bower_components/webcomponentsjs/webcomponents.min.js"></script>
        <script src="../node_modules/web-component-tester/browser.js"></script>
        <link rel="import" href="../src/utils.html">

    </head>
    <body>

        <my-utils id="utils"></my-utils>

        <script>

            test('utils.isNumeric', function(){
                var utils = document.getElementById('utils');
                [1,0,-1,1.83,-9.87].forEach(function(d){
                    assert.isTrue(utils.isNumeric(d));
                });
                [true, false, null, {}, 'a', new Date()].forEach(function(d){
                    assert.isFalse(utils.isNumeric(d));
                });
            });

        </script>

    </body>
</html>
```

# src / utils.html

```
<link rel="import" href="../bower_components/polymer/polymer.html">

<dom-module id="my-utils">
    <template></template>
</dom-module>

Polymer({

    is: "my-utils",

    isNumeric: function(n) {
        return !isNaN(parseFloat(n)) && isFinite(n);
    }

});
```

Lire Test d'unité en ligne: https://riptutorial.com/fr/polymer/topic/3898/test-d-unite

# Chapitre 13: Utilisation de bibliothèques Javascript externes avec Polymer

## Introduction

Comme tous les composants Web doivent être autonomes, y compris toutes leurs dépendances, l'importation de dépendances dupliquées deviendrait rapidement un problème avec les scripts inclus. Ainsi, Web Components (et, par extension, Polymer) utilisent les importations HTML du W3C pour gérer les dépendances des composants. Ces fichiers HTML importés peuvent être mis en cache par le navigateur et ne seront chargés qu'une seule fois.

La plupart des bibliothèques externes ne sont pas encore préparées pour les importations HTML. Heureusement, créer le wrapper HTML nécessaire est simple et rapide.

## Paramètres

| Paramètre | La description |
|---|---|
| this.resolveUrl ('../ library / turf.js') | Résout l'emplacement de votre bibliothèque |
| function () {this.doSomething (argument, anotherArgument);}. bind (this)); | Rappeler. Cet exemple utilise une fermeture pour réincorporer this.doSomething () |

## Remarques

Dans cet exemple, la bibliothèque utilisée dans le composant est installée avec le gestionnaire de packages bower. Cela permet une distribution facile d'un composant dépendant de la bibliothèque. Si la bibliothèque que vous souhaitez utiliser n'est pas distribuée via un gestionnaire de paquets, elle peut tout de même être chargée de la même manière, mais votre composant nécessitera plus d'efforts pour être utilisé par d'autres.

L'exemple de chargement différé utilise une chaîne simple pour le chemin de la bibliothèque. Si l'on souhaitait éviter les constantes de type chaîne magique, les chemins pouvaient être chargés à l'aide d'un fer-ajax à partir d'un fichier JSON dans un objet et transmis entre les composants si nécessaire.

## Examples

### Importer un fichier HTML statique

1. Créez un fichier HTML (dans cet exemple, `libraries/turf.html` ) avec la bibliothèque que vous souhaitez charger:

```
<script src="../../bower_components/turf/turf.min.js"></script>
```

2. Importez le fichier HTML ( `libraries/turf.html` ) dans votre composant avec le reste de vos importations:

```
<link rel="import" href="../../bower_components/polymer/polymer.html">
<link rel="import" href="../libraries/turf.html">
```

3. Appelez votre bibliothèque (exemple d'utilisation):

```
    var point = turf.point([42.123123, -23.83839]);
```

## Chargement paresseux

1. Créez un fichier HTML (dans cet exemple, `libraries/turf.html` ) avec la bibliothèque que vous souhaitez charger:

```
<script src="../../bower_components/turf/turf.min.js"></script>
```

2. Importez et utilisez votre bibliothèque si nécessaire:

```
doSomething: function(argument, anotherArgument): {

    // If library has not been loaded, load it
    if(typeof turf == 'undefined') {
        this.importHref(this.resolveUrl('../libraries/turf.js'), function() {
            // Once the library is loaded, recursively call the function
            this.doSomething(argument, anotherArgument);
        }.bind(this));

        return;
    }

    // Example usage of a library method
    var point = turf.point([42.123123, -23.83839]);
}
```

Lire Utilisation de bibliothèques Javascript externes avec Polymer en ligne: https://riptutorial.com/fr/polymer/topic/6034/utilisation-de-bibliotheques-javascript-externes-avec-polymer

# Crédits

| S. No | Chapitres | Contributeurs |
|---|---|---|
| 1 | Commencer avec le polymère | Community, Fuzzical Logic, fybw id, Keith, Marc, Randy Askin, tony19, Ümit |
| 2 | Création d'une application à l'aide du kit de démarrage Polymer | fybw id, Puru Vijay |
| 3 | En boucle, le modèle dom-repeat. | Jp_ |
| 4 | Exemple de toggleAttribute de polymère | a1626 |
| 5 | Feuille de triche en polymère | Josef Ježek |
| 6 | Gestion des événements | a1626, Ümit |
| 7 | Google Map Mark avec cache intégré | Schrodinger's cat |
| 8 | Le débogage | willsquire |
| 9 | Modaux réutilisables avec polymère | Schrodinger's cat |
| 10 | SUPER-optimisation pour la production | Puru Vijay |
| 11 | table de données de fer | Mowzer |
| 12 | Test d'unité | kashesandr, Marc |
| 13 | Utilisation de bibliothèques Javascript externes avec Polymer | antibrian, craPkit |