eBook Gratuit

APPRENEZ ember.js

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

#ember.js

Table des matières

À propos

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

It is an unofficial and free ember.js 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 ember.js.

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: Démarrer avec ember.js

Remarques

Cette section fournit une vue d'ensemble de ce qu'est ember.js et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les sujets importants dans ember.js et établir un lien avec les sujets connexes. La documentation de ember.js étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version à jour

Version	Date de sortie
2.14.0 beta	2017-04-29
2.13.0	2017-04-29

Examples

Installation ou configuration

Commencer avec Ember est facile. Les projets Ember sont créés et gérés via notre outil de génération de ligne de commande Ember CLI. Cet outil fournit:

- Gestion moderne des ressources applicatives (y compris concaténation, minification et gestion des versions).
- Générateurs pour aider à créer des composants, des itinéraires, etc.
- Une mise en page de projet classique, facilitant l'approche des applications Ember existantes.
- Prise en charge de JavaScript ES2015 / ES6 via le projet Babel . Cela inclut la prise en charge des modules JavaScript , utilisés dans ce guide.
- Un harnais de test complet QUnit .
- La capacité de consommer un écosystème croissant de Ember Addons .

Les dépendances

Node.js et npm

Ember CLI est construit avec JavaScript et attend le runtime Node.js. Il nécessite également des

dépendances récupérées via npm . npm est fourni avec Node.js, donc si Node.js est installé sur votre ordinateur, vous êtes prêt à partir.

Ember nécessite Node.js 0.12 ou supérieur et npm 2.7 ou supérieur. Si vous n'êtes pas sûr d'avoir Node.js ou la bonne version, exécutez ceci sur votre ligne de commande:

```
node --version
npm --version
```

Si vous obtenez une erreur *"commande introuvable"* ou une version obsolète pour le noeud:

- Les utilisateurs Windows ou Mac peuvent télécharger et exécuter ce programme d'installation de Node.js.
- Les utilisateurs de Mac préfèrent souvent installer Node en utilisant Homebrew . Après avoir installé Homebrew, exécutez le `brew install node` d'installation de `brew install node` pour installer Node.js.
- Les utilisateurs de Linux peuvent utiliser ce guide pour l'installation de Node.js sous Linux .

Si vous obtenez une version obsolète de npm, exécutez `npm install -g npm` .

Git

Ember exige que Git gère un grand nombre de ses dépendances. Git est livré avec Mac OS X et la plupart des distributions Linux. Les utilisateurs de Windows peuvent télécharger et exécuter ce programme d'installation de Git .

Gardien (optionnel)

Sur Mac et Linux, vous pouvez améliorer les performances de surveillance des fichiers en installant Watchman .

PhantomJS (optionnel)

Vous pouvez exécuter vos tests à partir de la ligne de commande avec PhantomJS, sans qu'un navigateur soit ouvert. Consultez les instructions de téléchargement de PhantomJS .

Installation

Installez Ember en utilisant npm:

```
npm install -g ember-cli
```

Pour vérifier que votre installation a réussi, exécutez:

```
ember -v
```

Si un numéro de version est affiché, vous êtes prêt à partir.

Créer une application

Ember CLI vous permet d'utiliser l'une des deux options suivantes pour générer une nouvelle application:

1. Créez un dossier et exécutez `ember init` (génère la structure de l'application et configure git et effectue votre premier commit)
2. Exécutez le `ember new <app name>` (crée un dossier avec le nom spécifié, le lance et exécute `ember init`)

Une fois le processus de génération terminé, démarrez un serveur live-reload dans le dossier de l'application en exécutant:

```
ember server
```

ou `ember s` pour faire court. * *Ta-da, maintenant vous avez une application Ember en cours d'exécution!* Documents officiels

Créer votre premier modèle

Créons un nouveau modèle en utilisant la commande `ember generate` .

```
ember generate template application
```

Le modèle d' `application` est toujours à l'écran lorsqu'un utilisateur visite votre application. Dans votre éditeur de choix, ouvrez `app/templates/application.hbs` et ajoutez le code suivant:

```
<h2>My first Ember application</h2>

{{outlet}}
```

Vous devriez maintenant voir le nouveau texte ajouté sur la page d'accueil de votre application. Notez également que Ember détecte automatiquement le nouveau fichier et recharge la page pour vous. Neat, non?

Déployer l'application

Pour déployer une application Ember, transférez simplement la sortie de la version ember sur un serveur Web. Cela peut être fait avec les outils de transfert de fichiers Unix standard tels que `rsync` ou `scp` . Il existe également des services qui vous permettront de déployer facilement.

```
ember build
scp -r dist/* myserver.com:/var/www/public/
```

normalement nous utiliserions `ember build --environment=production` qui fait plus pour rendre votre code prêt pour la production (gzip et minify code).

Comment travailler avec les plugins JavaScript

Il y a quatre façons de travailler avec les plugins JavaScript,

1. Ember add-on
2. Importer les plugins JavaScript globalement
3. Consommez les plugins AMD nommés
4. Via `ember-browserify`

Affecter des ports localhost (en particulier des problèmes de permissions / disponibilité, exécutant simultanément plusieurs sites de braises)

Parfois, il est utile d'assigner manuellement un ou plusieurs ports par rapport aux paramètres par défaut. Cela peut résoudre les problèmes de disponibilité / autorisation de port ou permettre l'exécution de plusieurs instances de ember à la fois.

Pour que ember-cli tente d'identifier et d'attribuer un port disponible, utilisez:

```
ember serve --port 0
```

Aide à la journée: "Passez 0 pour sélectionner automatiquement un port disponible". (Dans un terminal, tapez help de la braise).

Pour exécuter simultanément plusieurs sites de braise, chacun a besoin de son propre serveur et de ses ports de rechargement en direct. Une approche simple: dans des fenêtres de terminal distinctes, accédez à chaque instance et utilisez les éléments suivants pour les lancer avec leurs propres ports:

```
ember serve --port 0 --live-reload-port 0
```

Si vous obtenez une erreur concernant la disponibilité ou l'autorisation dans l'un de ces cas, entrez le script python suivant à l'invite du terminal pour identifier un port disponible:

```
python -c 'import socket; s=socket.socket(); s.bind(("", 0)); print(s.getsockname()[1]);
s.close()'
```

Utilisez les résultats pour spécifier les ports que vous savez maintenant être disponibles:

```
ember serve --port <known_port_1> --live-reload-port <known_port_2>
```

Lire Démarrer avec ember.js en ligne: https://riptutorial.com/fr/ember-js/topic/905/demarrer-avec-ember-js

Chapitre 2: Assistant de formatage de devise

Remarques

Plus de détails disponibles dans les guides Ember , d'où cet exemple est tiré.

Compatible avec Ember 2.2.0+ (2.11.0 était le plus récent au moment de la rédaction)

Examples

Créer un nouvel assistant

Utilisez Ember CLI pour générer une nouvelle aide dans votre application:

```
ember generate helper format-currency
```

Ensuite, éditez `helpers/format-currency.js` pour contenir les éléments suivants:

```
import Ember from 'ember';

export function formatCurrency([value, ...rest]) {
  const dollars = Math.floor(value / 100);
  const cents = value % 100;
  const sign = '$';

  if (cents.toString().length === 1) { cents = '0' + cents; }
  return `${sign}${dollars}.${cents}`;
}

export default Ember.Helper.helper(formatCurrency);
```

Vous pouvez maintenant utiliser `{{format-currency model.someNumericValue}}` dans les modèles.

Un test unitaire pour le nouvel assistant est automatiquement créé dans `tests/unit/helpers/`

Lire Assistant de formatage de devise en ligne: https://riptutorial.com/fr/ember-js/topic/6647/assistant-de-formatage-de-devise

Chapitre 3: Comment importer une bibliothèque / plugin JavaScript

Introduction

Ouvrez le répertoire de votre projet ember.js, vous y trouverez un fichier nommé ember-cli-build.js. Vous pouvez installer Vos bibliothèques ou vos plugins en utilisant bower, puis pointez l'importation vers le dossier bower_components, mais si vous avez un fichier à ajouter, faites-les glisser dans le dossier de votre projet et écrivez le fichier app.import dans ce fichier.

Syntaxe

- app.import ('chemin du fichier à partir du dossier du projet / fichier.js');

Examples

Exemple de fichier ember-cli-build.js

```
var EmberApp = require('ember-cli/lib/broccoli/ember-app');

module.exports = function(defaults) {
  var app = new EmberApp(defaults, {
    // Add options here
    datatables: {
        core: true,
        style: 'bs',
        extensions: [
            { name: 'buttons', extensions: ['colVis','flash','html5','print'] },
            { name: 'responsive', style: 'bs' },
            'select'
        ],
        pdfmake: false,
        vfs_fonts: false,
        jszip: true
    }
  });
  //Imports:
  app.import('bower_components/js-cookie/src/js.cookie.js');
  app.import('bower_components/moment/min/moment.min.js');
  app.import('bower_components/crypto-js/crypto-js.js');
  // Use `app.import` to add additional libraries to the generated
  // output files.
  //
  // If you need to use different assets in different
  // environments, specify an object as the first parameter. That
  // object's keys should be the environment name and the values
  // should be the asset to use in that environment.
  //
  // If the library that you are including contains AMD or ES6
  // modules that you would like to import into your application
```

```
  // please specify an object with the list of modules as keys
  // along with the exports of each module as its value.

  return app.toTree();
};
```

Lire Comment importer une bibliothèque / plugin JavaScript en ligne:
https://riptutorial.com/fr/ember-js/topic/9239/comment-importer-une-bibliotheque---plugin-javascript

Chapitre 4: Comment mettre à jour Ember, Ember Data et Ember CLI

Remarques

- Pour trouver la dernière version stable d' **Ember** , cliquez ici .
- Pour trouver la dernière version stable d' **Ember Data** , cliquez ici .
- Pour trouver la dernière version stable de **Ember CLI** , cliquez ici .

Toutes ces étapes ont été trouvées sur la note de publication d'Ember cli .

Examples

Mise à jour de Ember

Dans cet exemple, `2.13.2` est la dernière version. Nous l'installons via `bower` , en spécifiant la version particulière en tant que `ember#2.13.2` et en incluant l'indicateur de sauvegarde pour le conserver dans bower.json.

Au moment de l'écriture de cet article, la dernière version est `2.13.2` . A partir de la ligne de commande, à la racine du répertoire de votre application, exécutez:

```
bower install ember#2.13.2 --save
```

Vous pouvez être invité à choisir votre version d'Ember. Si vous êtes, ajoutez votre réponse avec un `!` pour vous assurer qu'il est enregistré.

Mise à jour des données Ember

Ember Data étant un module complémentaire Ember CLI, nous pouvons l'installer comme tout autre module complémentaire en utilisant `ember install` .

Au moment de la rédaction de cet article, la dernière version est `2.13.1` . A partir de la ligne de commande, à la racine du répertoire de votre application, exécutez:

```
ember install ember-data@2.13.1
```

Mise à jour de la CLI Ember

Ember CLI est un package npm normal. Pour le mettre à jour, il faut le désinstaller puis installer la version souhaitée.

Au moment de l'écriture de cet article, la dernière version est 2.13.2. A partir de la ligne de commande exécutée:

```
npm uninstall -g ember-cli
npm cache clean
bower cache clean
npm install -g ember-cli@2.13.2
```

Pour vérifier que la version correcte a été installée, procédez comme suit:

```
ember -v
```

Ensuite, mettez à jour votre projet. Cela effacera le cache et mettra à jour la version de la CLI Ember dans `package.json` . La dernière partie exécutera le nouveau plan du projet dans votre répertoire de projets. Suivez simplement les instructions et passez en revue toutes les modifications.

```
rm -rf node_modules bower_components dist tmp
npm install --save-dev ember-cli@2.13.2
npm install
bower install
ember init
```

les références

Lire Comment mettre à jour Ember, Ember Data et Ember CLI en ligne: https://riptutorial.com/fr/ember-js/topic/1722/comment-mettre-a-jour-ember--ember-data-et-ember-cli

Chapitre 5: Composant - communication entre enfant et composant parent.

Syntaxe

- `(yield` - Vous permet d'exporter des éléments d'un composant
- `(hash` - Permet d'exporter un hachage ou un objet, car cela est nécessaire pour appeler des composants de l' enfant au sein du bloc du parent L'exigence est qu'il ya un. . pour le composant à créer
- `(component` - Crée le composant enfant qui peut prendre n'importe quoi dans le contexte du parent. Le composant peut être curry, car il est uniquement appelé lorsque l'utilisateur l'utilise, donc ajoutez autant d'attributs que nécessaire et l'utilisateur peut ajouter le reste .
- `(action` - Crée une action basée sur une fonction ou une chaîne pointant vers une fonction dans le hachage d' `actions` du composant parent dans ce cas.

Remarques

Pour créer des composants qui interagissent avec un composant parent, les composants composables constituent la meilleure option, bien qu'ils requièrent Ember 2.3+.

Examples

Composants Composables

Dans `parent-component.hbs`

```
{{yield (hash
   child=(
     component 'child-component'
     onaction=(action 'parentAction')
   )
)}}
```

Dans `parent-component.js`

```
export default Ember.Component.extend({
  actions: {
    // We pass this action to the child to call at it's discretion
    parentAction(childData) {
      alert('Data from child-component: ' + childData);
    }
  }
});
```

Inside `child-component.js`

```
export default Ember.Component.extend({
  // On click we call the action passed down to us from the parent
  click() {
    let data = this.get('data');
    this.get('onaction')(data);
  }
});
```

`usage.hbs` intérieur.hbs

```
{{#parent-component as |ui|}}
  {{#each model as |item|}}
    {{ui.child data=item}}
  {{/each}}
{{/parent-component}}
```

Lire Composant - communication entre enfant et composant parent. en ligne: https://riptutorial.com/fr/ember-js/topic/3066/composant---communication-entre-enfant-et-composant-parent-

Chapitre 6: Essai

Introduction

Créer et maintenir une suite de tests complète devrait être une priorité pour chaque développeur. Tester dans Ember.js implique de gérer l'asynchronie, Ember Run Loop et de moquer votre API. Les développeurs Ember.js ont souvent du mal à écrire des tests. Cependant, il existe des conseils qui pourraient vous faire économiser du temps et de l'énergie.

Examples

En attente de promesses dans les tests de manière élégante

Vous pouvez faire en sorte que la `function` transmise à la méthode `test()` `async` - vous pouvez alors utiliser le mot-clé `await` . Votre test attendra jusqu'à ce que Promises se résolve et que le test du code asynchrone devienne plus facile et plus lisible. Dans l'exemple suivant, l'appel qui renvoie une promesse est `changeset.validate()` . S'il vous plaît noter également enroulant `set` appel à `Ember.run` . Définir la quantité a des effets asynchrones (observateurs, propriétés calculées) et nous devons donc l'emballer dans `Ember.run` .

```
test('quantity validation: greater than 0', async function (assert) {
    assert.expect(3);

    const model = this.subject({
        quantity: 1
    });

    const changeset = createChangeset(model);

    await changeset.validate();

    assert.ok(!changeset.get('error.quantity'));

    Ember.run(() => {
        changeset.set('quantity', -1);
    });

    await changeset.validate();

    assert.equal(changeset.get('error.quantity.validation.length'), 1);
    assert.ok(!changeset.get('isValid'));
});
```

Lire Essai en ligne: https://riptutorial.com/fr/ember-js/topic/10722/essai

Chapitre 7: Format de date

Examples

Aide pour un format de date et heure propre.

Lorsque vous voulez la date et l'heure actuelles, vous pouvez le faire avec la fonction Javascript `Date` , mais renverra le format suivant qui n'est pas toujours utile: `Wed Jun 07 2017 13:26:15 GMT+0200 (Romance (zomertijd))` .

Copiez le code suivant dans `app/helpers/helpers.js` et appelez simplement `getCurrentDateAndFormat()` au lieu de `new Date()` .

```
export function getCurrentDateAndFormat() {
    let today = new Date();
    let dd = today.getDate();
    let MM = today.getMonth()+1; //January is 0!
    let hh = today.getHours();
    let mm = today.getMinutes();
    let yyyy = today.getFullYear();

    if (dd<10) {
        dd= '0'+dd;
    }

    if (MM<10) {
        MM= '0'+MM;
    }

    if (hh<10) {
        hh= '0'+hh;
    }

    if (mm<10) {
        mm= '0'+mm;
    }

    today = dd+'/'+MM+'/'+yyyy+" "+hh+"h"+mm;

    return today;
}
```

L'assistant extrait toutes les valeurs de temps distinctes, ajoute un 0 si la valeur est inférieure à 10 (pour le format et la lisibilité) et les réassemble dans un format plus adapté. Dans ce cas: jour, mois, année, heures et minutes.

```
today = dd+'/'+MM+'/'+yyyy+" "+hh+"h"+mm;
```

affichera: `07/06/2017 13h26`

```
today = MM+'/'+dd+'/'+yyyy+" "+hh+"h"+mm;
```

affichera: `06/07/2017 13h26`

Changer le mois et la date en fonction de votre région est aussi simple que de remplacer `MM` par `dd` et vice versa, comme le montre l'exemple ci-dessus.

Lire Format de date en ligne: https://riptutorial.com/fr/ember-js/topic/10153/format-de-date

Chapitre 8: Initialiser Foundation ou Bootstrap sur ember-cli de manière appropriée

Introduction

Bootstrap: Je pense que ce n'est pas la bonne façon. Le meilleur moyen à mon avis est un addon ember-bootstrap.

ember-bootstrap utilise les classes CSS Bootstrap en remplaçant les comportements des composants implémentés par Bootstrap dans bootstrap.js, tels que toggle, navbar, modal, etc., avec des composants Ember natifs compatibles avec la classe CSS.

Foundation: Il y a un addon appelé Ember CLI Foundation 6 SASS, il est également installé en ligne de commande.

Paramètres

Paramètre	Usage
Ember installer	Télécharger un nouveau package d'extension avec Ember
npm installer	Téléchargez un nouveau package d'extension à l'aide de node.js
TOUPET	Langage CSS nécessaire dans Foundation
Ember-cli-build.js	Fichier avec importations Ember, configuration, etc.
{{# bs-modal-simple}}	Création d'un nouveau modal bootstrap
fondu = fondu	Définir les animations modales
{{# bs-button}}	Bouton avec un look cool Bootstrap
{{# bs-form onSubmit (action = "Submit")}}	Nouveau formulaire avec une action après soumission

Remarques

Les deux addons ne sont pas les miens, je pensais que ce serait sympa de vous les présenter, il y a des pages github des addons:

Ember Bootstrap: https://github.com/kaliber5/ember-bootstrap

Fondation Ember 6 https://github.com/acoustep/ember-cli-foundation-6-sass

Vous pouvez y trouver de la documentation.

Examples

Installer ember-bootstrap avec la version par défaut

```
ember install ember-bootstrap
```

Instal ember-bootstrap avec la version 4 et SASS - expérimental

```
ember generate ember-bootstrap --bootstrap-version=4 --preprocessor=sass
```

Installer SASS et Fondation

```
npm install --save-dev ember-cli-sass
ember install ember-cli-foundation-6-sass
```

Installer les dépendances de la fondation

```
ember g ember-cli-foundation-6-sass
```

Fichier de construction Ember avec des addons Foundation

```
// ember-cli-build.js

/* global require, module */
var EmberApp = require('ember-cli/lib/broccoli/ember-app');

module.exports = function(defaults) {
  var app = new EmberApp(defaults, {
    // Add options here
    'ember-cli-foundation-6-sass': {
            'foundationJs': [
                'core',
                'util.box',
                'util.keyboard',
                'util.mediaQuery',
                'util.motion',
                'util.nest',
                'util.timerAndImageLoader',
                'util.touch',
                'util.triggers',
                'abide',
                'accordion',
```

```
                'accordionMenu',
                'drilldown',
                'dropdown',
                'dropdownMenu',
                'equalizer',
                'interchange',
                'magellan',
                'offcanvas',
                'orbit',
                'responsiveMenu',
                'responsiveToggle',
                'reveal',
                'slider',
                'sticky',
                'tabs',
                'toggler',
                'tooltip'
            ]
        },
    }
  });

  return app.toTree();
};
```

Formulaire d'échantillon Ember Bootstrap

```
{{#bs-form model=this onSubmit=(action "submit") as |form|}}
  {{#form.element label="Email" placeholder="Email" property="email" as |el|}}
    <div class="input-group">
      <input value={{el.value}} class="form-control" placeholder="Email" oninput={{action (mut
el.value) value="target.value"}} onchange={{action (mut el.value) value="target.value"}}
id={{el.id}} type="text">
      <span class="input-group-addon">@example.com</span>
    </div>
  {{/form.element}}
  {{bs-button defaultText="Submit" type="primary" buttonType="submit"}}
{{/bs-form}}
```

Lire Initialiser Foundation ou Bootstrap sur ember-cli de manière appropriée en ligne: https://riptutorial.com/fr/ember-js/topic/10176/initialiser-foundation-ou-bootstrap-sur-ember-cli-de-maniere-appropriee

Chapitre 9: Le débogage

Examples

Enregistrement EmberData

Les modèles de données de braise ont une méthode toJSON qui extrait les données pertinentes:

```
console.log(model.toJSON());
```

Cette méthode utilise JSONSerializer pour créer la représentation JSON.

Si vous souhaitez enregistrer les données d'une manière plus spécifique à l'application, vous pouvez utiliser sérialiser :

```
model.serialize();
```

qui utilise la stratégie de sérialisation que vous pouvez définir dans l'adaptateur du magasin pour créer une représentation JSON du modèle.

Tous les objets d'une application Ember, y compris les modèles de données Ember, héritent d' Ember.CoreObject , qui possède une méthode toString qui imprime cette représentation:

```
<app-name@ember-type:object-name:id>
```

Explication:

- `app-name` est le nom de votre application
- `ember-type` est le type de braise de l'objet que vous enregistrez (peut être contrôleur, route, etc.)
- `object-name` est le nom de l'objet que vous enregistrez (nom de votre modèle ou contrôleur, ou itinéraire, etc.)
- id est soit un guId create avec Ember.guidFor ou, par exemple, l'identifiant du modèle.

Vous pouvez remplacer cette valeur en utilisant la méthode `toStringExtension` dans votre modèle particulier.

Pour l'exemple de comparaison, voici comment la journalisation d'un contrôleur d'application peut ressembler:

```
<my-awesome-app@controller:application::ember324>
```

Exécution du code de débogage uniquement

Ember a une méthode globale statique appelée runInDebug qui peut exécuter une fonction

destinée au débogage.

```
Ember.runInDebug(() => {
  // this code only runs in dev mode
});
```

Dans une version de production, cette méthode est définie comme une fonction vide (NOP). Les utilisations de cette méthode dans Ember lui-même sont supprimées de la version `ember.prod.js` .

Lire Le débogage en ligne: https://riptutorial.com/fr/ember-js/topic/2320/le-debogage

Chapitre 10: Tâches asynchrones dans les composants

Remarques

Dans `ember-concurrency` le paramètre supplémentaire d' `error` est un travail de contournement pour éviter que les exceptions `onerror` ne se `onerror` les `onerror` d'Ember (car elles sont censées être gérées dans le modèle). Il y a une demande de fonctionnalité pour gérer cela mieux.

Examples

tâche de simultanéité des braises

Une communauté de facto alternative standard est un addon appelé ember-concurrency qui fait que la confusion disparaît.

Il peut être installé avec la commande `ember install ember-concurrency` .

Avantages

- Raisonnement intuitif de code asynchrone complexe.
- Offre une API complète pour la gestion des tâches.
- Peut être annulé
- Peut être utilisé directement dans un composant sans avoir besoin d'un objet proxy.
- Détruit les promesses dans la fonction de tâche.
- Peut utiliser les blocs JavaScript `try` / `catch` / `finally` pour gérer l'affectation asynchrone, les exceptions et le nettoyage.
- Les tâches sont automatiquement annulées sur l'événement `willDestroy` , évitant les erreurs lors de la définition des valeurs sur les objets détruits (par exemple, après une minuterie)

Les inconvénients

- Non intégré - nécessite `ember install ember-concurrency`
- Utilise des fonctions de générateur pouvant confondre les développeurs utilisés pour promettre des chaînes.

JavaScript

```
import Ember from 'ember';
import { task, timeout } from 'ember-concurrency';

const { Component, set } = Ember;
```

```
export default Component.extend({
  myTask: task(function * () {
    set(this, 'error', null);
    try {
      yield timeout(2000);
      return 'Foobar';
    } catch (err) {
      set(this, 'error', error);
    }
  }).keepLatest()
});
```

Modèle

```
{{#if myTask.isIdle}}
  <button onclick={{perform myTask}}>
    Start Task
  </button>
{{else}}
  Loading&hellip;
{{/if}}

{{#if myTask.last.value}}
  Done. {{myTask.last.value}}
{{/if}}

{{#if error}}
  Something went wrong. {{error}}
{{/if}}
```

PromiseProxyMixin

Ember est livré avec un assistant intégré qui fournira des propriétés calculées pour le statut d'une tâche asynchrone.

Avantages

- Intégré - pas besoin d'un addon.
- Peut être géré dans le cycle de vie d'un composant.
- Fournit des propriétés d'état pratiques pouvant piloter la logique du modèle.

Les inconvénients

- Doit être enveloppé dans un `Ember.Object` et ne peut pas être appliqué directement à un `Ember.Component` .
- Crée une déconnexion entre la chaîne de promesses d'origine et la destruction de la valeur du `content` .
- N'est pas très intuitive et peut être difficile à raisonner.
- Ne peut être annulé

JavaScript

```
import Ember from 'ember';

const {
  Component, PromiseProxyMixin, get, set, computed,
  isPresent, run, RSVP: { Promise }
} = Ember;

const MyPromiseProxy = Ember.Object.extend(PromiseProxyMixin);

export default Component({
  myProxy: computed('promise', {
    get() {
      const promise = get(this, 'promise');
      return isPresent(promise) ? MyPromiseProxy.create({promise}) : null;
    }
  }),

  actions: {
    performTask() {
      const fakeTask = new Promise((resolve) => {
        run.later(resolve, 'Foobar', 2000);
      });
      set(this, 'promise', fakeTask);
    }
  }
});
```

Modèle

```
{{#if myProxy.isPending}}
  Loading&hellip;
{{else}}
  <button onclick={{action "performTask"}}>
    Start Task
  </button>
{{/if}}

{{#if myProxy.isFulfilled}}
  Done. {{myProxy.content}}
{{/if}}

{{#if myProxy.isRejected}}
  Something went wrong. {{myProxy.reason}}
{{/if}}
```

Lire Tâches asynchrones dans les composants en ligne: https://riptutorial.com/fr/ember-js/topic/1054/taches-asynchrones-dans-les-composants

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec ember.js	Cameron, Community, Eric D. Johnson, Kenneth Larsen, locks, Nicos Karalis, ownsourcing dev training, Patsy Issa, Sid, Xinyang Li
2	Assistant de formatage de devise	Alan Mabry, Andrius
3	Comment importer une bibliothèque / plugin JavaScript	Rafalsonn
4	Comment mettre à jour Ember, Ember Data et Ember CLI	Cameron, ddoria921, heat, locks
5	Composant - communication entre enfant et composant parent.	Cameron, knownasilya, locks, Xinyang Li
6	Essai	Daniel Kmak
7	Format de date	Daniel Kmak, Silvio Langereis
8	Initialiser Foundation ou Bootstrap sur ember-cli de manière appropriée	Rafalsonn
9	Le débogage	nem035
10	Tâches asynchrones dans les composants	Sukima, user2708383, wdspkr