2025/10/21 06:23 1/7 Components

Components

Création

Avec ember-cli:

```
ember generate component my-component-name
```

Les composants doivent avoir au moins un tiret - dans leur nom. **blog-post** est un nom acceptable, **audio-player-controls** également, mais **post** ne l'est pas. Cette règle permet d'éviter les collisions de nom avec les éléments html existants ou futurs.

Exemple de Template de composant :

```
<article class="blog-post">
  <h1>{{title}}</h1>
  {{yield}}
  Edit title: {{input type="text" value=title}}
</article>
```

- title est une propriété du composant (qui sera initialisée à sa création
- **yield** est un helper permettant d'afficher le contenu du composant (ce qui se trouvera entre ses balises de fermeture et d'ouverture)

Utilisation du composant blog-post dans un template :

```
{{#each model as |post|}}
{{#blog-post title=post.title}}
{{post.body}}
{{/blog-post}}
{{/each}}
```

Le template index.hbs est alimenté par le model chargé dans la route correspondante :

```
import Route from '@ember/routing/route';

export default Route.extend({
   model() {
     return this.get('store').findAll('post');
   }
});
```

Appel dynamique de component

Le helper {{component}} peut être utilisé pour définir le composant à solliciter à l'exécution, tandis que la

syntaxe {{my-component}} vue précédemment fera toujours appel au même composant my-component.

Le premier paramètre du helper est le nom du composant à solliciter. La référence à {{component 'blog-post'}} revient à {{blog-post}}.

Exemple d'utilisation:

Soient les 2 composants suivantes :

```
<h3>Hello from foo!</h3>
{{post.body}}
```

```
<h3>Hello from bar!</h3>
<div>{{post.author}}</div>
```

La route correspondante chargeant le model :

```
import Route from '@ember/routing/route';

export default Route.extend({
   model() {
     return this.get('store').findAll('post');
   }
});
```

Le template sollicite foo-component ou bar-component en fonction de chaque post affiché :

```
{{#each model as |post|}}
{{!-- either foo-component or bar-component --}}
{{component post.componentName post=post}}
{{/each}}
```

Passage de paramètres à un composant

Un composant peut disposer de propriétés, initialisées à l'appel du composant dans le template appelant.

Le template du composant utilise les propriétés :

```
<article class="blog-post">
  <h1>{{title}}</h1>
  {{body}}
</article>
```

La route charge les données...

```
import Route from '@ember/routing/route';
```

2025/10/21 06:23 3/7 Components

```
export default Route.extend({
  model() {
    return this.get('store').findAll('post');
  }
});
```

Le template index sollicite le composant et lui passe **title** et **body**.

```
{{#each model as |post|}}
{{blog-post title=post.title body=post.body}}
{{/each}}
```

Contenu

Un composant peut disposer d'un contenu passé entre ses balises ouvrante et fermante :

L'appel du composant blog-post dans index.hbs est défini en utilisant ses balises ouvrante $\{\#blog-post\}\}$ et fermante $\{\{blog-post\}\}$

```
{{#blog-post title=title}}
  by {{author}}
  {{body}}
{{/blog-post}}
```

Le template correspondant insère le contenu passé entre les balise grâce au helper yield

```
<h1>{{title}}</h1>
<div class="body">{{yield}}</div>
```

Personnalisation des éléments HTML

Balise principale

Par défaut le code HTML généré par un composant l'est dans une div.

```
<div id="ember180" class="ember-view">
  <h1>My Component</h1>
</div>
```

Il suffit de modifier la propriété **tagName** du composant dans la classe dérivée associée pour modifier cette balise par défaut :

```
import Component from '@ember/component';
export default Component.extend({
  tagName: 'nav'
```

Last update: 2019/08/31 14:21

```
});
```

```
  {#link-to "home"}}Home{{/link-to}}
  {{#link-to "about"}}About{{/link-to}}
```

Personnalisation de la classe css

La classe appliquée au composant qui sera généré peut être définie à son invocation, en tant qu'attribut html :

```
{{navigation-bar class="primary"}}
```

Il est également possible de déterminer quelle classe css sera appliquée à un composant en affectant à sa propriété **classNames** un tableau de strings.

```
import Component from '@ember/component';
export default Component.extend({
  classNames: ['primary']
});
```

Il est également possible de déterminer la classe css à partir des propriétés du composant en utilisant classNameBindings.

Si une propriété booléenne est utilisée, la classe sera présente ou absente en fonction de la valeur de la propriété (false ou true) :

```
import Component from '@ember/component';
export default Component.extend({
  classNameBindings: ['isUrgent'],
  isUrgent: true
});
```

classNameBindings peut également porter une condition (si/alors/sinon) sur la valeur d'une propriété booléenne :

```
import Component from '@ember/component';
export default Component.extend({
  classNameBindings: ['isEnabled:enabled:disabled'],
  isEnabled: false
});
```

2025/10/21 06:23 5/7 Components

Personnalisation des attributs HTML

La propriété attributeBindings permet de définir les attributs du DOM associés au HTML généré :

```
import Component from '@ember/component';
export default Component.extend({
  tagName: 'a',
  attributeBindings: ['href'],
  href: 'http://emberjs.com'
});
```

Il est possible de définir **attributeBindings** à partir d'une propriété du composant :

```
import Component from '@ember/component';
export default Component.extend({
  tagName: 'a',
  attributeBindings: ['customHref:href'],
  customHref: 'http://emberjs.com'
});
```

Gestion des événements

```
{{#double-clickable}}
This is a double clickable area!
{{/double-clickable}}
```

Ajout de la prise en charge de l'événement double click avec le hook doubleClick :

```
import Component from '@ember/component';

export default Component.extend({
   doubleClick() {
      alert("DoubleClickableComponent was clicked!");
   }
});
```

Pour autoriser la propagation de l'événement sur les parents DOM du composant, la méthode évènementielle peut retourner **true**.

```
import Component from '@ember/component';
```

```
export default Component.extend({
  doubleClick() {
    alert("DoubleClickableComponent was clicked!");
    return true;
  }
});
```

Voir la liste des events hook possibles

Actions

Création d'un composant appelant l'action définie dans la route active :

Générer le composant :

```
ember generate component action-button
```

Implémenter le template du composant action-button :

```
<button {{action "doButtonThing"}} class="btn btn-default">{{yield}}</button>
```

Le template invoque l'action **doButtonThing** du composant qu'il va falloir créer dans le fichier **action-button.js** :

```
import Ember from 'ember';

export default Ember.Component.extend({
   actions: {
      doButtonThing() {
        this.sendAction('actionToCall');
      }
   }
});
```

Le composant peut maintenant être utilisé dans une route /test :

```
{{#action-button actionToCall="continue"}}Continuer vers test/new{{/action-button}}
```

l'attribut **actionToCall** fait référence à l'action **continue** qu'il nous faut maintenant ajouter dans le route handler :

```
import Route from '@ember/routing/route';
```

2025/10/21 06:23 7/7 Components

```
export default Route.extend({
  actions: {
    continue() {
       console.log("continuing on test/new page");
       this.replaceWith('test.new');
    }
}
```

From:

http://slamwiki2.kobject.net/ - SlamWiki 2.1

Permanent link:

http://slamwiki2.kobject.net/richclient/emberjs/components

Last update: **2019/08/31 14:21**

