Computed properties

Pour faire simple, les **computed properties** permettent de déclarer des fonctions en tant que propriétés. Une **computed property** est déclarée en tant que fonction, qu'Ember appellera automatiquement lorsque la propriété sera sollicitée.

Une **computed property** est utilisable de la même manière que toute autre propriété statique normale.

Déclaration

Création de la computed property **fullName** sur la classe personne :

```
import EmberObject, { computed } from '@ember/object';

Person = EmberObject.extend({
  // these will be supplied by `create`
  firstName: null,
  lastName: null,

  fullName: computed('firstName', 'lastName', function() {
    let firstName = this.get('firstName');
    let lastName = this.get('lastName');

    return `${firstName} ${lastName}`;
  })
});

let ironMan = Person.create({
  firstName: 'Tony',
  lastName:  'Stark'
});

ironMan.get('fullName'); // "Tony Stark"
```

Dans ce cas, la propriété **fullName** sera automatiquement mise à jour sur les changements de **firstName** ou **lastName**.

Affectation : setter

Il est possible de faire en sorte qu'une computed property soit modifiable :
Il suffit dans ce cas de définir sa méthode **set** :

```
import EmberObject, { computed } from '@ember/object';

Person = EmberObject.extend({
  firstName: null,
  lastName: null,
```

```
  fullName: computed('firstName', 'lastName', {
    get(key) {
      return `${this.get('firstName')} ${this.get('lastName')}`;
    },
    set(key, value) {
      let [firstName, lastName] = value.split(/\s+/);
      this.set('firstName', firstName);
      this.set('lastName',  lastName);
      return value;
    }
  })
});


let captainAmerica = Person.create();
captainAmerica.set('fullName', 'William Burnside');
captainAmerica.get('firstName'); // William
captainAmerica.get('lastName'); // Burnside
```

Macros

Certain types de computed properties sont très communs et disposent de macros permettant d'éviter leur réécriture.

Le cas suivant illustre la computed property macro **equals** :

```
import EmberObject, { computed } from '@ember/object';
import { equal } from '@ember/object/computed';

Person = EmberObject.extend({
  fullName: 'Tony Stark',

  isIronManLongWay: computed('fullName', function() {
    return this.get('fullName') === 'Tony Stark';
  }),

  isIronManShortWay: equal('fullName', 'Tony Stark')
});
```

Macro	**Exemple**	**Rôle**
and	`Ember.computed.and('hasWalkingStick', 'hasBackpack')`	retourne vrai si hasWalkingStick et hasBackpack sont vrais
bool	`Ember.computed.bool('numBananas')`	retourne vrai si numBananas est différent de null et >0
collect	`Ember.computed.collect('hat', 'shirt')`	retourne un tableau des valeurs des propriétés
empty	`Ember.computed.empty('todos')`	retourne vrai si les propriétés passées sont vides (null, chaîne vide, tableau vide...)
equal	`Ember.computed.equal('percentCarrotsEaten', 100)`	retourne vrai si percentCarrotEaten est égal à 100

Macro	Exemple	Rôle
filter	`let Hamster = Ember.Object.extend({ remainingChores: Ember.computed.filter('chores.@each.done', function(chore, index, array) { return !chore.get('done'); }) });`	retourne la liste des éléments correspondant au critère
filterBy	`let Hamster = Ember.Object.extend({ remainingChores: Ember.computed.filterBy('chores', 'done', false) });`	retourne la liste des éléments correspondant au critère