Introduction to Backbone.js model

Welcome to lesson 4 of Getting started with Backbone.js course! In this lesson we are going to discuss two very important concepts of Backbone.js: **models** and **collections**. In this step I will introduce models to you.

Models contain data for an application as well as the surrounding logic like validations, default values, computed properties and so on. Model represent a concept, a blueprint of an item.

Creating a model

To define a model, use the extending mechanism you've seen earlier. Create an *event.js* file inside the *models* directory and include it into the app:

```
<script src="js/models/event.js"></script>
```

Now define a model and attach it to the global object:

```
Organizer.Event = Backbone.Model.extend({

});
```

Open the console, reload the page and instantiate the model:

```
var event = new Organizer.Event();
```

You can now turn this to a JavaScript object using `toJSON` method:

```
event.toJSON();
```

Of course, an empty object will be returned because our model is currently empty. Provide some data:

```
var event = new Organizer.Event({title: 'test'});
event.toJSON();
```

If you want to give a default values to your model, that can be done easily:

```
Organizer.Event = Backbone.Model.extend({
  defaults: {
    title: ''
  }
});
```

How to get a model's attribute

How to get value of a model's attribute? You might think that using something like

```
event.title
```

will do the trick but it won't. You cannot access model's attributes directly – use `get` and `set` methods instead:

```
event.get('title');
event.set('title', 'another test');
```

To get a raw list of attributes, use:

```
event.attributes
```

This way you can access attributes directly:

```
event.attributes.title
```

If you want to return escaped value of an attribute, use `escape` method instead of `get` Instantiate another model:

```
var unsafe = new Organizer.Event({body: '<script>alert("hacked!");</script>'})
unsafe.get('body');
```

So if you render this `body` directly into the view, that will execute a script. This is a big security hole because this way hacker may inject, for example, key logger onto your page. Therefore render escaped version instead:

```
unsafe.escape('body');
```

Some model's methods

To check if the model's attribute is not null and is defined, use `has` method:

```
var event2 = new Organizer.Event({title: 'defined title'});
event2.has('title');
event2.has('body');
```

You may remove any attribute of a model using `unset`:

```
event2.unset('title');
event2.has('title');
```

If you wish to clear all attributes at once, use `clear`:

```
event2.set('title', 'test');
event2.clear();
event2.has('title');
```

Models ID and CID

The last topic to discuss in this step is model's **id and cId**.

Id is the unique identifier of a model that is typically being assigned by your server when the model is saved. This id may have various formats, but in any case it has to be unique. Often id is a primary key of a table and has an index assigned. If for some reason you do not have id attribute on your model, you may use any other attribute as a unique identifier by overriding `idAttribute` property for a model.

Cid (or client id) is an identifier that is being assigned to the model by Backbone upon its creation. Every model has its own cid and unlike other attributes you can access it directly:

```
event2.cid
var event3 = new Organizer.Event();
event3.cid
```

Cid is useful when you want to render a model that is yet not saved and therefore does not have an id assigned.