# Advanced JavaScript Practical Exam: Halls Events

The goal of this sample exam is to practice the creation of a JavaScript structure, using the best practices of **quality JavaScript**.

### The Halls Events system

**The Halls Events system** is a system that manages events in different halls. Halls Events system should have Halls, Events, Lectures, Parties, Courses, Employess and Trainers.

### The Source Code

You are given the following components:

* **index.html** - The HTML file
* **components folder** - The libraries of application;
* **helpers folder** – Additional helper files to implement (if needed);
* **models folder** - The modules you need to implement;
* **generator.js** – Generates test data;
* **app.js** – Fills the tables with data;

Your task is to implement only the missing **models**. If you implement them correctly, your application will run without errors. You **do not need** to edit the **html/css** and **app.js** files.

## Implement function constructors or objects for all types

You can use Pseudo-classical or Prototypal OOP. All **modules** should have **visible (not hidden) getter** and **setters** (functions that set a certain value to a property with validation or throws an error) for all hidden (with underscore) properties.

1. Event – Event is an **abstract** (no initialization should be possible with this module) module and should have:

Constructor:

* + options parameter – The constructor should have only 1 parameter that is called options that will hold all the information needed to instantiate the module. E.g.

function Person(options) {  
 this.\_name = options.name;  
 this.\_age = options.age;  
}

Properties:

* + \_title – A string (only letters and whitespace)
  + \_type – A string (only letters and whitespace)
  + \_duration – A number (only digits)
  + \_date – A Date object (only Date instances)

1. Lecture – Lecture is module, **derived** from **Event** (extends Event) and should have:

Constructor:

* + options parameter – The constructor should have only 1 parameter that is called options that will hold all the information needed to instantiate the module. For example see **Event**.

Properties:

* + All parent properties
  + **\_trainer** – A Trainer object (only Trainer instances)
  + **\_course** – A Course object (only Course instances)

1. Party – Party is a module, **derived** from **Event** (extends Event) and should have:

Constructor:

* + options parameter – The constructor should have only 1 parameter that is called options that will hold all the information needed to instantiate the module. For example see **Event**.

Properties:

* + All parent properties
  + \_isCatered– A boolean (converts truthy values to True and falsy to False)
  + \_isBirthday– A boolean (converts truthy values to True and falsy to False)
  + \_organiser– A Employee object (only Employee instances)

1. Course – Course is a module that should have:

Properties:

* + \_name – A string (only letters and whitespace)
  + \_numberOfLectures– A number (only digits)

1. Hall – Hall is a module that should have:

Properties:

* + \_name – A string (only letters and whitespace)
  + \_**capacity** – A number (only digits)
  + **parties** – A visible empty array (filled with Party instances only)
  + **lectures** – A visible empty array(filled with Lecture instances only)

Methods:

* + addEvent – Adds a new event (Party/Lecture) to the corresponding property (**parties** or **lectures)**. The method checks whether the event is a Party or a Lecture and stores it in the respecting property.

1. **Employee –** Employee is a module that should have:

Properties:

* + \_name **–** A string (only letters and whitespace)
  + \_workHours– A number (only digits)

1. **Trainer –** Trainer is a module, **derived** from **Employee** (extends Employee) and should have:

Properties:

* + All parent properties
  + **courses** – A visible empty array (filled with Course instances only)
  + **feedbacks** –A visible empty array (filled with strings only)

Methods:

* + addFeedback – adds a new feedback(string) to the property **feedbacks** and validates that the given value is a string
  + **addCourse –** adds a new Course to the property **courses** and validates that the given value is a Course instance

**All hidden module properties should have visible (not hidden) methods – getProperty and setProperty (validates the given value) attached to the modules prototype!**

Separate all models in different files. Expose constructors for creating instances of current model in the global object “**app**”.

You can see how the objects are created in the generator.js file for example.

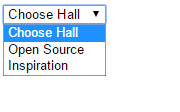
84 + 16 (unit tests) points

## Bonus: Implement showing halls

In the app.js file there is a TODO where you need to dynamically create the HTML option tags for the HTML select tag containing all halls. You are allowed to use only Native JS (no external libraries such as jQuery). Think of a way to optimize the appending of elements.

**Keep in mind that if you do not do this part you will not be able to visualize the test data given with the skeleton project!**

Bonus 5 points



If you implement correctly all types, the code will work and will show something like this in the browser:

