# **Краткое описание Facade**

Шаблон фасад (англ. Facade) — структурный шаблон проектирования, позволяющий скрыть сложность системы путем сведения всех возможных внешних вызовов к одному объекту, делегирующему их соответствующим объектам системы.

**Facade**

Как правило, фасад используется для создания некоторой абстракции, скрывающей за собой совершенно иную реальность. Паттерн «фасад» обеспечивает удобный высокоуровневый интерфейс для больших блоков кода, скрывая за собой их истинную сложность. Относитесь к фасаду, как к упрощенному API, который вы отдаете в пользование другим разработчикам.

Фасад — структурный паттерн. Часто его можно обнаружить в JavaScript-библиотеках и фреймворках, где пользователям доступен только фасад — ограниченная абстракция широкого диапазона поведений реализованных внутри.

Благодаря такому подходу, пользователь взаимодействует только с интерфейсом, не имея никакого представления о подсистемах, которые скрываются за ним.

Причина, по которой нам интересен фасад — возможность скрыть детали реализации конкретной функциональности, хранящиеся в модулях. Это позволит нам вносить изменения в реализацию, не сообщая об этом пользователям.

Надежный фасад — наш упрощенный интерфейс — позволит нам не беспокоиться о тесных связях некоторых модулей нашей системы с dojo, jQuery, YUI, zepto или какой-либо другой библиотекой. Это становится не так важно. Вы можете переходить с одной библиотеки на другую не меняя слой взаимодействия. К примеру, с jQuery на dojo. Более того, у вас появляется возможность совершить такой переход на более поздних этапах, без изменений в остальных частях системы.

Ниже я написал достаточно простой пример использования фасада. Как вы видите, у нашего модуля есть несколько приватных методов. Чтобы создать более простой интерфейс для доступа к этим методам мы используем фасад.

var module = (function() {

var \_private = {

i: 5,

get: function() {

console.log('Текущее значение:' + this.i);

},

set: function(val) {

this.i = val;

},

run: function() {

console.log('процесс запущен');

},

jump: function() {

console.log('резкое изменение');

}

};

return {

facade: function(args) {

\_private.set(args.val);

\_private.get();

if (args.run) {

\_private.run();

}

}

}

}());

module.facade({run:true, val:10}); // Текущее значение: 10, процесс запущен

Это и есть та причина, по которой мы добавили фасад к нашей архитектуре. В следующей главе мы обсудим медиатор. Принципиальное различие между этими двумя паттернами заключается в том, что фасад, как структурный паттерн, всего лишь передает существующую функциональность в медиатор, в то время как медиатор, как поведенческий паттерн, может эту функциональность расширять.

**Источники**

http://largescalejs.ru/the-facade-pattern/

<https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D1%81%D0%B0%D0%B4_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)>

http://www.dofactory.com/javascript/facade-design-pattern