Паттерн Модуль - оборачивание приватных и публичных методов в 1 сущность, с предоставлением возможности доступа из вне только к публичным методам.

Помогает избежать конфликта имен, делает код более чистым

Calc = (function () {

var pub = {};

var priv\_func = func () {}

pub.add = function(){}; - public

pub.sub = function(){

priv\_func () - private

}; - public

return pub;

})();

Можно импортировать параметры в модуль

Calc = (function ($) {

var pub = {};

var priv\_func = func () {}

pub.add = function(){}; - public

pub.sub = function(){

priv\_func () - private

}; - public

$.each ();..

return pub;

})(jQuery);

Можно разделять модуль между файлами

Файл 1

Calc = (function ($, pub) {

var pub = {};

var priv\_func = func () {}

pub.add = function(){}; - public

pub.sub = function(){

priv\_func () - private

}; - public

$.each ();..

return pub;

})(jQuery , Calc || {});

Файл 2

Calc = (function ($, pub) {

var pub = {};

var priv\_func = func () {}

pub.add = function(){}; - public

pub.sub = function(){

priv\_func () - private

}; - public

$.each ();..

return pub;

})(jQuery , Calc || {});

можно вместо return pub писать

return {

add : add,

sub : sub

}

В js паттерн модуль служит эмуляцией класса

Инкапсуляция приватных методов реализуется с помощью замыкания

Документация - документ, описывающий как пользоваться программным продуктом, содержит данные, необходимые для разработки, эксплуатации, сопровождения программы или программного средства.

Типы документаций:

Требования - положения, которые определяют характеристики, свойства, качества системы. Это то, что должно быть разработано, или уже было разработано

Архитектура проекта - обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО

Проектная документация обычно описывает продукт в общих чертах. Не описывая того, как что-либо будет использоваться, она скорее отвечает на вопрос «почему именно так». Например, в проектном документе программист может описать обоснование того, почему структуры данных организованы именно таким образом. Описываются причины, почему какой-либо класс сконструирован определённым образом, выделяются паттерны, в некоторых случаях даже даются идеи как можно будет выполнить улучшения в дальнейшем. Ничего из этого не входит в техническую или пользовательскую документацию, но всё это действительно важно для проекта.

Техническая - документация на код, алгоритмы, интерфейсы, API

Они получают информацию из специальным образом оформленных комментариев в исходном коде,

пользовательская — руководства для конечных пользователей, администраторов системы

маркетинговая - как продавать продукт и анализировать спрос на него