前端模块化

大纲：
- 模块化的历史
- 模块化技术方案介绍
- 手写实现

模块化的历史

> 1. 什么是模块？

实实在在的需求推进了所有技术的演进，模块化也是。

站在前端发展的上帝视角来看，随着前端的能力在纵深都得到增强之后，迫切的需要更好的代码管理、组织、通信的模式，各种模块化的技术方案开始出现。

模块已经成为了代码管理/编译，业务分离的基本单元。

本质上模块就是一种提供对外通信接口，进行代码切分/组合的管理方式。其呈现的方式因不同的模块化方案而不同，基本是以文件粒度区分。

> 2. 为什么要用模块化？

把复杂问题分解成多个子问题
  - 关注分离

大型软件开发的技术基础
  - 更优雅的代码管理
  - 替换、复用、拓展
  - 内聚（变量，行为内聚在模块内，对外暴露接口进行通信）

开发方式的革新
  - 方便多人协同，面向过程开发

> 3. 模块化发展历史

- 3.1 函数时代

模块化混沌时期，组织代码就是靠「经验」，基本就是全局函数各种调用

> 有什么缺点？

```js
```

```
function fn () {};
```

- 3.2 命名空间

模块化思想雏形初现，通过简单的命名空间进行「块儿」的切分，更分离，和内聚

弥补了全局函数的一些缺点，业界著名案例 「YUI2」

> 有什么缺点？

```js
var student = {
  name: 'tom',
  getScore: function() {}
}

student.name;
student.getScore();
```

- 3.3 巧用闭包

模块化的解决方案再次提升，利用闭包使得污染的问题得到解决，更加纯粹的内聚。

> 有什么缺点？

```js
// moduleA.js
(function(global) {
  var name = 'tom';

  function getScore() {};

  global.moduleA = { name, getScrore };
})(window)
```

怎么解决依赖的问题

```js
// moduleA.js
(function (global, $) {
  // ...
})(window, Jquery)
```

基于 IIFE 还有很多变种的玩儿法，适合于各种情况

以上可以说就是现代模块机制的雏形或者说基石

现代模块化机制

上面我们简单的回顾了过去在模块化机制上的一些尝试和解决思路，为现代化模块机制打下了基础。

总的来说，现代模块化机制要解决的问题如下：

1. 命名污染，全局污染，变量冲突等基础问题
2. 内聚且私有，变量不能被外界污染到
3. 怎么引入（依赖）其它模块，怎样暴露出接口给其它模块
4. 最烦人的是依赖顺序问题，还记得以前的 Jquery 问题嘛
5. 上述第三点可能导致的循环引用问题，等边界情况

大家的学习目标旨在有一定的了解就行，接下来开始跟着我一起玩儿一下主流的几种模块化解决方案。
通过实际的例子去体会一下，到底他们是如何解决这类问题的。