# 动态数据绑定（一）

## 任务目的

1. 了解 getter 和 setter
2. 了解 new

## 任务描述

动态数据绑定是 Vue 最为基础，最为有用的一个功能。

给定任意一个对象，如何监听其属性的读取与变化？也就是说，如何知道程序访问了对象的哪个属性，又改变了哪个属性？

请实现这样的一个 Observer，要求如下：

1. 传入参数只考虑对象，不考虑数组。
2. new Observer返回一个对象，其 data 属性要能够访问到传递进去的对象。
3. 通过 data 访问属性和设置属性的时候，均能打印出右侧对应的信息。

# 动态数据绑定（二）

## 任务目的

1. 在实践中使用递归思想
2. 了解设计模式中的“发布-订阅模式”

## 任务描述

1. 如果传入参数对象是一个“比较深”的对象（也就是其属性值也可能是对象），那该怎么办呢？
2. 如果设置新的值是一个对象的话，新设置的对象的属性是否能能继续响应 getter 和 setter。
3. 考虑传递回调函数。在实际应用中，当特定数据发生改变的时候，我们是希望做一些特定的事情的，而不是每一次都只能打印出一些信息。所以，我们需要支持传入回调函数的功能。

# 动态数据绑定（三）

## 任务目的

* 了解事件传播机制

## 任务描述

在第二题的基础上，我们再多考虑一个问题："深层次数据变化如何逐层往上传播"

let app2 = new Observer({

name: {

firstName: 'shaofeng',

lastName: 'liang'

},

age: 25

});

app2.$watch('name', function (newName) {

console.log('我的姓名发生了变化，可能是姓氏变了，也可能是名字变了。')

});

app2.data.name.firstName = 'hahaha';

// 输出：我的姓名发生了变化，可能是姓氏变了，也可能是名字变了。

app2.data.name.lastName = 'blablabla';

// 输出：我的姓名发生了变化，可能是姓氏变了，也可能是名字变了。

观察到了吗？firstName 和 lastName 作为 name 的属性，其中任意一个发生变化，都会得出以下结论："name 发生了变化。"这种机制符合”事件传播“机制，方向是从底层往上逐层传播到顶层。  
这现象想必你们也见过，比如：“点击某一个DOM元素，相当于也其父元素和其所有祖先元素。”（当然，你可以手动禁止事件传播） 所以，这里的本质是："浏览器内部实现了一个事件传播的机制"。

# 动态数据绑定（四）

## 任务目的

1. 熟练使用原生 JS对操作 DOM 结构

## 任务描述

有了前面的充分准备，相信你能搞定这一题。**请实现如下的这样一个 Vue，传入参数是一个 Selector 和一个数据对象，程序需要将 HTML 模板片段渲染成正确的模样。** 这就是一次性的静态数据绑定。

PS：此题尚未要求实现动态数据绑定。

# 动态数据绑定（五）

## 任务目的

* 综合应用本系列的所有知识点

## 任务描述

在第四题的基础上，我们考虑如何做到："**当数据发生改变时，重新渲染 DOM。**" 此目标又可以分为两层难度。

1. 相对简单的目标：数据任意部分发生了变化，都重新渲染 DOM。
2. 相对困难的目标：数据的某一部分发生了变动，只重新渲染对应这部分数据的 DOM。