# HTML5新特性 -- Unit05

# 1.Vuex

Vuex的核心是Store（仓库），其就是一个容器，包含了Vue应用中的状态。其基本结构是：

export default new Vuex.Store({   
 state: { //物品  
 },   
 getters:{ //记账员  
 },   
 mutations: { //理货员  
 },   
 actions: { //货车  
 },   
 modules: {   
 }   
})

## ·

## Actions

**Actions**用于**异步发送请求**，示例代码如下：

actions: {   
 get\_data\_action(){   
 axios.get('http://127.0.0.1:3000/data').then(res=>{

//将服务器返回的数据，添加到state的products数组中去

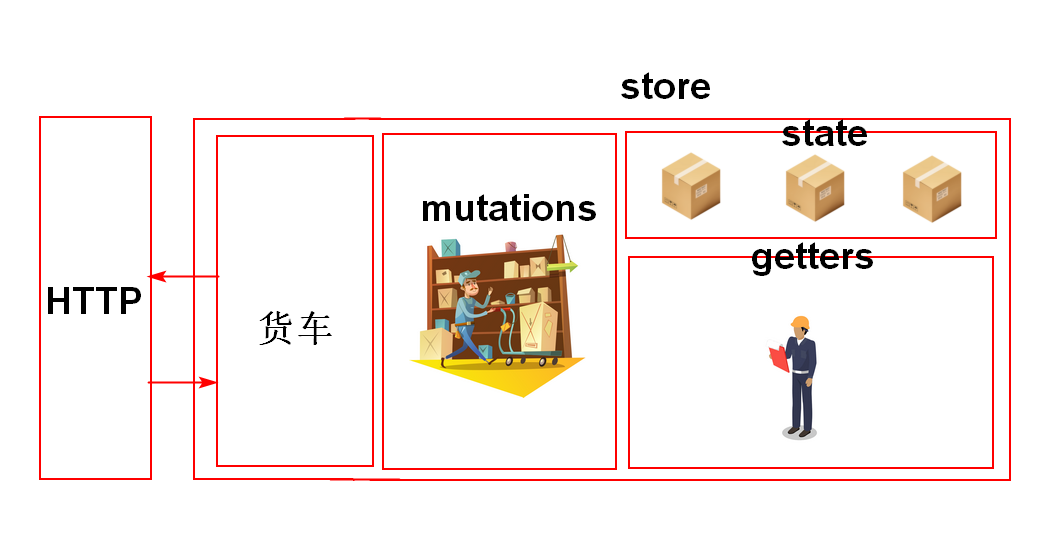
//因为actions不能直接操作state，只能通过mutations操作state

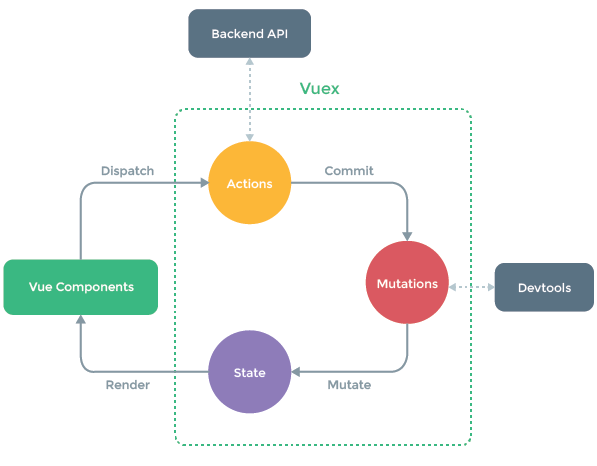
**context.commit**('add\_product\_mutation',res.data)

});   
 }   
}

## · 调用Actions的方法

this**.$store.dispatch("方法名称"[,payload])**





# 2.WebStorage

Webstorage提供了一种比cookie更加**直观的数据存储方式**，其以**名/值对**的形式进行**存储**。其分为**两种存储机制**：

**sessionStorage,**仅在**浏览器窗口打开期间有效。**

**localStorage**,其**永远有效**。

sessionStorage和localStorage有相同的属性和方法

· length属性

**length**属性用于**获取项目的数量**，其语法结构是：

**sessionStorage.length**   
   
**localStorage.length**

· setItem()方法

**setItem()**方法用于**设置存储项目**，其语法结构是：

**sessionStorage.setItem(key,value)**   
   
**localStorage.setItem(key,value)**

· getItem()方法

**getItem()**用于**获取项目的值**，其语法结构是：

**variable = sessionStorage.getItem(key)**   
**variable = localStorage.getItem(key)**

· removeItem()方法

**removeItem()**方法用于**删除项目**，其语法结构是：

**sessionStorage.removeItem(key)**   
   
**localStorage.removeItem(key)**

· clear()方法

**clear()**方法用于**删除所有项目**，其语法结构是：

**sessionStorage.clear()**

**localStorage.clear()**

# 

# 3.拖放

拖放即将**源对象从一个位置拖动到另一个位置**，在HTML5 中任何元素都可以实现拖放，但为保证浏览器的兼容性，建议为被**拖放的对象添加draggable="true"**属性。

## 

## 3.1 拖放事件

### 

### 对源对象的事件（3个）

### · dragstart事件

**dragstart**事件在**源对象开始被拖放时触发**，其语法结构是：

**HTMLElement.addEventListener('dragstart',()**=>**{**   
 //...   
});   
 **或者**

**HTMLElement.ondragstart = ()**=>**{** //...   
}

### ·

### drag事件

**drag**事件在**源对象拖放过程中被触发**，其语法结构是：

**HTMLElement.addEventListener('drag',()**=>**{** //...   
});   
 **或者**

**HTMLElement.ondrag = ()**=>**{** //...   
}

### 

### · dragend事件

dragend事件在**源对象释放后被触发**(也可能在目标区域内，也可能在目标区域外)，其语法结构是：

HTMLElement.addEventListener('dragend',()=>{   
 //...   
});   
 **或者**

HTMLElement.ondragend = ()=>{   
 //...   
}

### 对目标对象的事件（4个）

### · dragenter事件

dragenter事件是在源对象**进入**目标对象的区域范围时被触发，其语法结构是：

HTMLElement.addEventListener('dragenter',()=>{   
 //...   
});   
 **或者**

HTMLElement.ondragenter = ()=>{   
 //...   
}

### 

### · dragover事件

**dragover**事件在源对象在目标对象**悬停**时触发，其语法结构是：

HTMLElement.addEventListener('dragover',(event)=>{   
  **event.preventDefault();**   
 //...   
});   
 **或者**

HTMLElement.ondragover = (event)=>{   
 **event.preventDefault();** //...   
}

**必须在该事件内调用event.preventDefault()方法，**否则不会触发drop事件

### 

### · dragleave事件

dragleave事件在**源对象拖动离开目标对象时触发**，其语法结构是：

**HTMLElement.addEventListener('dragleave',()**=>**{   
 //...**});   
 **或者**

HTMLElement.ondragleave = ()=>{   
 //...   
}

### 

### · drop事件

drop事件将在源对象在目标对象范围内**释放**时触发，其语法结构是：

HTMLElement.addEventListener('drop',()=>{   
 //...   
});   
 **或者**

HTMLElement.ondrop = ()=>{   
 //...   
}

拖放事件的触发顺序：

dragstart --> drag --> dragenter-->dragover--> drop-->dragend

### 源对象开始被拖放🡪源对象拖放中🡪进入目标对象🡪在目标对象中悬停🡪在目标对象中释放🡪源对象拖放完毕

### 3.2 dataTransfer属性

**拖放事件**的**dataTransfer属性**将**返回DataTransfer 对象**，用于**保存拖放数据**，其语法结构是：

**DataTransfer DragEvent.dataTransfer**

**DragEvent代表拖放事件对象**

## 3.3 DataTransfer对象

### · setData()方法

**setData()**方法用于**给指定的类型设置数据**，如果类型**不存在则自动添加到到未尾**，如果**存在的话，则替换原来的值**，其语法结构是：

**DataTransfer.setData(type,value)**

### 

### · getData()方法

**getData()**方法用于**获取指定类型的数据**，其语法结构是：

**variable = DataTransfer.getData(type)**