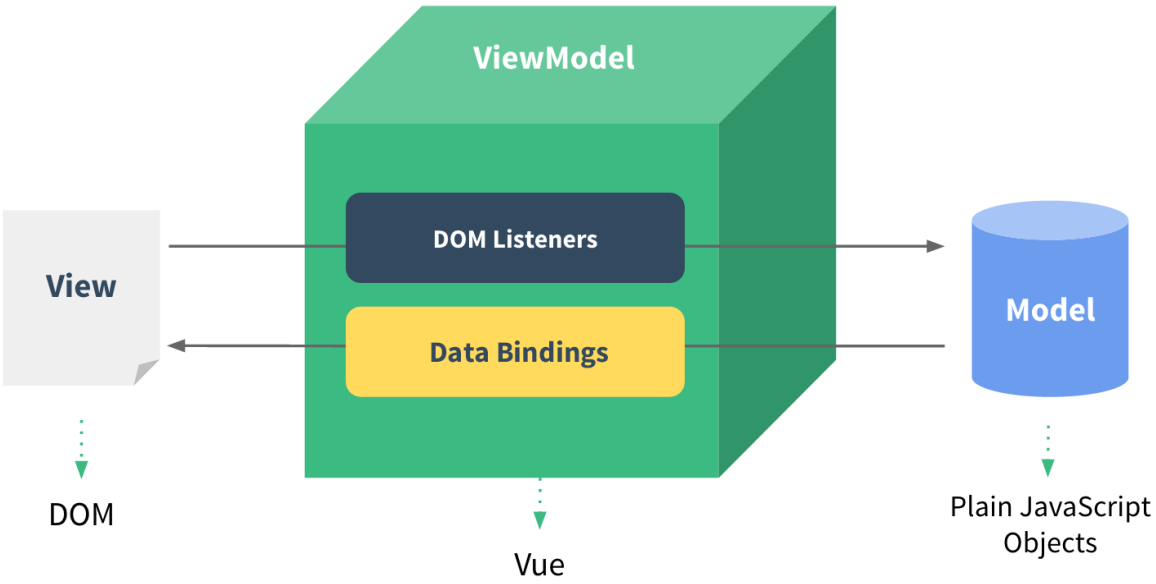
一、什么是 Vue

1.简介

Vue (读音 /vjuː/，类似于 view) 是一套用于构建用户界面的**渐进式的js框架**，发布于 2014 年 2 月。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。**Vue 的核心库只关注视图层**，不仅易于上手，还便于与第三方库（如：`vue-router`，`vue-resource`，`vuex`）或既有项目整合。 MVC

2.MVVM 模式的实现者——双向数据绑定模式

- Model：模型层，在这里表示 JavaScript 对象
- View：视图层，在这里表示 DOM（HTML 操作的元素）
- ViewModel：连接视图和数据的中间件，**Vue.js 就是 MVVM 中的 ViewModel 层的实现者**



在 MVVM 架构中，是不允许 **数据** 和 **视图** 直接通信的，只能通过 `ViewModel` 来通信，而 ViewModel 就是定义了一个 `Observer` 观察者

- ViewModel 能够观察到数据的变化，并对视图对应的内容进行更新
- ViewModel 能够监听到视图的变化，并能够通知数据发生改变

至此，我们就明白了，Vue.js 就是一个 MVVM 的实现者，他的核心就是实现了 `DOM 监听` 与 `数据绑定`

3.其它 MVVM 实现者

- AngularJS
- ReactJS
- 微信小程序

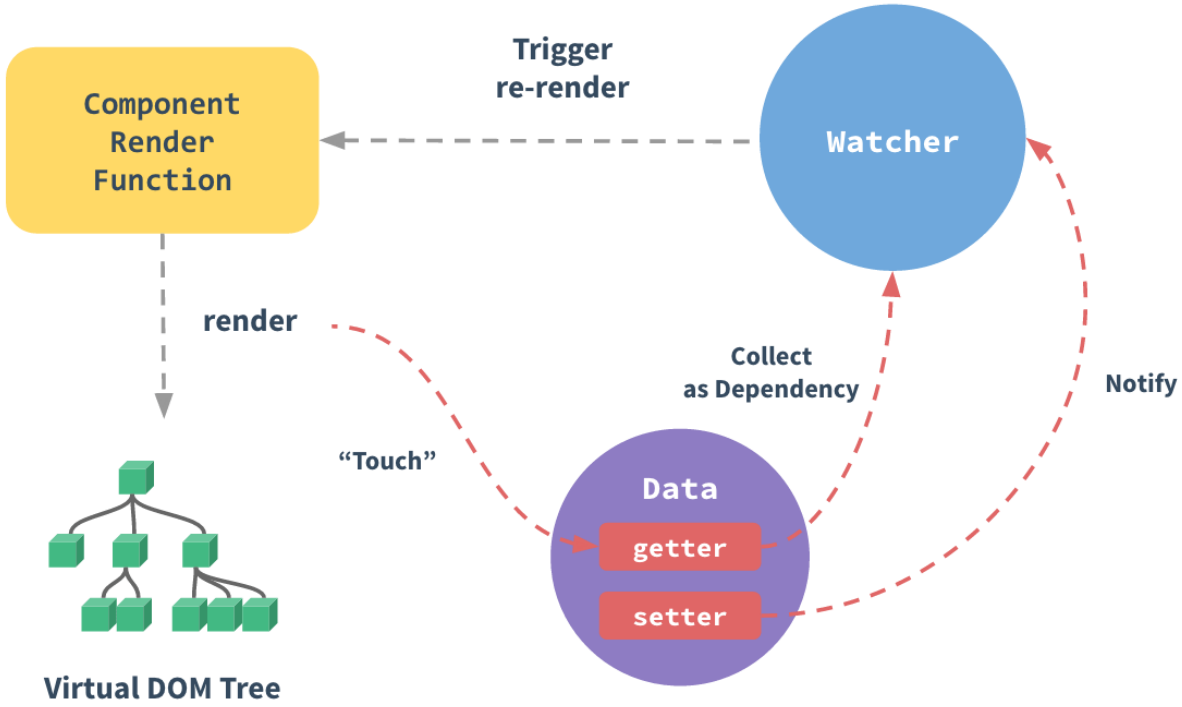
4.为什么要使用 Vue.js

- 轻量级，体积小是一个重要指标。Vue.js 压缩后有只有 **20多kb** （Angular 压缩后 **56kb+**，React 压缩后 **44kb+**）

- 移动优先。更适合移动端，比如移动端的 Touch 事件
- 易上手，学习曲线平稳，文档齐全
- 吸取了 Angular（`模块化`）和 React（`虚拟 DOM`）的长处，并拥有自己独特的功能，如：`计算属性`
- 开源，社区活跃度高

5.Vue.js 的两大核心要素

1）数据驱动



当你把一个普通的 JavaScript 对象传给 Vue 实例的 `data` 选项，Vue 将遍历此对象所有的属性，并使用 `Object.defineProperty` 把这些属性全部转为 `getter/setter`。**Object.defineProperty 是 ES5 中一个无法 shim 的特性，这也就是为什么 Vue 不支持 IE8 以及更低版本浏览器。**

这些 getter/setter 对用户来说是不可见的，但是在内部它们让 Vue 追踪依赖，在属性被访问和修改时通知变化。这里需要注意的问题是浏览器控制台在打印数据对象时 getter/setter 的格式化并不同，所以你可能需要安装 `vue-devtools` 来获取更加友好的检查接口。

每个组件实例都有相应的 **watcher** 实例对象，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的 `setter` 被调用时，会通知 `watcher` 重新计算，从而致使它关联的组件得以更新。

2）组件化

- 页面上每个独立的可交互的区域视为一个组件
- 每个组件对应一个工程目录，组件所需的各种资源在这个目录下就近维护
- 页面不过是组件的容器，组件可以嵌套自由组合（复用）形成完整的页面

二、快速开始

- 1.在页面引入vue的js文件即可。

注意：cdn是一种加速策略，能够快速的提供js文件

```
<script src="https://cdn.bootcss.com/vue/2.5.17-beta.0/vue.min.js"></script>
```

- 2.在页面中绑定vue元素

```
创建一个div，id是app
<div id="app"></div>
```

- 3.创建vue对象，设计对象的内容

其中该vue对象，绑定了页面中id是app的那个div

```
<script>
        new Vue({
            el:"#app",
            data:{
                title:"hello vue!",
                  args1:"hi!",
                 age:18,
                flag:true
            }
        });
</script>
```

- 4.在页面的元素中使用插值表达式来使用vue对象中的内容

```
<div id="app">
    {{ title }}
</div>
```

三、 插值表达式

插值表达式的作用是在View中获得Model中的内容

1.插值表达式

```
<div id="app">
            {{title}}
            {{[1,2,3,4][2]}}
            {{ {"name":"xiaoyu","age":20}.age }}
            {{ sayHello()}}
```

```
new Vue({
                el:"#app",
                data:{
                    title:"hello world!"
                },
                methods:{
                    sayHello:function(){
                        return "hello vue";
                    }
                }
            });
```

2.MVVM双向数据绑定：v-model

```
<div id="app">
    <input type="text" v-model="title" />
</div>
```

3.事件绑定: v-on

```
<input type="text" v-on:input="changeTitle" />
```

v-on叫绑定事件，事件是input，响应行为是changeTitle。也就是说，当input元素发生输入事件时，就会调用vue里定义的changeTitle方法

```
new Vue({
                el:"#app",
                data:{
                    title:"hello world!"
                },
                methods:{
                    sayHello:function(){
                        return "hello vue";
                    },
                    changeTitle:function(){
                        console.log("ct");//往日志里写
                    }
                }
            });
```

event.target.value == 当前事件的对象（input元素）的value值 注意：此时的this指的是当前vue对象。

所以：如果在method里要想使用当前vue对象中的data里的内容，必须加上this.

```
changeTitle:function(event){
    this.title = event.target.value;
}
```

4.事件绑定简化版：使用@替换v-on:

```
<input type="text" @input="changeTitle" />
```

5.属性绑定： v-bind

html里的所有属性，都不能使用插值表达式

```
<a href="{{link}}">baidu</a>

new Vue({
    el:"#app",
    data:{
        title:"hello world!",
        link:"http://www.baidu.com"
    },
    ...
```

上面的这种做法是错误的，可以使用绑定属性绑定来解决：

要想在html的元素中的属性使用vue对象中的内容，那么得用v-bind进行属性绑定

```
<a v-bind:href="link">baidu</a>
可以缩写成 冒号
<a :href="link">baidu</a>
```

6.v-once指令

指明此元素的数据只出现一次，数据内容的修改不影响此元素

```
<p v-once>{{titile}}</p>
```

7.v-html

就好比是innerHTML

```
<p v-html="finishedlink"></p>
```

```
new Vue({
        el:"#app",
        data:{
            title:"hello world!",
            link:"http://www.baidu.com",
            finishedlink:"<a href='http://www.baidu.com'>百度</a>"
                },
        ...
```

8.v-text

纯文本输出内容

```
<p v-text="finishedlink"></p>
```

四、事件

1.事件绑定范例

- 范例一：

```
<div id="app">
            <button type="button" v-on:click="increase">click</button>
            <p>
                {{counter}}
            </p>
```

```
new Vue({
        el:"#app",
        data:{
            counter:0
        },
        methods:{
            increase:function(){
                this.counter++;
        },
        ...
```

- 范例二：

```
<p v-on:mousemove="mo">mooooooo</p>
```

```
mo:function(event){
        console.log(event);
}
```

- 范例三：

```
<p v-on:mousemove="mo">
        mx:{{x}}
        my:{{y}}
</p>
```

```
new Vue({
        el:"#app",
        data:{
            counter:0,
            x:0,
            y:0,
        },
        methods:{
            increase:function(){
                this.counter++;
            },
            mo:function(event){
                this.x = event.clientX,
                this.y = event.clientY
            }
        }
});
```

2.参数传递

```
<button type="button" v-on:click="increase(2)">click</button>
```

```
            ...
            methods:{

                    increase:function(step){

                        this.counter+=step;

                    },
            ...
```

传多个参数:

```
<button type="button" v-on:click="increase(2,event)">click</button>
```

```
            ...
            methods:{
                    increase:function(step,event){
                        this.counter+=step;
            },

            ...
```

3.停止鼠标事件

```
<p v-on:mousemove="mo">
    mx:{{x}}
    my:{{y}}
    ---<span v-on:mousemove="dummy">停止鼠标事件</span>
</p>
```

```
dummy:function(event){
        event.stopPropagation();
}
```

另一种方式：

```
<span v-on:mousemove.stop>停止鼠标事件</span>
```

4.事件修饰符

输入回车键时提示

```
<input type="text" v-on:keyup.enter="alertE"/>
```

输入空格时提示

```
<input type="text" v-on:keyup.space="alertE"/>
```

五、vue改变内容——虚拟dom和diff算法

1.插值表达式的方式

- 范例一：

```
{{ count>10? "大于10","小于10"}}
```

- 范例二：

```
<p>
    {{result}}
</p>
```

```
new Vue({
        el:"#app",
        data:{
            counter:0,
            result:""
        },
        methods:{
            increase:function(step){
                this.counter+=step;
                this.result=this.counter>10?"大于10":"小于10"
            },
        }
});
```

2.计算属性：computed

1) 什么是计算属性

计算属性的重点突出在 `属性` 两个字上（**属性是名词**），首先它是个 `属性` 其次这个属性有 `计算` 的能力（**计算是动词**），这里的 `计算` 就是个函数；简单点说，它就是一个能够将计算结果缓存起来的属性（**将行为转化成了静态的属性**），仅此而已；

2) 计算属性与方法的区别

完整的 HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>布局篇 计算属性</title>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.5.21/dist/vue.js"></script>
</head>
<body>

<div id="vue">
    <p>调用当前时间的方法: {{currentTime1()}}</p>
    <p>当前时间的计算属性: {{currentTime2}}</p>
</div>

<script type="text/javascript">
    var vm = new Vue({
        el: '#vue',
        data: {
            message: 'Hello Vue'
        },
        methods: {
            currentTime1: function () {
                return Date.now();
            }
        },

        computed: {
```

```
            currentTime2: function () {
                this.message;
                return Date.now();
            }
        }
    });
</script>
</body>
</html>
```

说明

- `methods`：定义方法，调用方法使用 `currentTime1()`，需要带括号
- `computed`：定义计算属性，调用属性使用 `currentTime2`，不需要带括号；`this.message` 是为了能够让 `currentTime2` 观察到数据变化而变化

注意：`methods` 和 `computed` 里不能重名

3) 测试效果

仔细看图中说明，观察其中的差异

4) 结论

调用方法时，每次都需要进行计算，既然有计算过程则必定产生系统开销，那如果这个结果是不经常变化的呢？此时就可以考虑将这个结果缓存起来，采用计算属性可以很方便的做到这一点；**计算属性的主要特性就是为了将不经常变化的计算结果进行缓存，以节约我们的系统开销**

3.watch的用法：监控

watch用于监控参数的变化，并调用函数，newVal是能获得参数新的值，oldVal是参数老的值。

```
new Vue({
                el:"#app",
                data:{
                    counter:0,
                    result:""
                },
                methods:{
```

```
                increase:function(step){
                    this.counter+=step;
                    //this.result=this.counter>10?"大于10":"小于10"
                },
                getResult:function(){
                    return this.counter>10?"大于10":"小于10"
                }
            },
            computed:{
                getResultComputed:function(){
                    return this.counter>10?"大于10":"小于10"
                }
            },
            watch:{
                counter:function(newVal,oldVal){
                    this.result=newVal>10?"大于10":"小于10"
                }
            }
        });
```

watch的高端用法： 一秒后让count归为0，体现了vue的双向绑定

```
watch:{
    counter:function(newVal,oldVal){
        this.result=newVal>10?"大于10":"小于10";
        var vm = this;//当前data
        setTimeout(function(){
            vm.counter = 0;
        },1000);
    }
}
```

六、vue改变样式

1.class的动态绑定

```
v-bind:class="{red:attachRed}"
```

键名是类名，键值是布尔，如果是true，则将指定的类名绑定在元素上，如果是false，则不绑定。

```
<head>
        <meta charset="UTF-8">
        <title>下午</title>
        <style>
            .demo{
                width:100px;
                height:100px;
                background-color:gray;

                display:inline-block;
```

```
                margin:10px;
            }
            .red{background-color: red;}
            .green{background-color: green;}
            .blue{background-color: blue;}

        </style>
    </head>
    <body>
        <div id="app">
            {{attachRed}}
            <div class="demo" @click="attachRed=!attachRed" v-bind:class="{red:attachRed}">
</div>
            <div class="demo"></div>
            <div class="demo"></div>
        </div>
        <script src="https://cdn.bootcss.com/vue/2.5.17-beta.0/vue.min.js"></script>
        <script>
            new Vue({
                el:"#app",
                data:{
                    attachRed:false
                }
            });
        </script>
    </body>
```

2.加入computed

```
<div class="demo" :class="divClasses"></div>
```

```
new Vue({
                el:"#app",
                data:{
                    attachRed:false
                },
                computed:{
                    divClasses:function(){
                        return {//返回一个json对象
                            red:this.attachRed,
                            blue:!this.attachRed
                        }
                    }

                }
    });
```

3.双向绑定的体现

在input中输入颜色，就可以设置div的class

```
<div id="app">
            <input type="text" v-model="color"/>
            {{attachRed}}
            <div class="demo" @click="attachRed=!attachRed" v-bind:class="{red:attachRed}">
</div>
            <div class="demo"></div>
            <div class="demo" :class="divClasses"></div>
            <div class="demo" :class="color"></div>
        </div>

        <script>
            new Vue({
                el:"#app",
                data:{
                    attachRed:false,
                    color:"green"
                },
               ...
```

4.多个样式的操作

```
.red{background-color: red;color: white;}
<div class="demo" :class="[color,{red:attachRed}]">hahaha</div>
```

5.通过style设置样式

```
<div class="demo" :style="{backgroundColor:color}"></div>
```

设置div的style属性的值，style里放json对象，键是驼峰式写法，值是变量color

6.使用computed设置样式

```
<div class="demo" :style="myStyle"></div>
<input type="text" v-model="width"/>

    new Vue({
                el:"#app",
                data:{
                    attachRed:false,
                    color:"green",
                    width:100
                },
                computed:{
                    divClasses:function(){
                        return {//返回一个json对象
                            red:this.attachRed,
                            blue:!this.attachRed
                        }
                    },
                    myStyle:function(){
```

```
                        return {
                            backgroundColor:this.color,
                            width:this.width+"px"
                        }
                    }

                }
            });
        </script>
```

7.设置style属性的多个样式

```
<div class="demo" :style="[myStyle,{height:width*2+'px'}]"></div>
```

七、vue中的语句

1.分支语句

- v-if
- v-else-if
- v-else
- v-show: 实际上是让该元素的display属性为none,隐藏的效果。所以性能更好。

```
<div id="app">
            <p v-if="show">hahah</p>
            <p v-else>hohoho</p>
            <p v-show="show">hehehe</p>
            <input type="button" @click="show=!show" value="dianwo"/>
        </div>
        <script src="https://cdn.bootcss.com/vue/2.5.17-beta.0/vue.min.js"></script>
        <script>
            new Vue({
                el:"#app",
                data:{
                    show:false
                }
            });
        </script>
```

通过模板标签对多个元素进行同一的if和else管理

```
<template v-if="show">
                <h1>heading</h1>
                <p>inside text</p>
            </template>
```

2.循环语句

vue中只有for循环

```
<body>
        <div id="app">
            <ul>
                <li v-for="str in args">
                    {{str}}
                </li>
            </ul>
        </div>
        <script src="https://cdn.bootcss.com/vue/2.5.17-beta.0/vue.min.js"></script>
        <script>
            new Vue({
                el:"#app",
                data:{
                    args:["a","b","c","d"]
                }
            });
        </script>
    </body>
```

改进版：for语句里，key建议加上，作为标识.i是下标

```
<ul>
    <li v-for="(str,i) in args" :key="i">
        {{str}}{{i}}
    </li>
</ul>
```

使用templete实现循环

```
<template v-for="(str,i) in args":key="i">
        <p>{{str}}{{i}}</p>
</template>
```

循环中操作对象

```
<template v-for="person in persons">
    <p>
        <span v-for="value in person">
            {{value}}
        </span>
    </p>
    <p>
        <span v-for="(v,k,i) in person">
            {{k}}:{{v}}:{{i}}====
        </span>
    </p>
</template>

<script>
    new Vue({
```

```
        el:"#app",
        data:{
            args:["a","b","c","d"],
            persons:[
                {name:"xy",age:20,color:"red"},
                {name:"yh",age:18,color:"green"}
            ]
        }
    });
</script>
```

循环的另一种用法：

```
v-for="n in 10"   //可以在分页组件中使用
```

```
<nav>
<ul class="pagination">
    <li>
      <a href="#" aria-label="Previous">
        <span aria-hidden="true">&laquo;</span>
      </a>
    </li>
    <!--======v-for====-->
    <li v-for="n in 10"><a href="#">{{n}}</a></li>

    <li>
      <a href="#" aria-label="Next">
        <span aria-hidden="true">&raquo;</span>
      </a>
    </li>
  </ul>
</nav>
```

八、总结

vue是以数据为驱动，渐进式的web框架，MVVM双向绑定，虚拟DOM为核心和diff算法，所谓的虚拟dom和diff算法，是指当页面内容发生变化时，只更新改变的部分，是通过虚拟dom和diff算法实现这样的操作，效率非常高。