# # 如何使用data

# data : { product : “helloWorld” } 基本渲染

    <div id="app">

        <p>{{ product }} HELLO WORLD</p>

    </div>

    <script>

        const vm = new Vue({

            el: '#app',

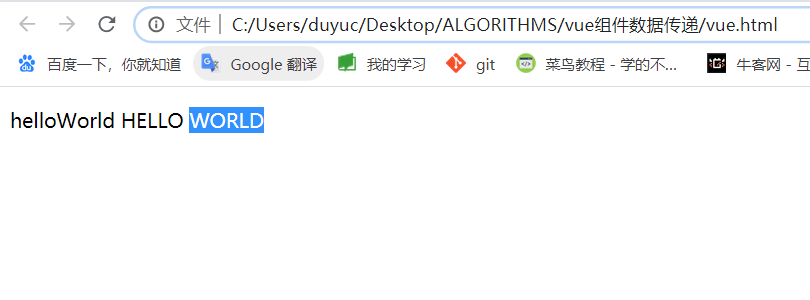
            data: {

                product : 'helloWorld'

            }

        });

    </script>



# 不要忘记符号 🡪 #

# data : { product : [“dyc1”, “dyc2”, “dyc3”] } 在无序html列表中渲染

    <div id="app">

        <ul>

            <li v-for="key in product">{{ key }}</li>

        </ul>

    </div>

    <script>

        const vm = new Vue({

            el: '#app',

            data: {

                product : [

                    'dyc1',

                    'dyc2',

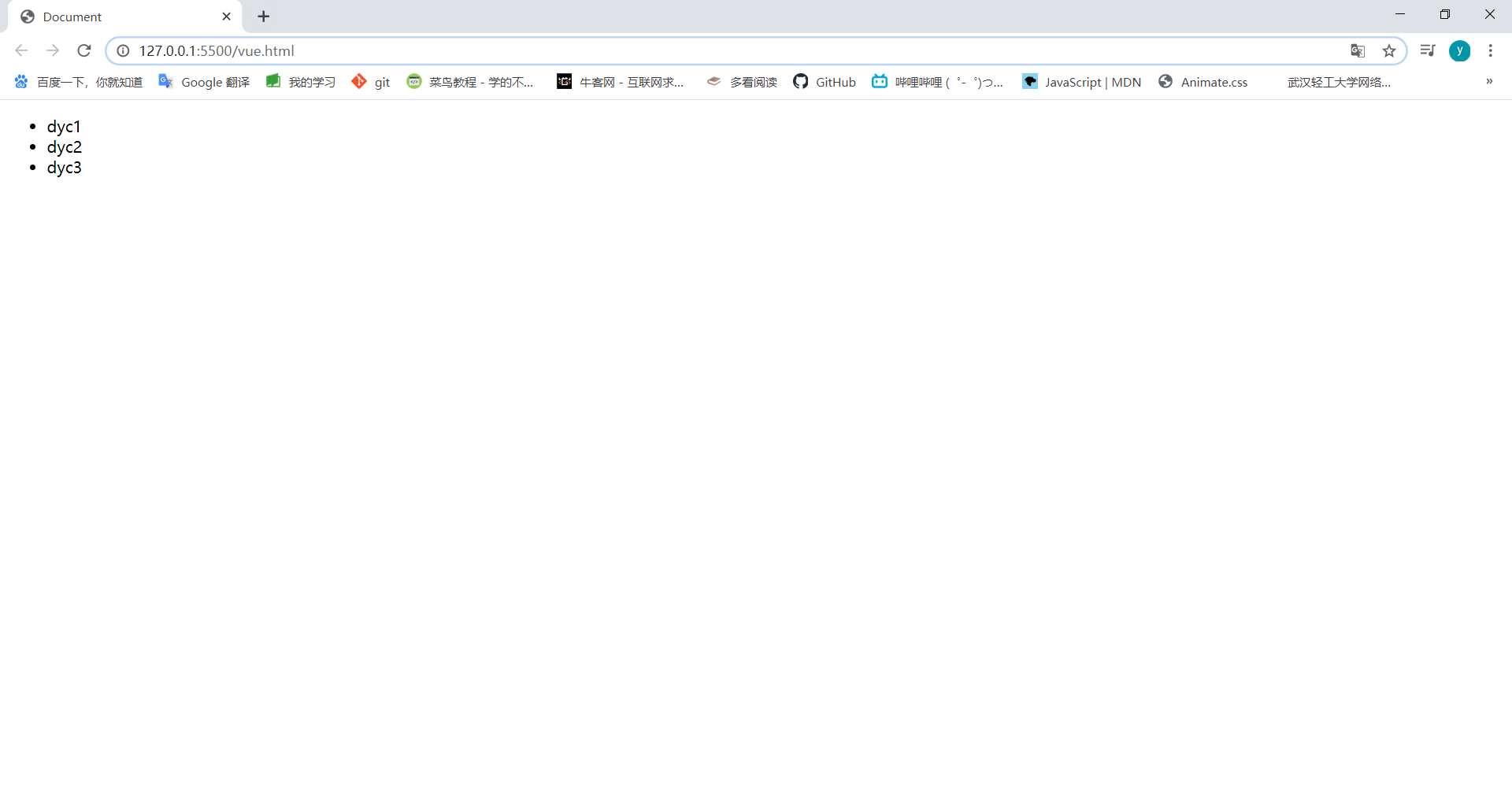
                    'dyc3'

                ]

            }

        });

    </script>



# 在created方法中使用fetch获取api上的数据, 然后渲染

    <div id="app">

        <ul>

            <li v-for="key in product">{{ key }}</li>

        </ul>

    </div>

    <script>

        const vm = new Vue({

            el: '#app',

            data: {

                product : []

            },

            created () {

                fetch("https://api.myjson.com/bins/74l63")

                    .then(response => response.json())

                    .then(json => {

                        this.product = json.products

                    })

            }

        });

# 可能是因为接口过期了,然后没有显示出来,不过过程就是这样的

# 使用v-if

    <div id="app">

        <ul>

            <li v-for="product in products">

                {{ product.name }} {{ product.number }}

                <span v-if="product.number === 5"> ---I AM FIVE </span>

            </li>

        </ul>

    </div>

    <script>

        const vm = new Vue({

            el: '#app',

            data: {

                products : [

                    {id : 1, name : "dyc1", number : 2},

                    {id : 2, name : "dyc1", number : 3},

                    {id : 3, name : "dyc1", number : 4},

                    {id : 4, name : "dyc1", number : 5}

                ]

            }

        });

    </script>

# 注意: <span v-if>一定要放在<li>标签里面

# 使用computed得到number的和

    <div id="app">

        <ul>

            <li v-for="product in products">

                {{ product.name }} {{ product.number }}

                <span v-if="product.number === 5"> ---I AM FIVE </span>

            </li>

            <h2>Total : {{ total }}</h2>

        </ul>

    </div>

    <script>

        const vm = new Vue({

            el: '#app',

            data: {

                products : [

                    {id : 1, name : "dyc1", number : 2},

                    {id : 2, name : "dyc1", number : 3},

                    {id : 3, name : "dyc1", number : 4},

                    {id : 4, name : "dyc1", number : 5}

                ]

            },

            computed: {

                total () {

                    return this.products.reduce((sum, product) => {

                        return sum + product.number

                    }, 0)

                }

            }

        });

    </script>

# reduce()方法对数组中的每个元素执行函数(升序执行), 将结果汇总为单个返回值

<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce>

# 使用@click添加交互button

    <div id="app">

        <ul>

            <li v-for="product in products">

                {{ product.name }} {{ product.number }}

                <span v-if="product.number === 5"> ---I AM FIVE </span>

                <button @click="product.number += 1">

                    add

                </button>

            </li>

            <h2>Total : {{ total }}</h2>

        </ul>

    </div>

    <script>

        const vm = new Vue({

            el: '#app',

            data: {

                products : [

                    {id : 1, name : "dyc1", number : 2},

                    {id : 2, name : "dyc1", number : 3},

                    {id : 3, name : "dyc1", number : 4},

                    {id : 4, name : "dyc1", number : 5}

                ]

            },

            computed: {

                total () {

                    return this.products.reduce((sum, product) => {

                        return sum + product.number

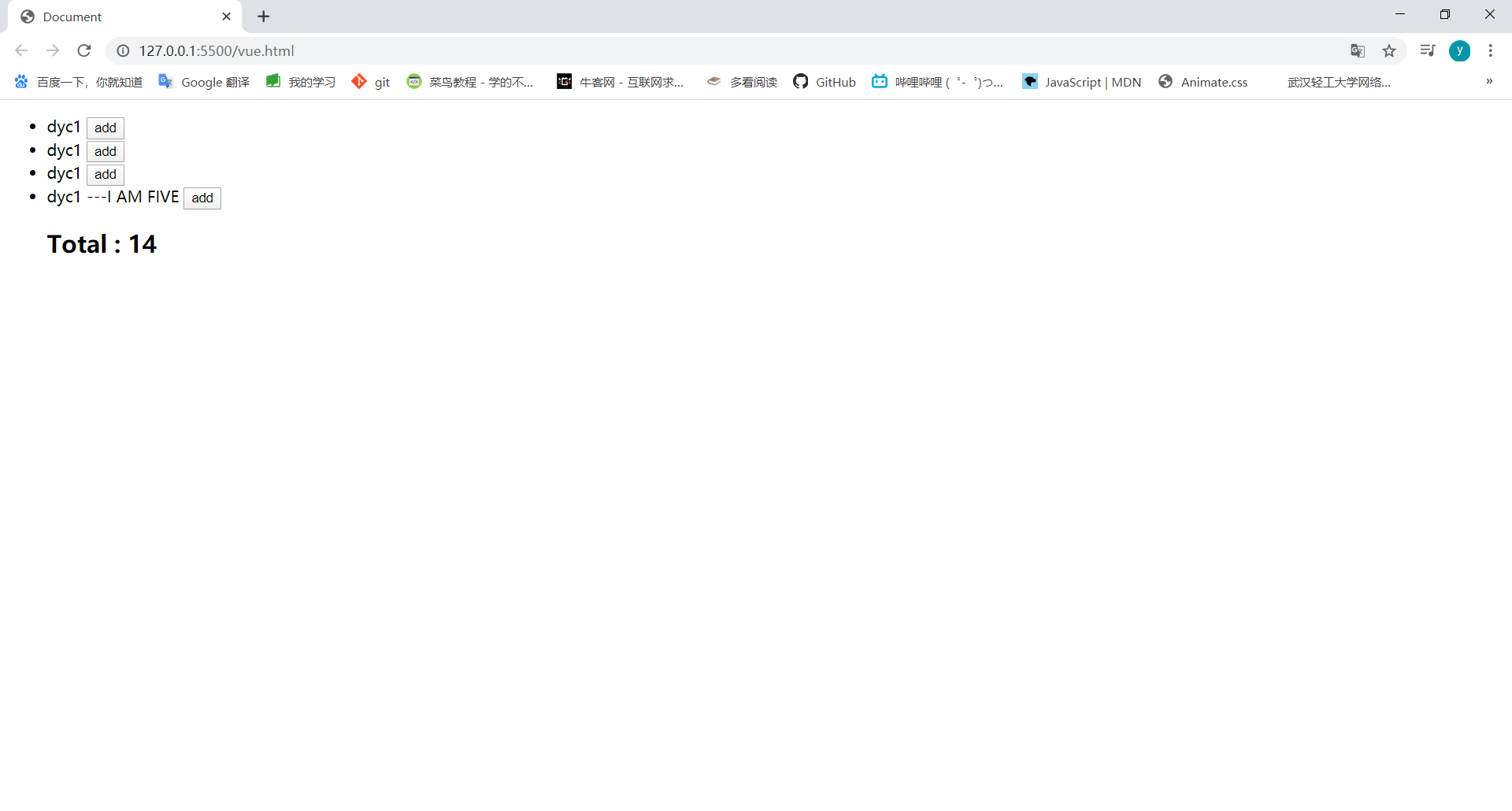
                    }, 0)

                }

            }

        });

    </script>



# 使用v-model

    <div id="app">

        <ul>

            <li v-for="product in products">

                {{ product.name }}

                <input type="number" v-model.number="product.number">

                <span v-if="product.number === 5"> ---I AM FIVE </span>

                <button @click="product.number += 1">

                    add

                </button>

            </li>

            <h2>Total : {{ total }}</h2>

        </ul>

    </div>

    <script>

        const vm = new Vue({

            el: '#app',

            data: {

                products : [

                    {id : 1, name : "dyc1", number : 2},

                    {id : 2, name : "dyc1", number : 3},

                    {id : 3, name : "dyc1", number : 4},

                    {id : 4, name : "dyc1", number : 5}

                ]

            },

            computed: {

                total () {

                    return this.products.reduce((sum, product) => {

                        return sum + product.number

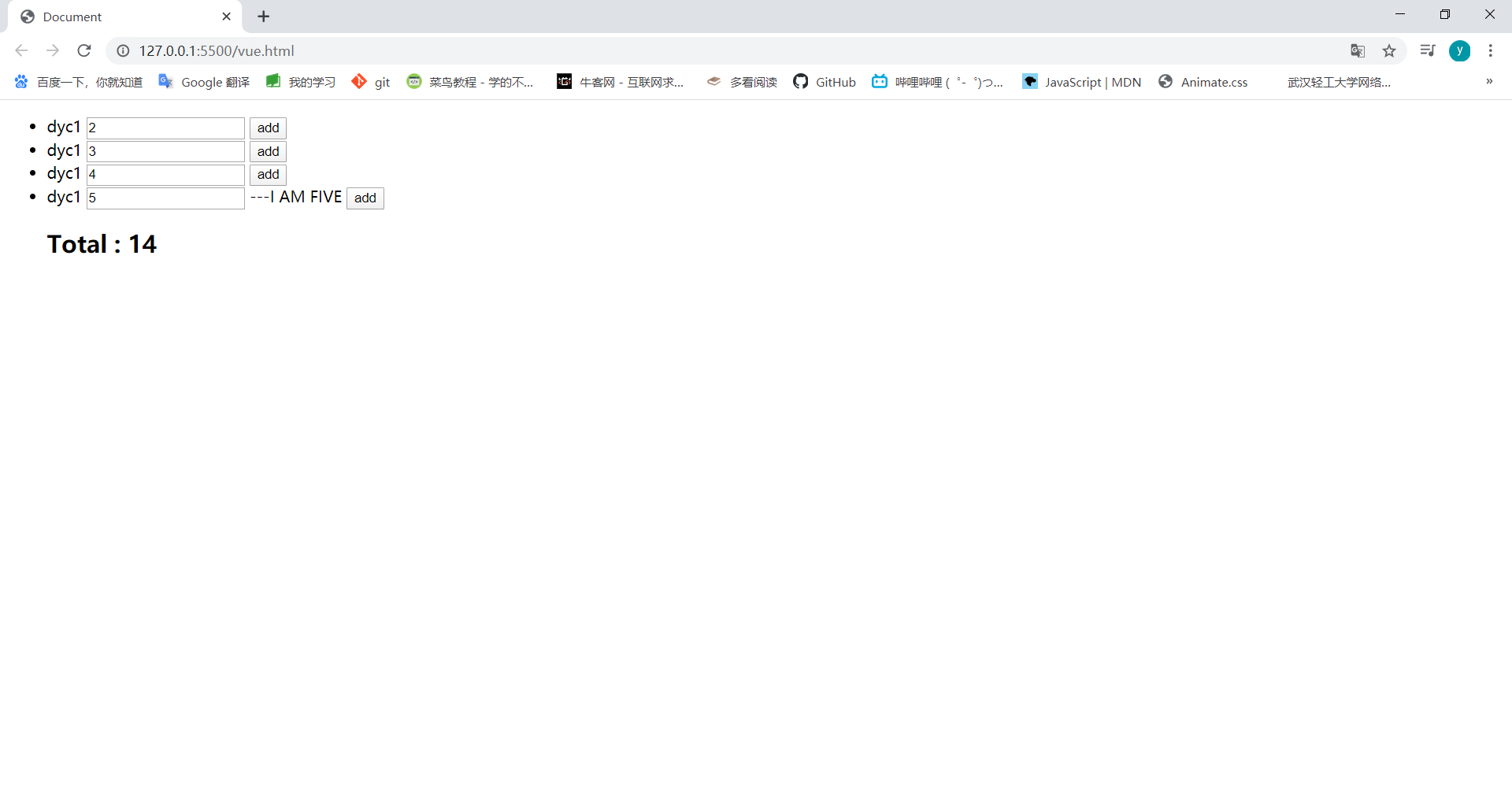
                    }, 0)

                }

            }

        });

    </script>



# # 组件基础

    <div id="app">

        <button-counter></button-counter>

    </div>

    <script>

        Vue.component('button-counter', {

            data : function () {

                return {

                    count : 0

                }

            },

            template : '<button v-on:click="count++"> You clicked me {{ count }} times.</button>'

        });

        var vm = new Vue({

            el : '#app',

            data : {

            }

        });

    </script>

# 注意: var vm = new Vue({ })得放后面

# 组件可以通过props为组件定义属性

        Vue.component('button-counter', {

            props : [

                'title'

            ],

            data : function () {

                return {

                    count : 0

                }

            },

            template : '<button v-on:click="count++">You clicked me {{ count }} times.</button>'

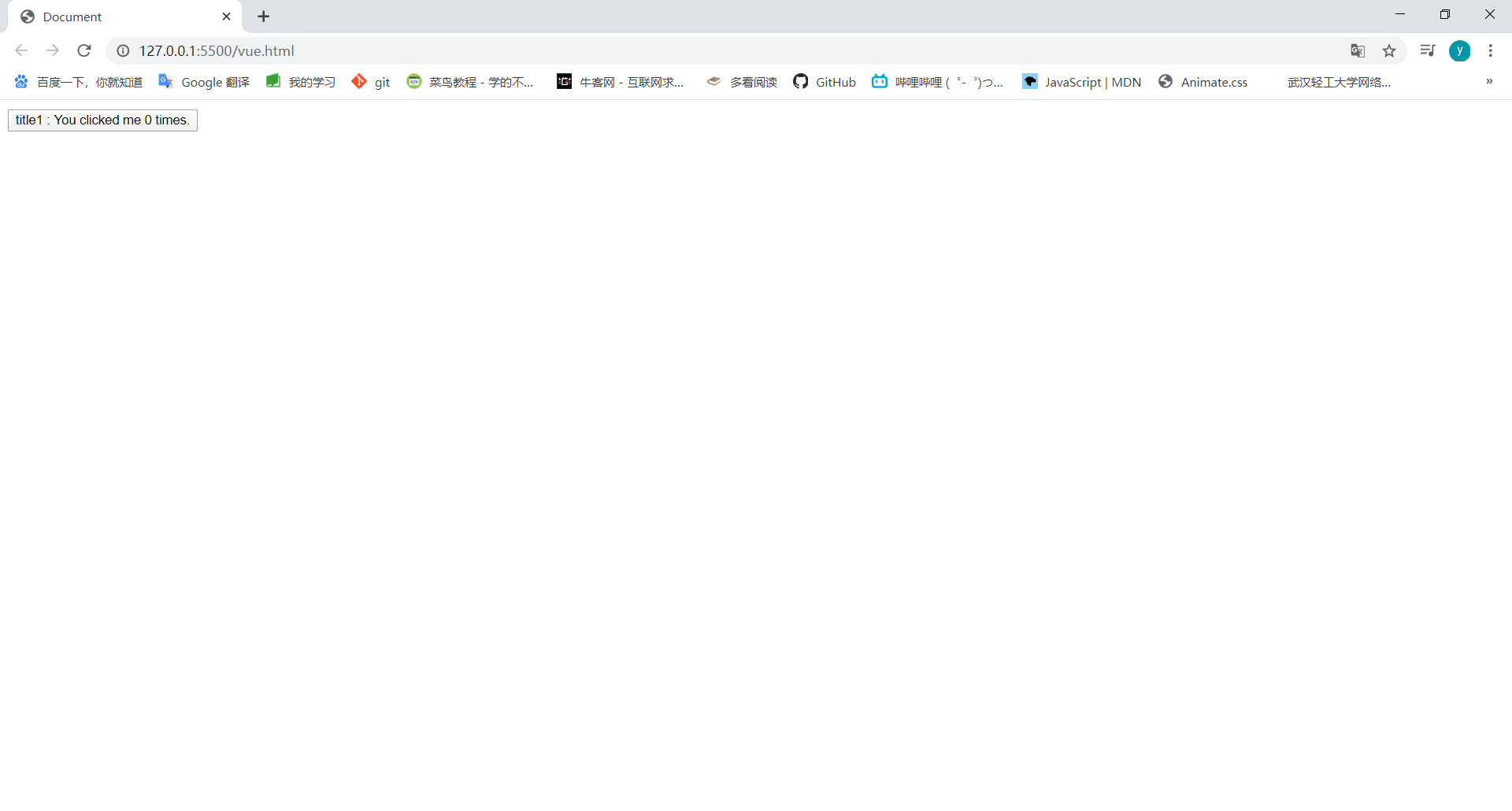
        });

# 在模板上对组件进行应用和渲染

template : '<button v-on:click="count++"> {{ title }} You clicked me {{ count }} times.</button>'

# 在调用组件的时候, 可以通过组件的属性给去为其赋值

<button-counter title="title1 : "></button-counter>



# 组件的模板需要一个根节点 <div>

            template : `<div>

                            <h1>helloWorld</h1>

                            <button v-on:click="count++">

                                {{ title }} You clicked me {{ count }} times.

                            </button>

                        </div>`

# 组件的事件监听

# 把@click的js脚本改写成js的函数

# 用methods去声明对应的函数

            template : `<div>

                            <h1>helloWorld</h1>

                            <button v-on:click="cilckFunction">

                                {{ title }} You clicked me {{ count }} times.

                            </button>

                        </div>`,

            methods : {

                cilckFunction () {

                    this.count++

                }

            }

# 通过对象的$emit()方法去触发一个事件

# 它的第一个参数是事件名称

# 必须小写!!!!

# 它的第二个参数是可携带的一些参数

# 在组件的父级可以通过事件绑定的方式去接受$emit()方法去触发的事件

    <div id="app">

        <button-counter title="title1 : " @clicknow="nowClick"></button-counter>

    </div>

    <script>

        Vue.component('button-counter', {

            props : [

                'title'

            ],

            data : function () {

                return {

                    count : 0

                }

            },

            template : `<div>

                            <h1>helloWorld</h1>

                            <button v-on:click="cilckFunction">

                                {{ title }} You clicked me {{ count }} times.

                            </button>

                        </div>`,

            methods : {

                cilckFunction () {

                    this.count++ ;

                    this.$emit('clicknow', this.count);

                }

            }

        });

        var vm = new Vue({

            el : '#app',

            data : {

            },

            methods : {

                nowClick : (e) => {

                    console.log(e)

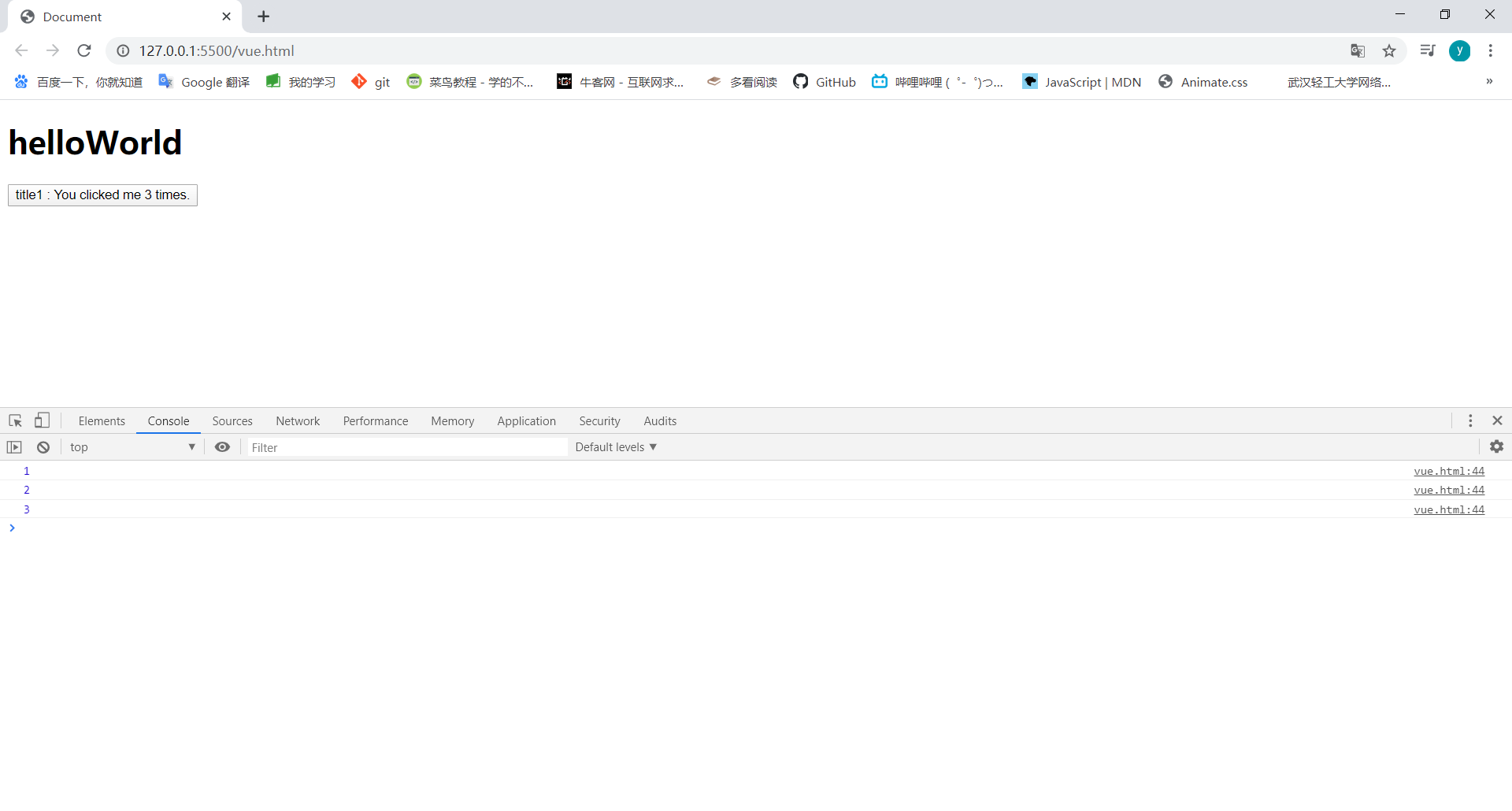
                }

            }

        });

</script>

# 此时父级就接收到了子级的值!!!!



# <slot> 组件的插槽

# 在组件的标签内部就可以插入html内容

        <button-counter title="title1 : " @clicknow="nowClick">

            <h1>i am the king of the world</h1>

        </button-counter>

            template : `<div>

                            <h1>helloWorld</h1>

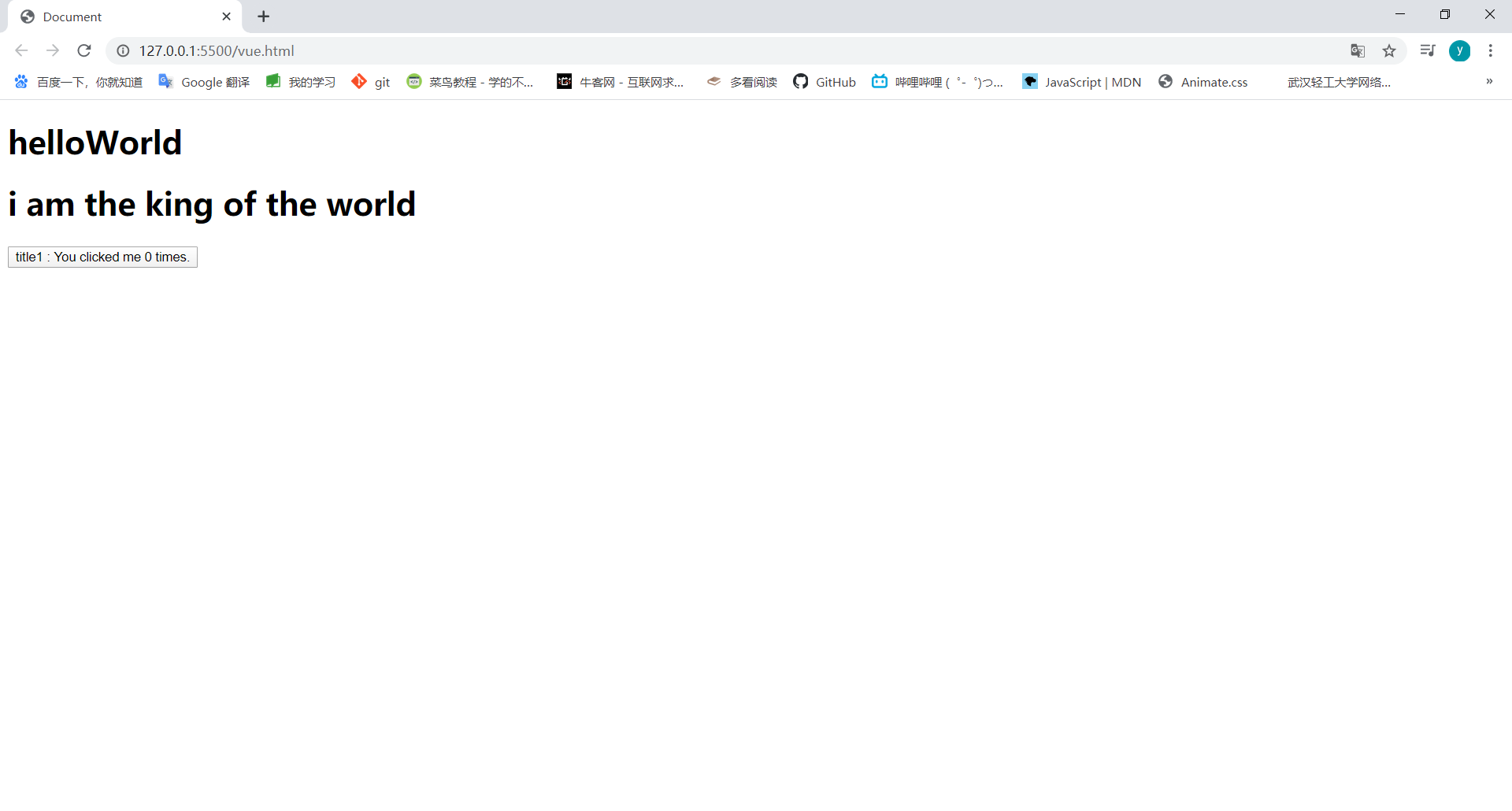
                            <slot></slot>

                            <button v-on:click="cilckFunction">

                                {{ title }} You clicked me {{ count }} times.

                            </button>

                        </div>`



# # 组件注册

# 全局注册 Vue.component()

# 局部注册 在对Vue函数初始化的时候( 创建Vue对象的时候 ), 声明components属性

        var vm = new Vue({

            el : '#app',

            data : {

            },

            components : {

                test : {

                    template : ``,

                    method : {}

                }

            }

        });

# # 单文件组件

# 安装npm

# Node Package Manager

# 安装vue-cli

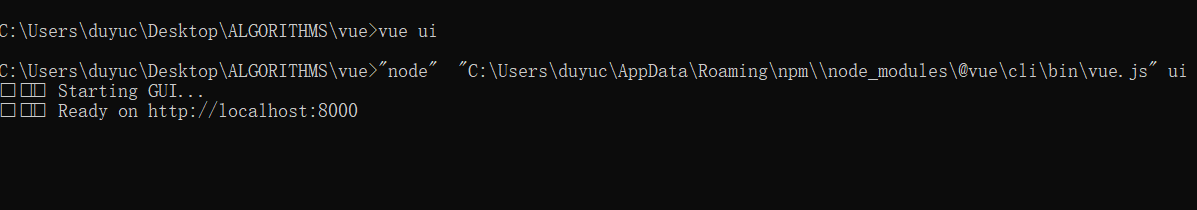
# npm install -g @vue/cli

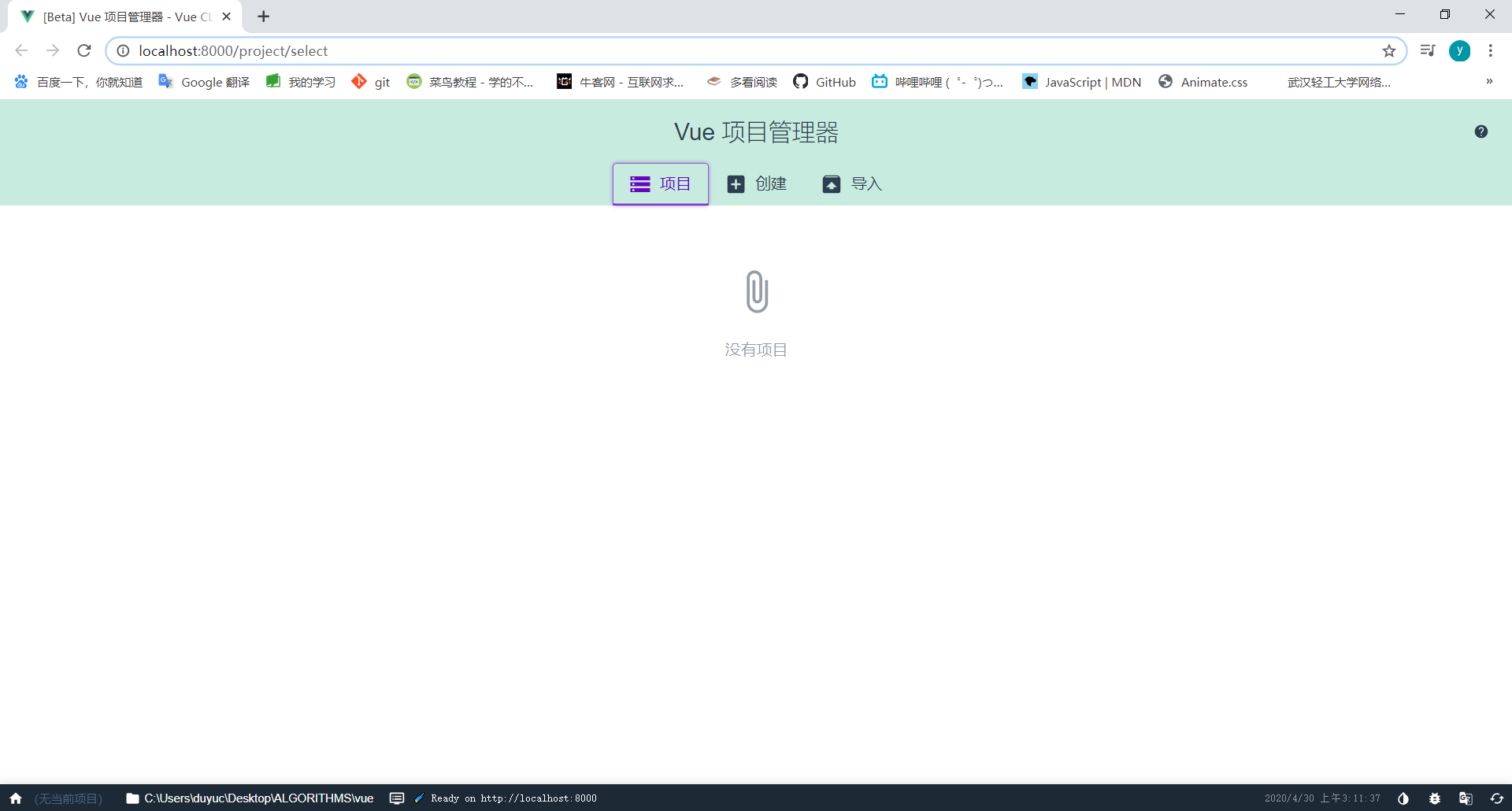
# 安装webpack

# npm install -g webpack

# webpack是JavaScript打包器( module bundler )

# vue ui 打开 vue项目管理器







# 然后就可以创建项目了

# 目录结构

# public 打包之后用于部署到生产环境中的目录

# src 开发的目录

# components 整个项目的组件目录

# HelloWorld.vue 单文件组件

# App.vue 整个项目的入口文件, 完成对组件的引入