### 符号代表含义

&：和

&&：并且

||： 或

-S & --save // 运行时依赖 (默认安装，可以不用写)

-D & // 安装时依赖

### 记忆不熟----------------区域

input

outline: none; //去除input的外轮廓

text-indent: 10px; //光标距离

caret-color: #fffbfb; //改变光标的颜色

// 改变input里面文字的颜色

input::-webkit-input-placeholder{

    color:#ffffff;

}

### html---------------------------区域

**下拉菜单**

<select name="" id="">

        <option value="">---选择省---</option>

    </select>

    <select name="" id="">

        <option value="">---选择市---</option>

        </select>

    <select name="" id="">

        <option value="">---选择县---</option>

    </select>

**css---------------------------区域**

input

outline: none; //去除input的外轮廓

text-indent: 10px; //光标距离

caret-color: #fffbfb; //改变光标的颜色

// 改变input里面文字的颜色

input::-webkit-input-placeholder{

    color:#ffffff;

}

**设置字与字之间的间距**

*letter-spacing*: 10px;

**-----------------------------------------------**

**JS---------------------区域**

setTime

location 可以获取地址栏上的所有信息

location.search.substr() //获取地址栏上的id

Location.href = ’index.html ’ //地址跳转

**append() 方法**

// $('#members').append(需要加入的内容);

                $('#members').append(str);      //在元素的内部，插入指定的内容

**焦点**

blur() //失去焦点 例：blur(function(){//失去光标后的事件})

**事件类型：//这是原生js用法**

onchange 事件 // 改变内容时触发

onclick 事件 //点击是触发

**事件类型：//这是jQuery用法**

change 事件 // 改变内容时触发

$('#input\_avatar').change()

click 事件 //点击是触发

**JOSN格式转对象**

// JSON格式转对象

*var* res = JSON.parse();

在webAPI day3

// 转化为JOSN(带有一定格式的字符串)

    // var arr = JSON.stringify(arr);

    // localStorage.setItem('key', arr);  //setItem() 存数据

    //----------------------------------------------------前端需要做的工作

    //1.获取数据

    var key = localStorage.getItem('key'); //getItem() 读数据

    // 转成复杂数据类型

    var arr = JSON.parse(key);

**通过图片路径展示图片**

createObjectURL

如果我们做文件上传的时候，在没有上传[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的情况下想看到上传图片的效果图的时候就可以通过var url=URL.createObjectURL(obj.files[0]);获得一个http格式的url路径，这个时候就可以设置到<img>中显示了。

// 2.找到文件对象 files-->文件的意思

*var* file = this.files[0];

*console*.log(file);

      // 为文件对象创建一个URL（临时的URL），该URL用于访问图像/获取文件路径

*var* url = URL.createObjectURL(file);  //URL是js内置对象

**设置被选元素的属性值**

attr()

$('.btn').attr(‘属性’，‘属性值’);

**按钮-->阻止页面刷新**

preventDefault()

$('.input\_sub').click(*function*(*e*) {

                // 按钮提交一次，页面刷新， 然后阻止提交按钮的默认行为，可以使页面不刷新

                e.preventDefault();

**setItem和getItem的区别**

setItem-设置的意思

getItem-获取的意思

// setItem('属性名', 属性值), setItem-设置的意思

    window.localStorage.setItem('token', backData.token)

// getItem('属性名', 属性值), getItem-获取的意思

window.localStorage.getItem('a', backData.token);

// removeItem('属性名'), removeItem-删除的意思

   window.localStorage.removeItem('a');

# **[Number()、parseInt() 和 parseFloat() 的区别](https://www.cnblogs.com/yi0921/p/6196841.html)**

处理整数的时候parseInt()更常用 --购物车买东西

parseInt() 遇到小数点会停止解析，因为小数点并不是有效的数字字符。

parseFloat() 所解析的字符串中第一个小数点是有效的

**Git 版本管理**

**先存到本地然后在存到远程**

**本地仓库使用**

1.初始化仓库  git init

        2.添加文件(暂存区) git add .  -->【.代表所有的文件】      git add 文件名 -->【添加单个文件】

         3.提交更改(本地仓库区) git commit -m "信息"

git status //查看文件夹状态 可以查看到哪些文件夹被修改了

         1.查看分支 git branch //查看分支(简写)git log --oneline

         2.创建分支 git branch 分支名

         //6.创建并直接切换分支 git checout -b 分支名

         3.切换分支 git checkout 分支名

         4.合并分支 git merge 分支名

         5.删除分支 git branch -d 分支名

6.删除本地仓库 rm -rf .git master

7.向管理员请求合并分支 Pull Requests

8.删除关联的远程url git remote rm origin

9.添加关联的远程url git remote add origin URL

10.查看远程url git remote -v

11.更新本地仓库 git pull

注意:

1. 之前设置过信息,`commit` 使用的就是之前的

2. 修改了信息之后,再次提交使用设置之后的

3. 如果要删除之前的

1. 回滚: git reset --hard 版本号 //版本回退

2. 重新提交 git add .  然后 git commit -m"信息"

**远程仓库使用**

1.新建远程仓库: 比如 github, 码云, gitlab等等

         2.把本地仓库和远程仓库建立关系。（此时还没有推送）

            git remote add 远程仓库地址别名  远程仓库地址

         3.第一次推送：git push -u 远程仓库地址别名 本地分支:远程分支

         4.如果本地分支名和远程分支名一样， 则可以省略 :远程分支

         5.后续的每一次推送 git push 就可以了

6.加了参数-u后，以后即可直接用git push 代替git push origin master

注意事项：

            首次推送 == 会弹出一个窗口， 让我们填写GitHub的账号和密码

重新设置账户和密码:

清空所有用户名和密码 => git config --system --unset credential.helper

**从远程仓库克隆一个版本库到本地。**

# 默认在当前目录下创建和版本库名相同的文件夹并下载版本到该文件夹下

$ git clone <远程仓库的网址>

# 指定本地仓库的目录

$ git clone <远程仓库的网址> <本地目录>

# -b 指定要克隆的分支，默认是master分支

$ git clone <远程仓库的网址> -b <分支名称> <本地目录>

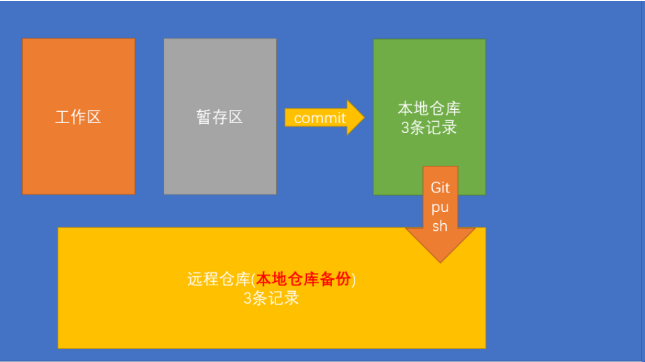
Git强行把自己的项目推上GitHub（需要慎重！！）

<!--

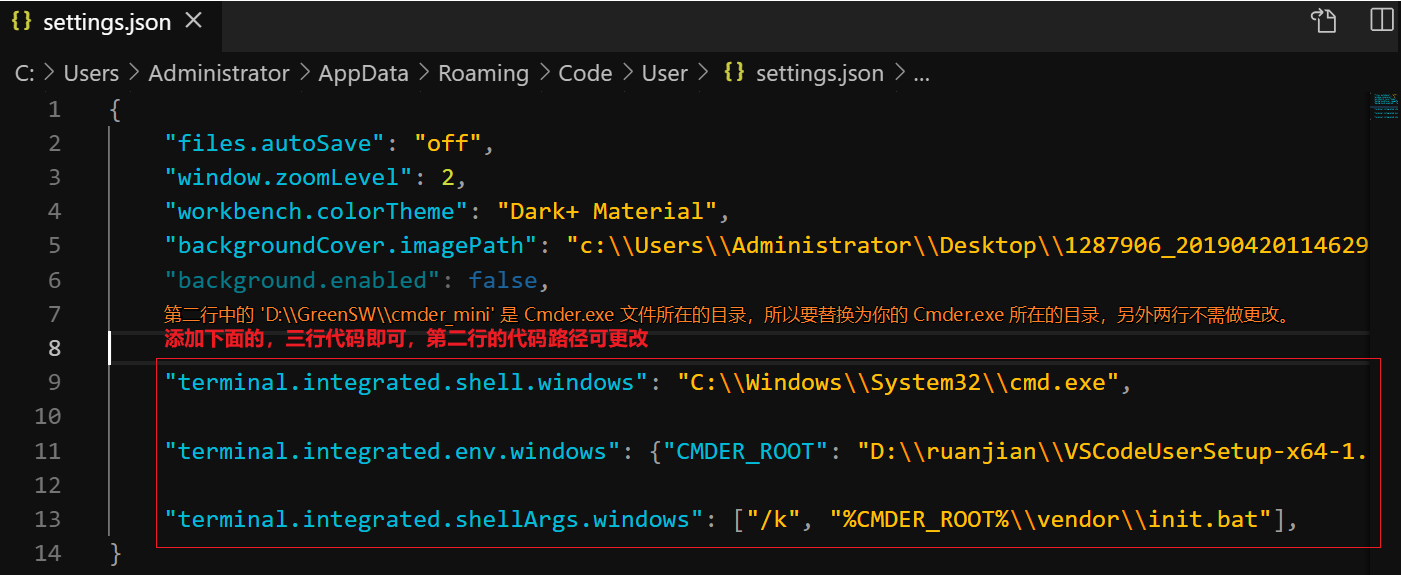
git push -u origin master --force

-->

Git工作描述



**Visual Studio Code 终端打不开或者路径错误问题**



**Ajax**

get和post区别

**不同点：**

**get:** 用于向服务器请求资源（图片，文件，数据....），它只是请求，而不会改变服 务器上的资源。

**post:** 用于向服务器上提交数据,它可能会修改服务器上的资源。

**相同点：**

1.get和post请求都可以在发请求时附带一些数据。

2.get和post请求都能够从服务器上获取返回的数据。

**备注：这是Vue day3所学****\_\_\_\_**

**JSON Server**

JSON Server 是什么：

1.他是一个提供测试环境接口的工具，它可以帮我们快速生成一套接口服务，专门用于学习测试。

2.它是免费开源的命令行工具

**安装教程**

1.npm安装：npm install -g json-server

2.测试是否安装成功：json-server --version

如果看到一个版本输出，就证明OK了

3.创建一个目录 json-server-demo; 然后在该目录中创建一个文件夹 db.json 并写入以下内容

{

                "posts": [

                    { "id": 1, "title": "json-server", "author": "typicode" }

                ],

                "comments": [

                    { "id": 1, "body": "some comment", "postId": 1 }

                ],

                "profile": { "name": "typicode" },

                "users": [{

                    "id": 1,

                    "name": "张三",

                    "age": 18,

                    "gender": "男"

                }]

            }

4.最后，在命令行中进入 `db.json` 文件所属目录，执行 json-server --watch db.json

如果成功，将看到以下输入：【这个也是开启服务】

\{^\_^}/ hi!

            Loading db.json

            Oops, db.json doesn't seem to exist

            Creating db.json with some default data

            Done

            Resources

            http://localhost:3000/posts

            http://localhost:3000/comments

            http://localhost:3000/profile

            Home

            http://localhost:3000

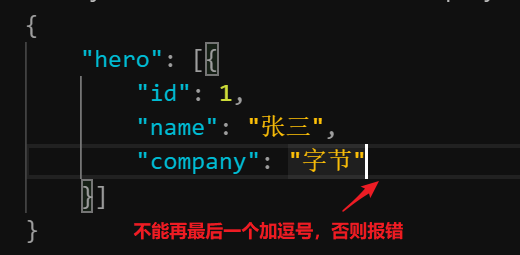
            Type s + enter at any time to create a snapshot of the database

            Watching...

1. 接口服务默认占用 3000 端口

json-server

**在里面加 “ ，”规则**



**axios介绍：** <http://www.axios-js.com/>

Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中。 axios 本身和 Vue 没有一毛钱关系，只是简单纯粹的封装了 HTTP 请求功能。可以运行在任何支持 JavaScript 环境的平台。

**特点：**

1.从浏览器中创建 XMLHttpRequests

2.从 node.js 创建 http 请求

3.支持 Promise API

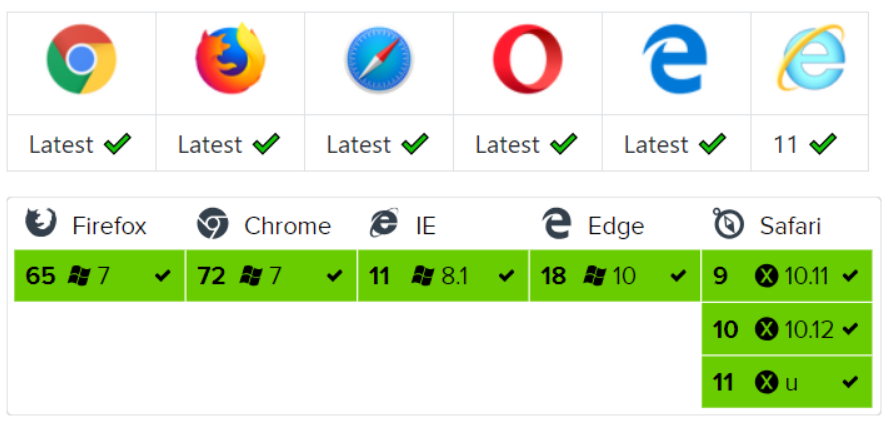
4.拦截请求和响应

5.转换请求数据和响应数据

6.取消请求

7.自动转换 JSON 数据

8.客户端支持防御 XSRF



注意：

axios 依赖原生的 ECMAScript 6 Promise 支持。

如果浏览器不支持 ECMAScript 6 Promise，可以使用 [es6-promise](https://github.com/stefanpenner/es6-promise) 进行兼容处理。

**基本使用：**

Npm下载：npm install axios

请求参数的使用：get post

GET /posts 查询

GET /posts/1

POST /posts 新增

PUT /posts/1 修改

PATCH /posts/1

DELETE /posts/1 删除

**-------------------------------------------这是尾部**

**Vue组件**

**全局组件：**

// 注册一个全局组件

        // 参数1：组件名称

        // 参数2：组件对象

        Vue.component("",{

            // 模板   代表页面的结构，有且只有一个根元素

            template:``,

            data(){},

            methods:{},

        });

**局部组件：**

//  注册局部组件

             component:{"":{

                template:``,

                data(){},

                methods:{},

             }},

**整体代码**

<script src="./vue.js"></script>

    <script>

        // 注册一个全局组件

        // 参数1：组件名称

        // 参数2：组件对象

        Vue.component("",{

            // 模板   代表页面的结构，有且只有一个根元素

            template:``,

            data(){},

            methods:{},

        });

        const vm = new Vue({

             el: "#app",

             data: {},

             methods: {},

            //  注册局部组件

             component:{"":{

                template:``,

                data(){},

                methods:{},

             }},

         });

    </script>

**npm安装卸载命令**

**安装模块**

npm install -g xxx //安装全局模块

npm install xxx //安装xxx模块到当前命令行所在目录

**卸载其他模块** -g代表全局

npm uninstall -g xxx

**Vue-cli工具安装 创建项目**

1.创建一个2.0版本的项目

        vue  init webpack-simple 名字     //创建2.0项目

        ? Project name (heroes1)      //意思是：你的项目名确定叫heroes1吗？如果确定直接回车

        ? Project description (A Vue.js project)    //Project description 项目说明/描述

        ? Author (a123456789 <2650553461@qq.com>)   //Author 作者，找的是git上注册的账号

        ? License (MIT)   //许可证，不用管直接回车

        ? Use sass? (y/N)   //是否使用sass，不使用，直接回车下一步即可

        To get started:        //意思是：如果你想启动该项目，请执行以下代码

        cd heroes1      //进入项目

        npm install     //安装依赖包

        npm run dev     //启动项目

    2.导入素材处理样式--示例项目

        2.1 安装 bootstrap固定版本     npm i  bootstrap@3.3.7

        2.2 安装完成之后 ,在main.js入口处引入css文件

            import "./../node\_modules/bootstrap/dist/css/bootstrap.css"; // 引入 bootstarp的样式文件

            import "./assets/index.css"; // 引入index.css

        2.3

            重启运行,发现bootstrap.css文件 运行报错

            根据错误 需要在webpack.config.js增加对不识别文件的处理

            {

            test: /.(ttf|woff2|woff|eot)$/,

            loader: "file-loader",

            options: {

            name: "[name].[ext]?[hash]"

            }

            }

\*.创建一个4.0版本的项目 //-----------------------------------------------------

vue create 名字 //创建项目

        ? Please pick a preset: (Use arrow keys)

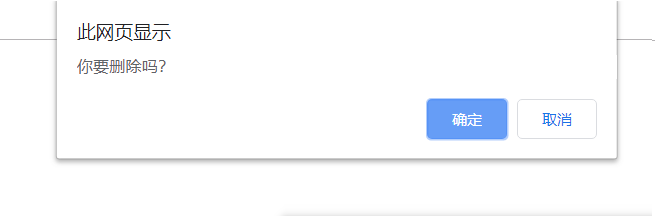
        > default (babel, eslint)                   //遇到这两个直接回车就可以了

        $ cd heroes4.0      //进入项目

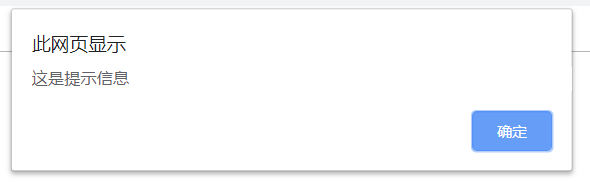
        $ npm run serve     //启动项目

**页面弹框**

confirm("你要删除吗？");



alert("这是提示信息");



**npm、cnpm、yarn三种下载方式**

这三种下载方式不一样，但是里面的代码都一样，里面文件夹可能不一样：

npm: 是从国外服务器下载的;

cnpm: 淘宝搞出来的，是从国内服务器下载的;

yarn:

1.yarn安装

<!--

// 全局安装yarn

npm install -g yarn

// 安装成功后，查看版本号

    yarn --version

   -->

**CSS样式**

Height:”100vh” //意思是高度百分之百

**Vue项目里面的各个文件夹代表的含义**

**需要一个样式的初始化：less文件在，（备注：此文件在day6里面）**

**各个文件介绍：**

Views 一般放路由级组件

assets 资源文件

**按需加载**

// {

  //   path: '/about',

  //   name: 'about',

  //   // route level code-splitting

  //   // this generates a separate chunk (about.[hash].js) for this route

  //   // which is lazy-loaded when the route is visited.

  //   component: () => import(/\* webpackChunkName: "about" \*/ '../views/About.vue')

  // }

**强制跳转**

redirect: “要跳转的地方 ”

**ref 使用**



**获取DOM节点**

**原生js获取方法：**

document.getElementById(); // id名

document.getElementsByTagName(); //标签名

document.getElementsByClassName(); //类名

document.querySelector() //css选择符模式

document.querySelectorAll() //获取所有元素

**Vue获取DOM方法：**

$refs

例如：console.log(this.$refs.div)

**数据存储到本地**

普通存储

window.localStorage.setItem('user-token', result.data.data.token)

带有一定格式的字符串存储

window.localStorage.setItem('user-token2', JSON.stringify(result.data.data.token))

**form的默认提交**

//.prevent 阻止默认提交

<form action="" @submit.prevent="onAdd"></form>

处理容易发生异常的代码

<!--

        把容易引发异常的代码放到 try 里面

        在 try 中一旦遇到异常，就直接进入 catch

        放到 try-catch中的代码如果发生异常就不会导致程序的退出了

        如果 try 里面的代码没有异常是不会执行 catch 里面的代码的

     -->

    <script>

        try {

            console.log(2)

            JSON.parse('{ "a": 123 }')

            console.log(3)

        } catch (err) {

            window.alert('转换失败了')

        }

    </script>

**JSON-bigint**

JSON-bigint 他的作用是把 json格式字符串转为 JavaScript 对象

        它和JSON.parse 的区别是，他会把数据中的大数字（超出JS安全整数范围的数字）通过一种算法

**在一个组件内使用（事件）不好使？解决方案**

**添加 .native 修饰符**

<!--

      如果给一个组件注册一个原生 JavaScript 事件

      需要使用 .native 修饰符

 -->

   <el-dropdown-item @click.native="onLogout">退出</el-dropdown-item>

**删除token**

**removeItem('user-token')**

// 删除 token

        window.localStorage.removeItem('user-token')

        // 确认退出  然后跳转到登录页

        this.$router.push('/login')

**用npm测试安装的XXX是否成功了**

**例如：检查hexo是否安装成功**

hexo --version

**webpack打包工具**

**打包项目：**

npm run build

npm和cnpm切换使用

1.设置使用淘宝cnpm

npm config set registry https://registry.npm.taobao.org

2.查看是否配置成功，成功后即可直接使用cnpm命令

npm config get registry

3.还原npm仓库

npm config set registry https://registry.npmjs.org

jq删除css中的属性

$('.layui-form-label').css('width', 'inherit')