目录

第一阶段：前端入门

1.Html5

2.Css3

第二阶段：JavaScript

①JavaScript 基础语法

②JavaScript 高级（WebAPI/Dom Bom）

③JavaScript 框架（jQuery）

④数据可视化（echarts）

第三阶段 服务端编程

1.node js基础

2.ajax

3.promise

4.axios

5.了解学习es6语法，http协议规则

6. TypeScript

第四阶段 Vue.js

①框架：Vue.Js【vue2，vue3】

②学会使用UI组件库（网站快速成型工具）

③vue项目实战

第五阶段 微信小程序

第六阶段 React.js 基础

第一阶段：前端入门

### CSS & HTML

对于刚入门的朋友我依旧建议先将 CSS(3) & HTML(5) 的知识点认真学习一边。学习的途中最好是学习完一部分就自己在敲一遍代码，加深自己的记忆。

当然如果你愿意，建议你可以先仿一个网站的静态页面（掘金、知乎等都可以），有一些属性就可以了解他实际的实现场景。

当然刚开始敲代码的时候还是不要过分依赖自动补全功能，一开始就使用自动补全对你记忆一些属性时没有帮助的；踏实点学习，日后会有回报的。

#### 关于 CSS(3) 你需要了解的一些知识点

* 盒模型（标准 & IE ）
* flex、float、Normal Flow 等的理解
* CSS 常用选择器
* 行内、内部、外部样式的区别
* CSS 层叠规则
* BFC 与 IFC 的了解
* CSS3 的 transform、transition、animation 等属性的运用了解
* 响应式布局的理解
* ……

CSS 说容易也容易，说复杂也复杂；因为 CSS 总是能给你意外的惊喜。

### HTML(5) 你需要了解的一些知识点

说到 HTML 我想有很多人是 div 一把梭。因为 div 用的爽，不用担心默认样式。

有人说 HTML 语义化的优点很多，比如清晰的页面结构、有利于 SEO、便于团队开发和维护；这些我都承认，不过我还是喜欢 div 一把梭。

* HTML 语义化( 不是很理解为什么面试总会问 )
* canvas
* 本地存储（ localStorage、sessionStorage、cookie 的理解 ）
* video 和 audio 的使用
* 应用缓存( cache manifest )
* ……

第二阶段：JavaScript

[JavaScript 基础语法](https://blog.csdn.net/m0_67911008/article/details/129846397?ops_request_misc=%7B%22request%5Fid%22%3A%22169571230016800213061915%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=169571230016800213061915&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-6-129846397-null-null.142^v94^insert_down28v1&utm_term=%E5%89%8D%E7%AB%AF&spm=1018.2226.3001.4187" \l "%C2%A0 %E2%91%A0JavaScript %E5%9F%BA%E7%A1%80%E8%AF%AD%E6%B3%95" \t "https://blog.csdn.net/m0_67911008/article/details/_self)

1、JS中的变量

1.1、变量的创建

1.2、动态类型

2、基本数据类型

2.1、 number数字类型

2.2、string字符串类型

2.3、boolean布尔类型

2.4、undefined未定义数据类型

2.5、null空值类型

3、运算符

3.1、比较运算符

3.2、逻辑运算符

4、条件语句和循环语句

5、数组

5.1创建数组

5.2获取数组元素

5.3新增数组元素

5.4删除数组元素

6、函数

6.1 语法格式

6.2函数表达式

6.3作用域

7、对象

7.1创建对象

JavaScript 高级（WebAPI/Dom Bom）

### Web API介绍

#### 2.API的概念API的概念

#### API（Application Programming Interface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。

#### 任何开发语言都有自己的API

#### API的特征输入和输出(I/O)

#### API的使用方法(console.log())

1. Web API的概念

浏览器提供的一套操作浏览器功能和页面元素的API(BOM和DOM)此处的Web API特指浏览器提供的API(一组方法)，Web API在后面的课程中有其它含义

#### 4.掌握常见的浏览器提供的API的调用方式

5.ECMAScript - JavaScript的核心定义了javascript的语法规范

JavaScript的核心，描述了语言的基本语法和数据类型，ECMAScript是一套标准，定义了一种语言的标准与具体实现无关

##### BOM - 浏览器对象模型

一套操作浏览器功能的API

通过BOM可以操作浏览器窗口，比如：弹出框、控制浏览器跳转、获取分辨率等

##### DOM - 文档对象模型

一套操作页面元素的API

DOM可以把HTML看做是文档树，通过DOM提供的API可以对树上的节点进行操作

#### BOM的概念

一套操作页面元素的API

DOM可以把HTML看做是文档树，通过DOM提供的API可以对树上的节点进行操作

#### BOM的顶级对象window

window是浏览器的顶级对象，当调用window下的属性和方法时，可以省略window  
注意：window下一个特殊的属性 [window.name](http://window.name/)

#### 10.页面加载事件

#### 11.定时器

#### 12.location对象

location对象是window对象下的一个属性，时候的时候可以省略window对象

location可以获取或者设置浏览器地址栏的URL

#### navigator对象

* userAgent 通过userAgent可以判断用户浏览器的类型
* Platform 通过platform可以判断浏览器所在的系统平台类型.

#### DOM的概念

文档对象模型（Document Object Model，简称DOM），是W3C组织推荐的处理可扩展标志语言的标准编程接口。在网页上，组织页面（或文档）的对象被组织在一个树形结构中，用来表示文档中对象的标准模型就称为DOM。Document Object Model的历史可以追溯至1990年代后期微软与Netscape的“浏览器大战”，双方为了在JavaScript与JScript一决生死，于是大规模的赋予浏览器强大的功能。微软在网页技术上加入了不少专属事物，既有VBScript、ActiveX、以及微软自家的DHTML格式等，使不少网页使用非微软平台及浏览器无法正常显示。DOM即是当时蕴酿出来的杰作。

#### 15.模拟文档树结构

* 16.DOM经常进行的操作获取元素
* 动态创建元素
* 对元素进行操作(设置其属性或调用其方法)
* 事件(什么时机做相应的操作)

#### ****JavaScript 框架（jQuery）****

### 为什么使用 jQuery？

非常重要的理由就是：它能够兼容市面上主流的浏览器， IE 和FireFox， Google 浏览器处理 AJAX，创建异步对象是不同的，而jQuery 能够使用一种方式在不同的浏览器创建AJAX 异步对象。

其他优点：

写少代码,做多事情【writeless do more】

免费，开源且轻量级的js库，容量很小

兼容市面上主流浏览器，例如 IE，Firefox，Chrome

能够处理HTML/JSP/XML、CSS、DOM、事件、实现动画效果， 也能提供异步AJAX功能

文档手册很全，很详细

成熟的插件可供选择，多种js组件，例如日历组件（点击按钮显示下来日期）

出错后，有一定的提示信息

不用再在html里面通过

通过ID属性：getElementById()

通过 class属性：getElementsByClassName()

通过标签名：getElementsByTagName() 上面代码可以看出JavaScript 方法名太长了，大小写的组合太多了，编写代码效率，容易出错。jQuery 分别使用(“#id”) , (“.class 名”) , $(“标签名) 封装了上面的 js 方法。

### 2.jQuery是什么？

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库(或JavaScript框架)。

jQuery设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。它封装JavaScript常用的功能代码，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互。

jQuery的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口;具有高效灵活的css选择器，并且可对CSS选择器进行扩展;拥有便捷的插件扩展机制和丰富的插件。jQuery兼容各种主流浏览器，如IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+等

jQuery是优秀的JavaScript框架，能使用户更方便地处理HTML Documents、events、实现动画效果，并且方便地为网站提供Ajax交互

### DOM 对象

文档对象模型（Document Object Model，简称 DOM），是 W3C 组织推荐的处理可扩展标志语言的标准编程接口。 通过 DOM 对 HTML 页面的解析，可以将页面元素解析为元素节点、属性节点和文本节点，这些解析出的节点对象，即 DOM 对象。DOM 对象可以使用 JavaScript 中的方法。

### DOM 对象和jQuery 对象

DOM 对象是用JavaScript 语法创建的对象，也看做是 js 对象。

#### 4.1. DOM 对象转换jQuery 对象：

(*DOM*对象)方式，可以*DOM*对象转换为*jQuery*对象，转换为*jQuery*对象才可以使用*jQuery*中的提供的方法，操作*DOM*对象。一般情况下，在命名*jQuery*对象时，为了与*DOM*对象进行区分，习惯性的以 开头，这不是必须的。

#### 4.2. jQuery 对象转为DOM 对象

jQuery 对象本身为数组对象，该数组中的第 0 个元素即为该 jQuery 对象对应 的 DOM 对象。所以有两种方式可以获取到 DOM 对象：get(0) 方式与下标[0] 例：新建html 文件 jQueryToDom.html 1.页面添加 text ，button

### 选择器

选择器: 就是定位条件；通知jquery 函数定位满足条件的DOM 对象

#### 6.基本选择器

根据ID，class 属性，标签类型名定位HTML 元素，转为jQuery 对象.

****1. id 选择器****

语法：$(“#id”)

2. class 选择器

语法：$(“.class 名称”)

3. 标签选择器

语法：$(“标签名”)

#### . 所有选择器

语法：$(“\*”) 选取页面中所有DOM 对象。

#### . 组合选择器

组合选择器是多个被选对象间使用逗号分隔后形成的选择器，可以组合 id， class，标签名等。

语法：$(“#id, .class, 标签名”)

### 表单选择器

表单相关元素选择器是指文本框、单选框、复选框、下拉列表等元素的选择方式。该方法无论是否存在表单均可做出相应选择。表单选择器是为了能更加容易地操作表单，表单选择器是根据元素类型来定义的

### 过滤器

jQuery 对象中存储的DOM 对象顺序与页面标签声明位置关系

#### 基本过滤器

1. 选择第一个first, 保留数组中第一个DOM 对象

语法：$(“选择器:first”)

2. 选择最后个last, 保留数组中最后DOM 对象

语法：$("选择器:last")

3. 选择数组中指定对象

语法：$(“选择器:eq(数组索引)”)

4. 选择数组中小于指定索引的所有DOM 对象

语法：$(“选择器:lt(数组索引)”)

5. 选择数组中大于指定索引的所有DOM 对象

语法：$(“选择器:gt(数组索引)”)

# 事件绑定

选择器的使用需要进行事件的绑定，一般来说我们可以将事件绑定到一个按钮上去，通过按钮的点击来触发相应的事件响应。

# 入口函数

同时在jQuery中，这样的事件绑定是需要写在一个入口函数中去的，在的代码段中建立入口函数的语法是：

javascript

复制代码

$(function(){});

IMG_256在这里需要注意的是：在js中定义入口函数的window.onload方法和$(function)方法是有区别的，

具体如下：

window.onload只能定义一次，如果定义多次，后面的会将前面的覆盖掉

$(function)方法是可以定义多次的，后面的不会将前面的覆盖掉

# 样式控制

当我们对选择器的事件进行了绑定之后，我们就可以通过选择器进行相应元素的样式控制了，在这里通常的表现是css方法，通过css方法来对相应的元素样式进行修改。

# 数据可视化（echarts）

## 数据可视化

（1）什么是数据可视化

数据可视化主要目的：借助于图形化手段，清晰有效地传达与沟通信息。

数据可视化可以把数据从冰冷的数字转换成图形，揭示蕴含在数据中的规律和道理。

（2）数据可视化使用场景

目前互联网公司通常有这么几大类的可视化需求：

通用报表

移动端图表

大屏可视化

图编辑&图分析

地理可视化

（3）常见的数据可视化库

D3.js：目前WEB端评价最高的JavaScript可视化工具库（入手难）

ECharts.js：百度出品的一个开源JavaScript数据可视化库

Highcharts.js：国外的前端数据可视化库，非商用免费，被许多国外大公司所使用

AntV：蚂蚁金服全新一代数据可视化解决方案

**ECharts的基本使用并不难，你首先需要记住下面的五步曲，其次就是要会查文档。**

**ECharts使用五步曲：**

**步骤一：下载并引入echarts.js文件；（图表依赖这个js库）**

**步骤二：准备一个具备大小的DOM容器；（生成的图表会放入这个容器内）**

**步骤三：初始化echarts实例对象；（实例化echarts对象）**

**步骤四：指定配置项和数据(option)；（根据具体需求修改配置选项）**

**步骤五：将配置项设置给echarts实例对象；（让echarts对象根据修改好的配置生效）**

Echarts-基础配置

需要了解的主要配置：series xAxis yAxis grid tooltip title legend colorseries

系列列表。每个系列通过 type 决定自己的图表类型

大白话：图标数据，指定什么类型的图标，可以多个图表重叠。

xAxis：直角坐标系 grid 中的 x 轴

boundaryGap: 坐标轴两边留白策略 true，这时候刻度只是作为分隔线，标签和数据点都会在两个刻度之间的带(band)中间。

yAxis：直角坐标系 grid 中的 y 轴

grid：直角坐标系内绘图网格。

title：标题组件

tooltip：提示框组件

legend：图例组件

color：调色盘颜色列表

## ****第三阶段 服务端编程****

# node js基础

#### 1. Node.js的特性

(1) Nodejs语法完全是js语法，只要你懂js基础就可以学会Nodejs后端开发。

(2) Node.js超强的高并发能力，能够实现高性能服务器。

(3) 开发周期短、开发成本低、学习成本也低

#### 2. 浏览器和Node.js环境对比

下图中，在左边Chrome浏览器环境中，V8引擎用于解析JS代码，而Blink引擎是浏览器的排版引擎，用于处理网页的html、css和DOM操作。

而右边的Node环境中，只有V8引擎，用于处理JS代码，所以Node.js中，访问DOM和BOM是无法实现的。但Node.js因为有libuv中间层加持，能够实现浏览器所不能实现的一些功能。

## 3.Node.js的优势

1、Node.js语法完全是js语法，只要你懂js基础就可以学会Node.js后端开发

​ Node打破了过去JavaScript只能在浏览器中运行的局面。前后端编程环境统一，可以大大降低开发成本。

2、Node.js超强的高并发能力

​ NodeJs的首要目标是提供一种简单的、用于创建高性能服务器及可在该服务器中运行的各种应用程序的开发工具。

​ 首先让我们来看一下现在的服务器端语言中存在着什么问题。在Java、PHP或者.NET等服务器语言中，会为每一个客户端连接创建一个新的线程。而每个线程需要耗费大约2MB内存。也就是说，理论上，一个8GB内存的服务器可以同时连接的最大用户数为4000个左右。要让web应用程序支持更多的用户，就需要增加服务器的数量，而web应用程序的硬件成本当然就上升了。

​ NodeJs不为每个客户连接创建一个新的线程，而仅仅使用一个线程。当有用户连接了，就触发一个内部事件，通过非阻塞I/O、事件驱动机制，让Node.js程序宏观上也是并行的。使用Node.js，一个8GB内存的服务器，可以同时处理超过4万用户的连接。

##### 3、实现高性能服务器

​ 严格地说，Node.js是一个用于开发各种web服务器的开发工具。在Node.js服务器中，运行的是高性能V8 JavaScript脚本语言，该语言是一种可以运行在服务器端的脚本语言。

​ 那么，什么是V8 JavaScript脚本语言呢？该语言是一种被V8 JavaScript引擎所解析并执行的脚本语言。V8 JavaScript引擎是由Google公司使用C++语言开发的一种高性能JavaScript引擎，该引擎并不局限于在浏览器中运行。Node.js将其转用在了服务器中，并且为其提供了许多附加的具有各种不同用途的API。例如，在一个服务器中，经常需要处理各种二进制数据。在JavaScript脚本语言中，只具有非常有限的对二进制数据的处理能力，而Node.js所提供的Buffer类则提供了丰富的对二进制数据的处理能力。

​ 另外，在V8 JavaScript引擎内部使用一种全新的编译技术。这意味着开发者编写的高端的 JavaScript 脚本代码与开发者编写的低端的C语言具有非常相近的执行效率，这也是Node.js服务器可以提供的一个重要特性。

4、开发周期短、开发成本低、学习成本低

​ Node.js能花费最小的硬件成本，追求更高的并发，更高的处理性能。

**4.Node.js的特点**

我们先来看看Node.js官网上的介绍：

Node.js is a platform built on Chrome’s JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

其特点为：

它是一个Javascript运行环境

依赖于Chrome V8引擎进行代码解释

异步事件驱动

非阻塞I/O

轻量、可伸缩，适于实时数据交互应用

单进程，单线程（这里指主线程）

性能出众

## **5.NodeJS带来的对系统瓶颈的解决方案**

它的出现确实能为我们解决现实当中系统瓶颈提供了新的思路和方案，下面我们看看它能解决什么问题

### **1. 并发连接**

举个例子，想象一个场景，我们在银行排队办理业务，我们看看下面两个模型

1. 系统线程模型：这种模型的问题显而易见，服务端只有一个线程，并发请求（用户）到达只能处理一个，其余的要先等待，这就是阻塞，正在享受服务的请求阻塞后面的请求了
2. （2）多线程、线程池模型：

　这个模型已经比上一个有所进步，它调节服务端线程的数量来提高对并发请求的接收和响应，但并发量高的时候，请求仍然需要等待，它有个更严重的问题：

　回到代码层面上来讲，我们看看客户端请求与服务端通讯的过程：我们同样是要发起请求，等待服务器端响应；但是与银行例子不同的是，这次我们点完餐后拿到了一个号码，

　拿到号码，我们往往会在位置上等待，而在我们后面的请求会继续得到处理，同样是拿了一个号码然后到一旁等待，接待员能一直进行处理。

　等到饭菜做号了，会喊号码，我们拿到了自己的饭菜，进行后续的处理（吃饭）

　这个喊号码的动作在NodeJS中叫做回调（Callback），能在事件（烧菜，I/O）处理完成后继续执行后面的逻辑（吃饭），

　这体现了NodeJS的显著特点，异步机制、事件驱动

　整个过程没有阻塞新用户的连接（点餐），也不需要维护已经点餐的用户与厨师的连接

基于这样的机制，理论上陆续有用户请求连接，NodeJS都可以进行响应，因此NodeJS能支持比Java、PHP程序更高的并发量

　虽然维护事件队列也需要成本，再由于NodeJS是单线程，事件队列越长，得到响应的时间就越长，并发量上去还是会力不从心

## **6.NodeJS的优缺点**

1、优点：

高并发（最重要的优点）

适合I/O密集型应用

2、缺点：

不适合CPU密集型应用；CPU密集型应用给Node带来的挑战主要是：由于JavaScript单线程的原因，如果有长时间运行的计算（比如大循环），将会导致CPU时间片不能释放，使得后续I/O无法发起；

解决方案：分解大型运算任务为多个小任务，使得运算能够适时释放，不阻塞I/O调用的发起；

只支持单核CPU，不能充分利用CPU

可靠性低，一旦代码某个环节崩溃，整个系统都崩溃

原因：单进程，单线程

解决方案：（1）Nnigx反向代理，负载均衡，开多个进程，绑定多个端口；

（2）开多个进程监听同一个端口，使用cluster模块；

开源组件库质量参差不齐，更新快，向下不兼容

Debug不方便，错误没有stack trace

**7.适合NodeJS的场景**

1、RESTful API

这是NodeJS最理想的应用场景，可以处理数万条连接，本身没有太多的逻辑，只需要请求API，组织数据进行返回即可。它本质上只是从某个数据库中查找一些值并将它们组成一个响应。由于响应是少量文本，入站请求也是少量的文本，因此流量不高，一台机器甚至也可以处理最繁忙的公司的API需求。

2、统一Web应用的UI层

目前MVC的架构，在某种意义上来说，Web开发有两个UI层，一个是在浏览器里面我们最终看到的，另一个在server端，负责生成和拼接页面。

3、大量Ajax请求的应用

例如个性化应用，每个用户看到的页面都不一样，缓存失效，需要在页面加载的时候发起Ajax请求，NodeJS能响应大量的并发请求。　　总而言之，NodeJS适合运用在高并发、I/O密集、少量业务逻辑的场景。

4、适合I/O密集型的应用

如在线多人聊天，多人在线小游戏，实时新闻，博客，微博之类的。

不适合的场景有：cpu密集型的应用

如计算圆周率，视频解码等业务场景较多的。

# Ajax

### **一、定义**

1、什么是Ajax

Ajax：即异步 JavaScript 和XML。Ajax是一种用于创建快速动态网页的技术。通过在后台与服务器进行少量数据交换，Ajax可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。而传统的网页(不使用 Ajax)如果需要更新内容，必需重载整个网页面。

2、同步与异步的区别

同步提交：当用户发送请求时，当前页面不可以使用，服务器响应页面到客户端，响应完成，用户才可以使用页面。

异步提交：当用户发送请求时，当前页面还可以继续使用，当异步请求的数据响应给页面，页面把数据显示出来 。

3、ajax的工作原理

客户端发送请求，请求交给xhr，xhr把请求提交给服务，服务器进行业务处理，服务器响应数据交给xhr对象，xhr对象接收数据，由javascript把数据写到页面上

### **二、实现AJAX的基本步骤**

要完整实现一个AJAX异步调用和局部刷新,通常需要以下几个步骤:

1.创建XMLHttpRequest对象,即创建一个异步调用对象.

2.创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息.

3，设置响应HTTP请求状态变化的函数.

4.发送HTTP请求.

5.获取异步调用返回的数据.

6.使用JavaScript和DOM实现局部刷新.

**三、应用场景**

1、在线视频、直播平台等…评论实时更新、点赞、⼩礼物、…

2、会员注册时的信息验证，⼿机号、账号唯⼀

3、百度关键搜索补全功能

#### **3.promise**

1.什么是Promise

Promise 是异步编程的一种解决方案，其实是一个构造函数，自己身上有all、reject、resolve这几个方法，原型上有then、catch等方法。（ps:什么是原型：javascript中的原型与原型链\_傻小胖的博客-CSDN博客\_javascript原型和原型链）

1. Promise的特点

Promise对象有以下两个特点。

（1）对象的状态不受外界影响。Promise对象代表一个异步操作，有三种状态：pending（进行中）、fulfilled（已成功）和rejected（已失败）。只有异步操作的结果，可以决定当前是哪一种状态，任何其他操作都无法改变这个状态。这也是Promise这个名字的由来，它的英语意思就是“承诺”，表示其他手段无法改变。

（2）一旦状态改变，就不会再变，任何时候都可以得到这个结果。Promise对象的状态改变，只有两种可能：从pending变为fulfilled和从pending变为rejected。只要这两种情况发生，状态就凝固了，不会再变了，会一直保持这个结果，这时就称为 resolved（已定型）。如果改变已经发生了，你再对Promise对象添加回调函数，也会立即得到这个结果。这与事件（Event）完全不同，事件的特点是，如果你错过了它，再去监听，是得不到结果的。

3.为什么要使用promise

最重要也是最主要的一个场景就是[ajax和axios](https://so.csdn.net/so/search?q=ajax&spm=1001.2101.3001.7020" \o "ajax和axios)请求。通俗来说，由于网速的不同，可能你得到返回值的时间也是不同的，但是我们下一步要执行的代码依赖于上一次请求返回值，这个时候我们就需要等待，结果出来了之后才知道怎么样继续下去。

4.promise的三种状态

Promise对象有三种状态，他们分别是：

1.pending: 等待中，或者进行中，表示还没有得到结果

2.resolved(Fulfilled): 已经完成，表示得到了我们想要的结果，可以继续往下执行

3.rejected: 也表示得到结果，但是由于结果并非我们所愿，因此拒绝执行

这三种状态不受外界影响，而且状态只能从pending改变为resolved或者rejected，并且不可逆

### **promise的用法**

### **1.promise的实例方法**

①then()得到异步任务的正确结果

②catch()获取异常信息

③finally()成功与否都会执行(尚且不是正式标准)

# [Axios](https://so.csdn.net/so/search?q=Axios&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_32442973/article/details/_blank)介绍

1、Axios是什么？

Axios是一个基于promise的HTTP库，类似于jQuery的ajax，用于http请求。可以应用于浏览器端和node.js，既可以用于客户端，也可以用于node.js编写的服务端。

2、Axios特性

（1）支持Promise API

（2）拦截请求与响应，比如：在请求前添加授权和响应前做一些事情。

（3）转换请求数据和响应数据，比如：进行请求加密或者响应数据加密。

（4）取消请求

（5）自动转换JSON数据

（6）客户端支持防御XSRF

3、浏览器支持情况

Firefox、Chrome、Safari、Opera、Edge、IE8+。

### 4.Axios常用请求方法

方法列举：get, post, put, patch, delete等。

* get：一般用户获取数据
* post：一般用于表单提交与文件上传
* patch：更新数据（只将修改的数据推送到后端）
* put：更新数据（所有数据推送到服务端）

### delete：删除数据

### 5.Axios拦截器、错误处理与取消请求

#### **1、拦截器**

****什么是拦截器？****

在请求前或响应被处理前拦截他们，分为两种：请求拦截器与响应拦截器。

### 6.axios请求的封装与使用

#### **思路分析**

*axios* 请求的封装，无非是为了方便代码管理，我们可以使用抽离拆分的思想，将不同功能模块的接口处理成不同的模块，然后封装一个方法，专门用于数据交互。

# ECMAScript6详解

### **1.ECMAScript6简介**

* ECMAScript 6.0（以下简称 ES6）是 JavaScript 语言的下一代标准，已经在 2015 年 6 月正式发布了。

### 它的目标，是使得 JavaScript 语言可以用来编写复杂的大型应用程序，成为企业级开发语言

### 2.**ES6基本语法**

* ES标准中不包含 DOM 和 BOM的定义，只涵盖****基本数据类型、关键字、语句、运算符、内建对象、内建函数****等通用语法。
* 本部分只学习前端开发中ES6的必要知识，方便后面项目开发中对代码的理解。

##### 3.变量声明——let和const

(作用域) let和const是块级作用域，var是函数作用域

(变量提升) var声明变量存在变量提升

(暂时性死区) let和const声明变量存在暂时性死区

(重复声明) var声明的变量允许重复声明，后声明的变量覆盖先声明的变量，let和const不允许

(全局变量) var在浏览器全局环境下声明的变量会自动加入全局对象的属性，不可被delete。

(初始值) var和let不需要设置初始值，const声明变量必须要有初始值

(指针指向) const的指针不可变，var和let可变

## **4.ES6**[模板字符串](https://so.csdn.net/so/search?q=%E6%A8%A1%E6%9D%BF%E5%AD%97%E7%AC%A6%E4%B8%B2&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/m0_53321320/article/details/_blank)

在使用模板字符串前，我们拼接字符串变量使用 +

使用 ES6 提供的模板字符串，首先使用 `` 把字符串包起来，当要使用变量时 ，使用 ${变量}

5.ES6运行环境

所有主流浏览器已经几乎都支持es6的新功能，只有少部分不支持

Node是JavaScript的运行环境，他对es6的支持度更高，除 了那些默认打开的功能，还有一些语法功能已经实现了，但是默认没有打开。