爱创课堂前端培训

# 大数据可视化与git、模块化开发

第1天课堂笔记（本课程共2天）

班级：北京前端训练营19期

讲师：彭帅伟

日期：2018年12月25日

彭帅伟老师

QQ: 284337908

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

目录

[大数据可视化与git、模块化开发 1](#_Toc11640)

[复习： 3](#_Toc15043)

[一、大数据可视化 6](#_Toc28409)

[二、git 7](#_Toc323)

[2.1 git指令 7](#_Toc13333)

[2.2介绍git 8](#_Toc15169)

[2.3 使用git 8](#_Toc28870)

[2.4建立信任关系 9](#_Toc12616)

[2.5 创建项目 10](#_Toc17922)

[2.6 上传代码 12](#_Toc22721)

[2.7 预览项目 12](#_Toc29360)

[2.8 克隆项目 12](#_Toc8083)

[复习： 13](#_Toc5447)

[三、Seajs介绍 14](#_Toc14784)

[3.1 体验seajs 14](#_Toc2955)

[3.2 定义模块 15](#_Toc10422)

[3.3 require模块 16](#_Toc26880)

[3.4 加载具有id的模块 17](#_Toc12919)

[3.5 暴露接口 18](#_Toc511)

[3.5 模块信息 19](#_Toc28364)

# 复习：

严格模式：

ES5拓展了严格模式，可以使我们的代码更加的安全可靠

开启方式：

在js第一行加入“use strict”, 此时代码将处于“全局严格模式”

如果在某个函数中的第一行加入“use strict”， 当函数执行的时候，此时将处于“局部严格模式”

严格模式与正常模式之间的区别：

在严格模式下，定义变量必须使用var进行声明

在严格模式下，定义数值的时候，不可以使用八进制

在严格模式下，不可以使用arguments.callee

在严格模式下，不能使用eval函数

在严格模式下，不能使用with

在严格模式下，不能使用保留字定义变量

在严格模式下，定义对象中属性不可以是同名的

在严格模式下，全局函数的作用域是undefined

在严格模式下，函数中的参数也不可以同名

特性：

设置单一属性特性：

Object.defineProperty(obj, property, options)

obj: 要设置的对象

property: 要设置的属性

options: 描述特性对象

// 配置值

value: “”

// 配置可修改性

writable: true | false

// 配置可枚举性

enumerable: true | false

// 可配置性

configurable: true | false

设置多个属性特性：

Object.defineProperties(obj, options)

obj: 要设置的对象

options: 特性对象

key: 要设置的属性

value: 描述特性对象

特性方法：

get方法用来获取值的

没有参数

返回值就是要获取的属性值

作用域是当前对象

千万不要在该方法中，获取该属性值，通常是获取该属性的一个备用属性

set方法用来设置值的

参数就是要设置的属性值

没有返回值

作用域当前对象

千万不要在方法中设置该属性，通常是设置该属性的一个备用属性

原型方法：

ES5为原型拓展了几个方法

isPrototypeOf:

原型对象的方法

判断原型对象是否是参数对象的原型

参数就是实例化对象

会查找整个原型链

getPrototypeOf:

用于获取原型对象

是对象的静态方法

setPrototypeOf:

该方法用于设置某个对象的原型对象

接受两个参数

第一个参数要设置的对象  
 第二个参数：新的原型对象 （可以是null也可以是一个对象）

对象的拓展：

取消对象的可拓展性：

Object.preventExtensions(obj)

当取消了对象可拓展性之后， 不能添加属性，可以删除属性，可以修改属性值

查看对象是否被取消了可拓展性:

Object.isExtensible(obj)

封闭对象：

Object.seal(obj)

当封闭了对象之后，不能添加属性， 不能删除属性，可以修改属性值

查看对象是否被封闭：

Object.isSealed(obj)

冻结对象：

Object.freeze(obj)

当冻结了对象之后， 不能添加属性， 不能删除属性， 不能修改属性值

查看对象是否被冻结：

Object.isFrozen(obj)

创建对象的新方式：

Object.create(prototype, options)

接受两个参数：

第一个参数是一个对象 （可以是null）

第二个参数是一个对象 （可以省略）

# 一、大数据可视化

简单来说： 做图表

官网：<https://www.hcharts.cn/>

下载：

第一步：



第二步：



第三步：



出现弹出框，我们不需要理会，稍后即可自动下载

# 二、git

linux将代码发布到网站了，开源了，因此很多开发者不断的贡献代码，这样代码越来越多了，就有了管理代码的问题了，所以创建git，就是用来管理linux代码的，后来发现很好用，所以越来越多的项目开始放在git上了，后来项目越来越多了，其他系统也就开始支持git了（windows， mac）

我们要使用git就要安装客户端

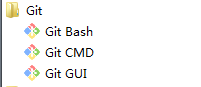


双击进行下一步安装

安装完成，提供了控制台

Git Bash 为linux家族（unix，mac等）使用的

Git CMD 为window使用的



由于git为linux提供的，所以建议我们使用linux指令

## 2.1 git指令

cd目录，打开目录

ls 查看当前目录文件

pwd 查看当前目录所在文件的系统路径

mkdir 文件名 创建文件夹

touch 创建文件

echo 内容 >> 文件

小案例：C:\Users\Dr.z\Desktop\2018年12月25日大数据与git\案例\git在当前目录下，创建一个demo文件夹，并且创建readme.txt文件写入爱创课堂

cd 目录

pwd 查看当前文件目录

mkdir 创建文件夹

ls 查看当前目录文件

cd 文件

touch 创建文件

echo 内容 >> 文件中

## 2.2介绍git

在git中有三类文件

第一类是未纳入缓存的文件

这类文件， 一旦删除就再也无法找到

第二类文件是纳入缓存的文件

这类文件， 可以通过git来找到

第三类文件纳入版本库的文件

这类文件，可以通过计算的各个位置来找到

## 2.3 使用git

通过git init来初始化项目文件

objects 用来存放纳入缓存的文件

refs 用来存放纳入版本库的文件

我们可以通过git指令来实现三类文件的切换

git add 文件

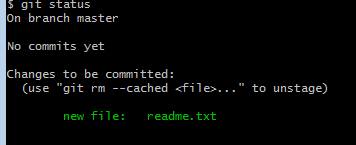
实现将未纳入缓存的文件纳入缓存中 （变为二类文件）

git add \*

实现将所有未纳入缓存的文件纳入缓存中

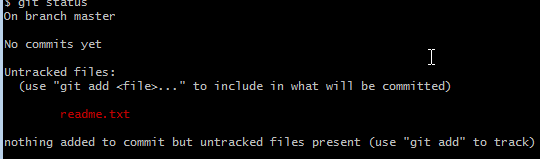
git status

查看当前文件的状态



git rm --cached 文件

实现将纳入缓存的文件变为第一类文件 （未纳入缓存）



一旦纳入缓存之后， 将有机会纳入版本库

通过git commit -m “说明”

说明一定要有语义化

一旦纳入版本库，此时文件可以通过计算机的各个位置来找到，但是如果计算机坏掉了，文件就无法被找到，

所以， 我们要将文件上传到云端（服务器）， 即使计算机坏掉，仍然可以找到文件

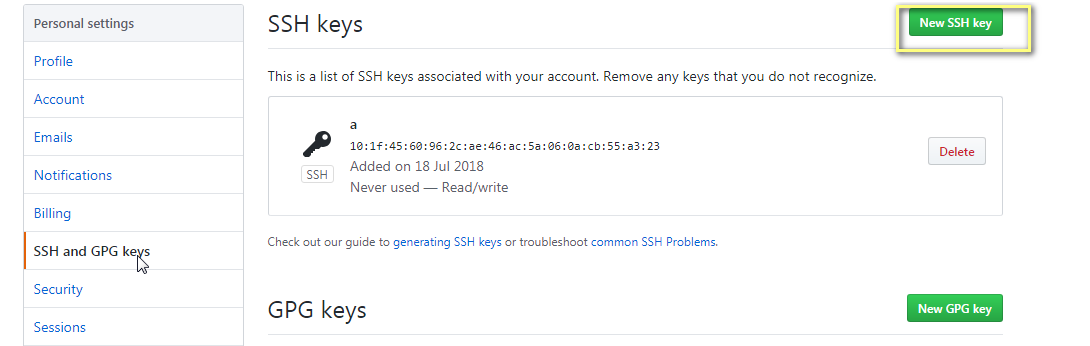
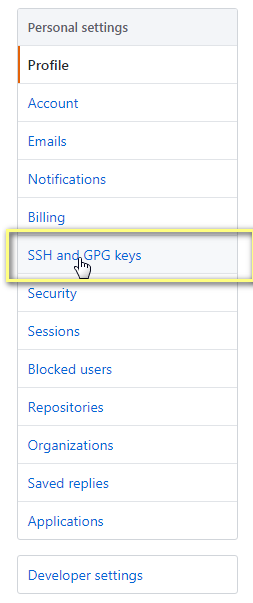
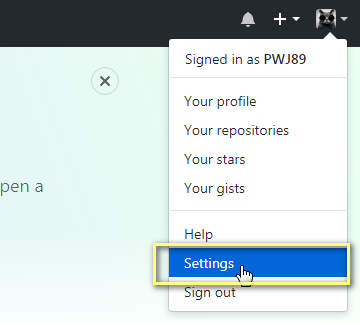
## 2.4建立信任关系

执行指令 ssh-keygen -t rsa -C 邮箱

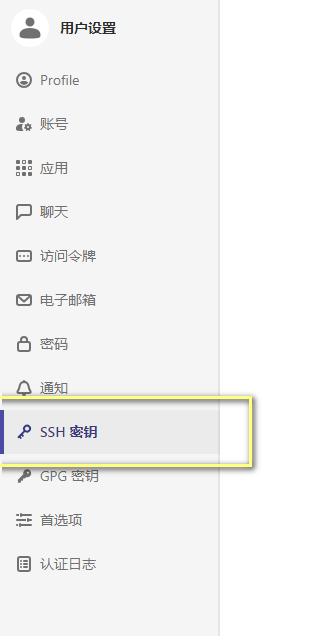
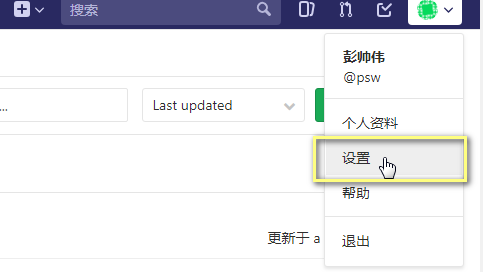
进入目录C:\Users\think\.ssh

打开id\_rsa.pub文件，复制里面的内容

进入github

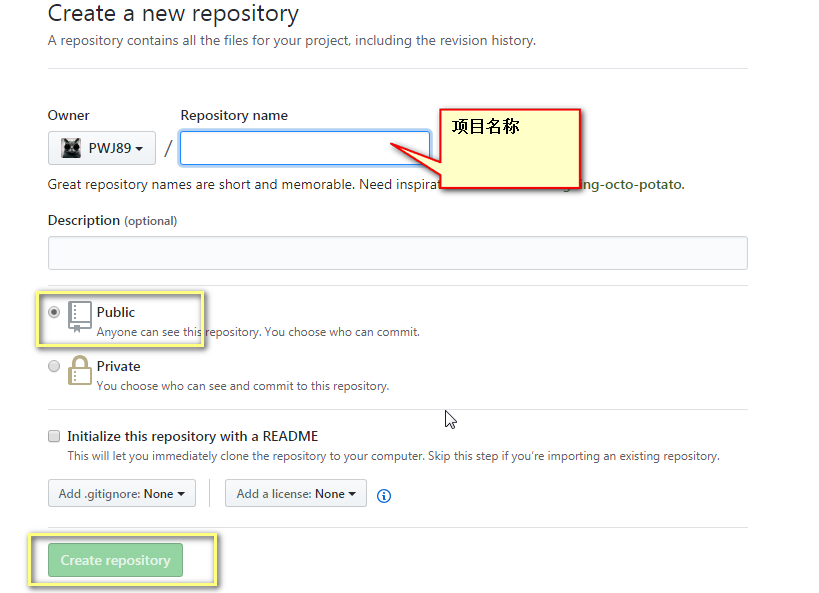
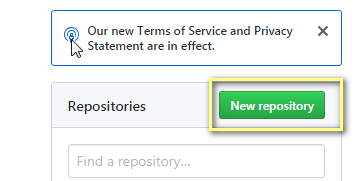


进入gitlab

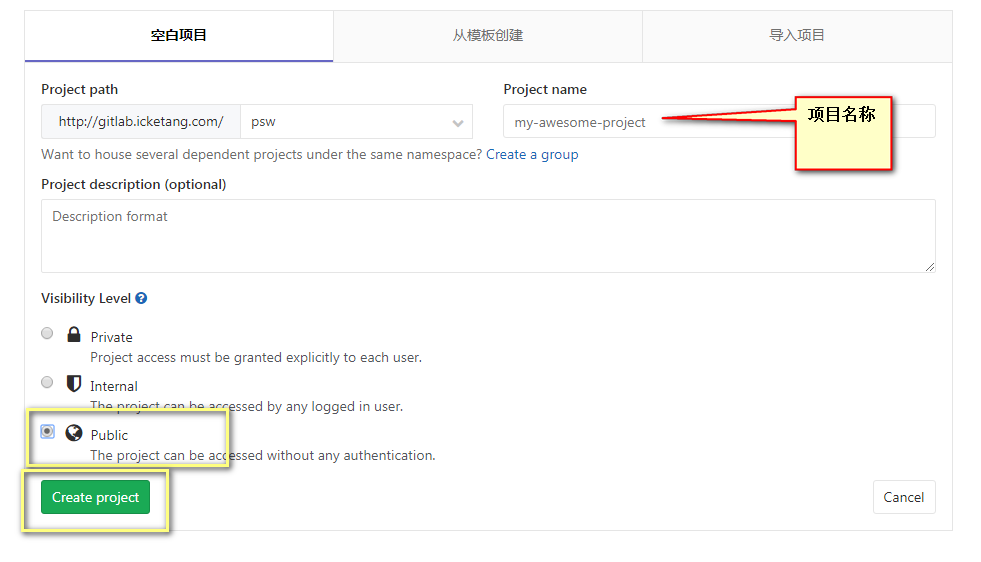
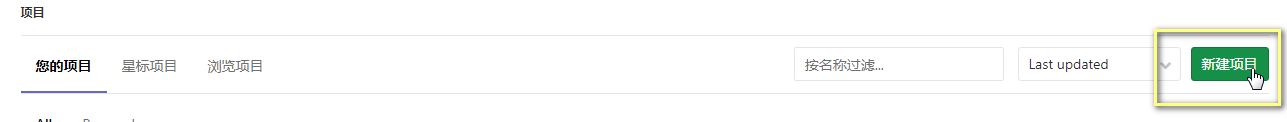


## 2.5 创建项目

进入github



进入gitlab中



## 2.6 上传代码

建立连接：

git remote add origin https://github.com/PWJ89/demo1.git

上传代码：

git push -u origin master

## 2.7 预览项目

git提供了一个可以预览项目的网址http://htmlpreview.github.io/

在输入框中，输入首页模块地址即可预览项目

## 2.8 克隆项目

通过git clone 文件可以下载别人的项目

## 复习：

git介绍：

git中有三类文件

第一类是未纳入缓存文件

这类文件， 一旦删除， 就再也无法找到

第二类文件纳入缓存的文件

这类文件，可以通过git来找到

第三类文件纳入版本库的文件

这类文件， 可以通过计算机的各个位置来找到

git指令：

cd 目录 打开目录

ls 查看当前目录中的文件

pwd 查看文件所在系统的路径

mkdir 文件名 创建文件夹

touch 文件

echo 内容 >> 文件

# 三、Seajs介绍

是国内的一套模块化开发框架，有淘宝工程师王伯写的，有中文文档，学习简单

开源代码库

<https://github.com/seajs/seajs>

Seajs出现的比较晚，因此借鉴了，nodejs的commonjs规范（加载时同步的），但是前端使用文件是要异步加载文件的，加载完成之后才能使用，又借鉴了前端的AMD规范，seajs的规范称之为cmd规范，这套框架在使用的时候，建议我们使用commonjs规范

## 3.1 体验seajs

如果想要使用seajs需要在页面中引入该文件

当引入seajs之后，会向全局暴露两个变量：seajs, define

在模块外部引入一个文件，要通过use方法来引入

第一个参数是一个数组， 数组中的每一项都是一个模块的文件地址（如果引入的只有一个文件，可以省略数组）

第二个参数是一个回调函数

函数中的参数，就是前面模块向外暴露的功能（一一对应）

seajs引入的文件路径是以seajs所在的文件目录（seajs所在的目录就是根目录）

通常我们将seajs放在最外面（与index.html文件是同一级）

为了引入seajs方便

seajs对js文件比较敏感，因为可以省略.js后缀

seajs:

|  |
| --- |
| 1. seajs本身是一个对象，对象中包含各种功能函数以及相关信息 2. seajs.config: 是规定如何配置模块 3. seajs.use：是规定如何使用模块 |

define:

|  |
| --- |
| 1. define: 是用来定义模块 |

## 3.2 定义模块

在seajs中提供了define方法用来定义模块，可以传递三个参数， 因此提供了六种定义模块的方式：

传递一个参数：

可以是值类型（数字，字符串，bool）

这些数据会做为接口直接暴露

可以是引入类型（数组，对象）

这些数据也会作为接口向外暴露

可以传递一个函数（90%以上都是使用这种方式，因为它是基于commonjs规范的）

此时函数中有三个参数：

require: 引入其它模块的

exports: 向外暴露功能的对象

module: 模块的信息

作用域是向外暴露功能的对象

此时，向外暴露功能一定是基于commonjs规范

exports和module是定义向外暴露功能的对象

传递两个参数：

第一个参数可以是字符串， 表示模块的id

此时，seajs.use方法不能直接使用该模块 （要配合require方法）

第一个参数还可以是数组，表示模块的依赖集合（模块依赖集合中的文件，不管有没有使用，都会被加载）

第二个参数是回调函数（与第三种定义模块方式中传递的函数表现形式一致）

传递三个参数：

第一个参数是字符串， 表示模块的id

第二个参数是一个数组， 模块的依赖集合

第三个参数是回调函数（与第三种定义模块方式中传递的函数表现形式一致）

举例：

|  |
| --- |
| 1. // 第一种定义模块的方式 2. // 传递数字 3. define(1); 4. // 传递字符串 5. define("abc"); 6. // 传递布尔值 7. define(true); 8. // 第二种方式 9. // 传递数组 10. define(["red", "blue", "green"]); 11. // 传递对象 12. define({ 13. num: 100, 14. color: "red" 15. }) 16. // 第三种方式 17. define(function(require, exports, module) { 18. console.log(this, arguments); 19. console.log(require); 20. console.log(exports); 21. console.log(module); 22. this.color = "red"; 23. }) 24. // 第四种方式 25. // define("abc", function() { 26. define("modules/main", function() { 27. console.log(this, arguments); 28. }) 29. // 第五种方式 30. define(["modules/dom"], function() { 31. console.log(this, arguments); 32. }) 33. // 第六种方式 34. // define("abc", ["modules/dom"], function() { 35. define("modules/main", ["modules/dom"], function() { 36. console.log(this, arguments); 37. }) |

## 3.3 require模块

在一个模块中引入其它模块要分为两步：

第一步通过require引入该模块文件

第二步从模块文件中读取该模块

1 require不能被简写

|  |
| --- |
| 1. // 1 require不能被简写 2. define(function(req, exports, module) { 3. var dom = req("module/dom"); 4. console.log(dom); 5. }) |

2 require不能被修改

1 require不能被赋值

2 require不能赋值给其它变量

3 require不能在子函数中作为参数传递

4 require在子函数中不能被修改（以上3点）

|  |
| --- |
| 1. // 2 require不能被修改 2. define(function(require, exports, module) { 3. // 1 require不能被赋值 4. // require = 123; 5. // 2 require不能赋值给其它变量 6. // var a = require; 7. // 3 不能在子函数中作为参数传递 8. function demo(require) { 9. // 1 require不能被赋值 10. // require = 123; 11. // 2 require不能赋值给其它变量 12. var a = require; 13. // 引入其它模块文件 14. var dom = a("module/dom"); 15. console.log(dom); 16. } 17. demo(require); 18. // var dom = require("module/dom"); 19. // console.log(dom); 20. }) |

3 require只能是字符串，不能拼接

|  |
| --- |
| 1. // 3 不能拼接 2. define(function(require, exports, module) { 3. // 引入其它模块文件 4. // var dom = require("module" + "/" + "dom"); 5. var dom = require("module/dom"); 6. console.log(dom); 7. }) |

## 3.4 加载具有id的模块

加载具有id的模块要分为两步走：

第一步通过模块的依赖集合加载模块文件

第二步通过require指定id

|  |
| --- |
| 1. define(["module/dom"], function(require, exports, module) { 2. // 引入dom.js 3. var dom = require("myId"); 4. console.log(dom); 5. }) |

如果在一个模块文件中，存在两个没有id的模块，那么后面的模块会覆盖前面的模块

如果在一个模块文件中，存在两个相同id的模块，前面会覆盖后面的模块

如果在一个模块文件中，存在多个不同id的模块，是可以同时存在，引入的文件是以require指定的id为准

## 3.5 暴露接口

|  |
| --- |
| 1. // 定义模块 2. define(function(require, exports, module) { 3. // 第一种向外暴露功能的方式 4. exports.a = 123; 5. // 这种方式，是绝对不允许使用 6. exports = { 7. color: "red" 8. } 9. // 第二种方式 10. module.exports.b = 123; 11. // 3 12. module.exports = 123; 13. // 4 14. // 这种方式会覆盖上面向外暴露的数据 15. module.exports = { 16. color: "red" 17. } 18. // 5 19. // 这种暴露功能的方式，如果外部要使用，要作为方法来使用 20. module.exports = function() { 21. console.log("success"); 22. } 23. // 6 24. return 123; 25. // 7 26. return { 27. a: 1, 28. b: 2 29. } 30. // 8 31. // 外部如果要使用，要作为方法来使用 32. return function() { 33. console.log("success"); 34. } 35. }) 36. // 9 37. define(1); 38. define("abc"); 39. // 10 40. define({ 41. color: "red" 42. }) |

## 3.5 模块信息

每一个模块都有一个对象来保存模块的信息

id: 表示模块的id

uri: 模块的文件路径，默认情况下（没有显示的id）与模块的id是相同

dependencise: 是模块的依赖集合，是一个数组， 数组中每一项都是依赖集合中的成员

deps: 根据模块的依赖集合产生

exports: 向外暴露功能的对象

status: 模块的状态

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |