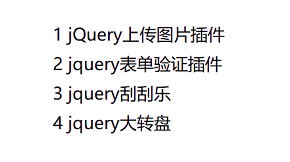
## HUI：管理后台框架

## 明天讲ajax，尽量多讲

# Less



# Redis？？？？？？？？？？！！！

## 原生js的状态

## Animate看看

## Mui

## JQ队列？？？

## 原生操作表单？？？？

## 队列动画？？？

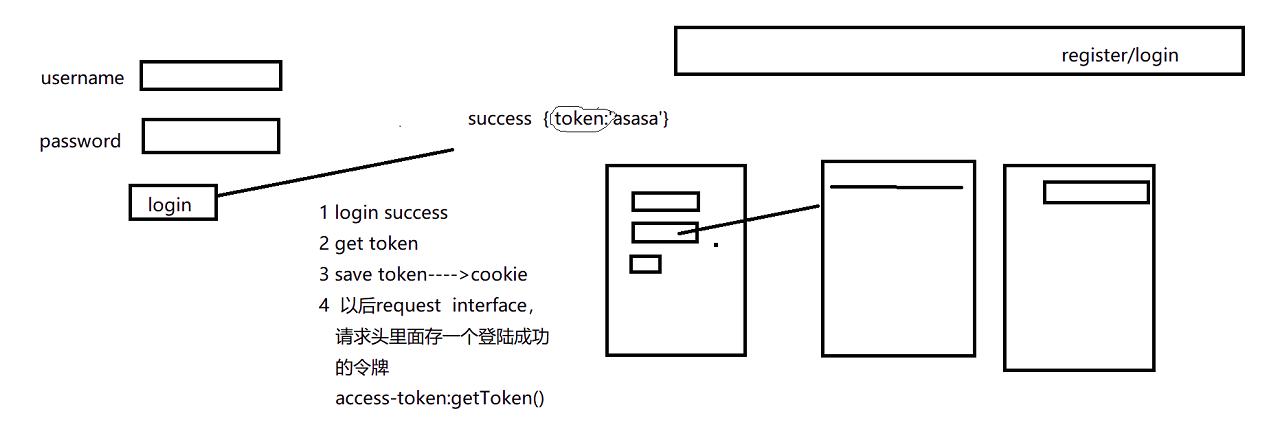
## Super admin：管理后台

## 2020-01-09：a、轮播图 b、桌面系统 c、dom操作表格

## 操作数组方法

# 跨域

## Cookie购物车：（sessionStorage+localStorage）购物车



## Chatcode+keycode？？？？？？明天分享

# XML???

## 函数节流：(输入框搜索)使的一定时间内只触发一次函数。原理是通过判断是否到达一定时间来触发函数

## 防抖：(滚动条)将几次操作合并为一个操作进行

函数节流和防抖区别：函数节流不管事件触发有多频繁，都会保证在规定时间内一定会执行一次真正的事件处理函数，而函数防抖只是在最后一次事件后白触发一次函数（比如在页面的无限加载场景下，我们需要用户在滚动页面时，每隔一段时间发一次 Ajax 请求，而不是在用户停下滚动页面操作时才去请求数据。这样的场景，就适合用节流技术来实现。）

## 单例模式？？？

## 数组深拷贝和浅拷贝？？？

全选/反选/不选

若元素节点返回标签名，

若是文本节点返回文本内容

过滤空白节点

# 路线：JS---Jqurey-----nodejs-----vue

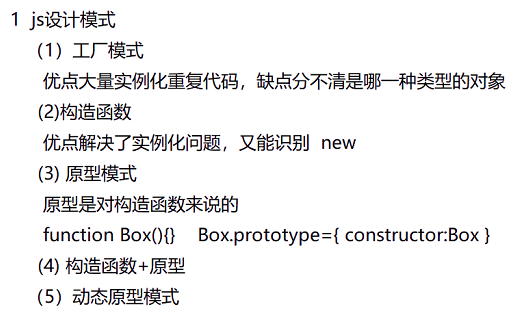
## 弄一套试试，后台数据调用！！！！！

## 注册登录全穿起来

## 小雪花

# JS设计模式

1. 工厂模式
2. 构造函数
3. 原型模式
4. 构造原型+原型
5. 动态原型模式



# JS继承

1. 原型链继承

一个对象的原型指向另一个对象的实例化（Per.prototype = new Person()）

1. 借用构造函数继承
   1. 通过call和apply继承
2. 组合继承(原型链继承+借用构造函数继承)
3. 原型式继承
4. 寄生式继承
5. 寄生组合式继承

# 匿名函数自执行

# 私有属性：

**function Box(){**

**This.name = ‘xiaowei’ //是有属性**

**This.run = function(){} //私有方法**

**}**

# 静态属性：

**function Box(){}**

**Box.sex = ‘男’ //Math.PI**

# 静态方法

**Box.say = function(){} //Math.random**

**原型上是实例共享的，是半实例属性的**

# 静态私有属性

**(function(){**

**var name = ‘xiaowei’ //静态是有属性**

**function a(){} //静态私有方法**

**Person = function(){**

**this.run = function(){**

**return name** //把私有属性推出去

**}**

**this.execa = function(){**

**a()** //调用上面的方法

**}**

**}**

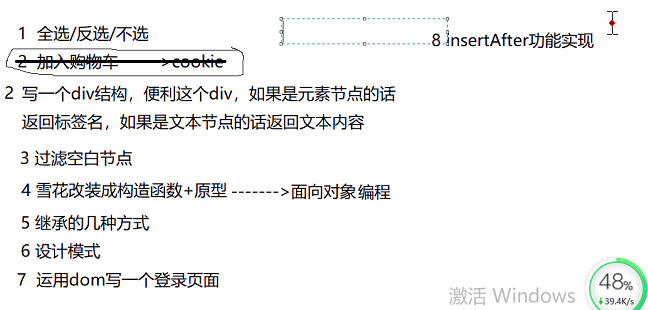
**})()**

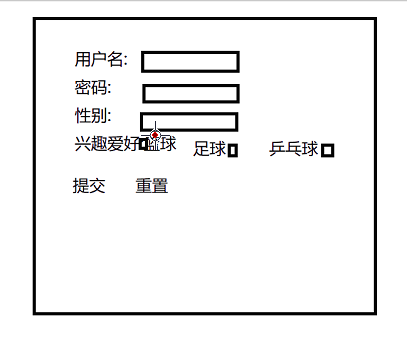
**var p = new Person(){}**

**p.run()**

**P.execa()**

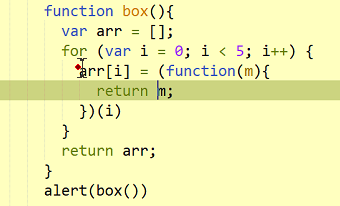
# 作业

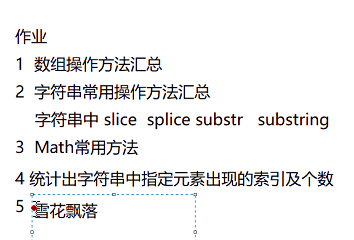




## 轮播图-->构造函数+原型

## Ul--li 点击打印当前索引

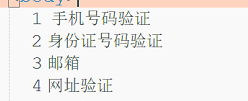




## 构造函数+原型-->弹框

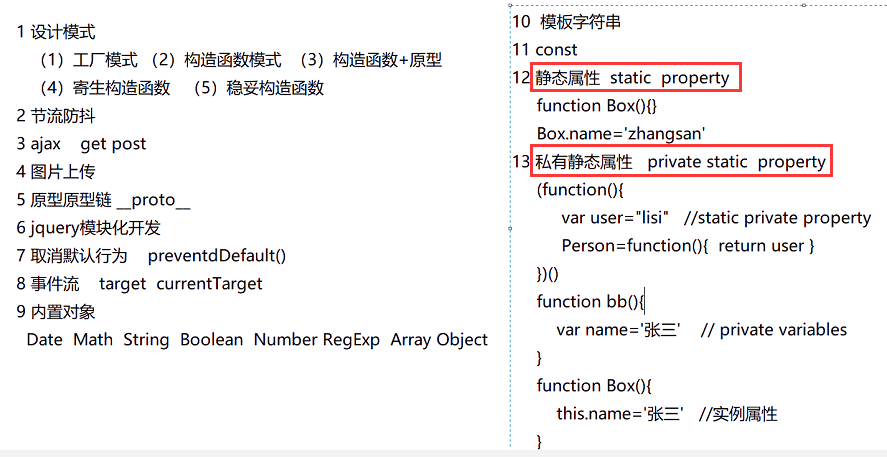


# 正则验证

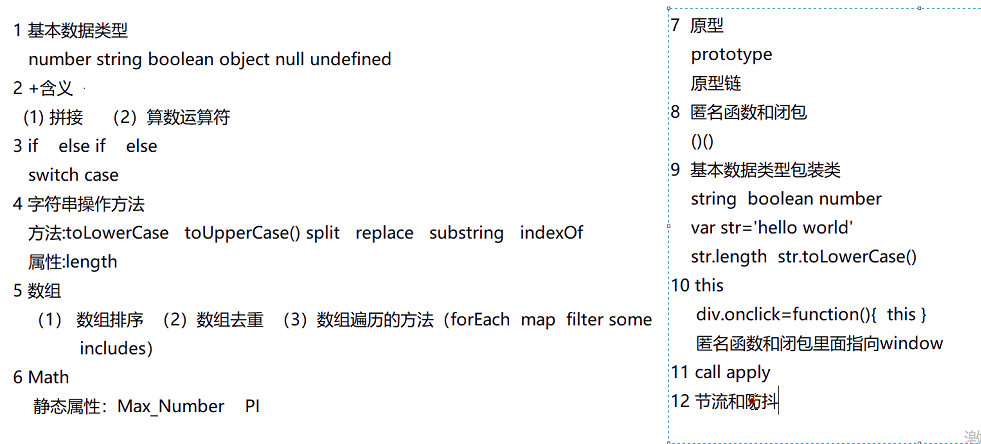


姓名 密码 用后台试试手

# 2020-01-13



# 2020-01-10

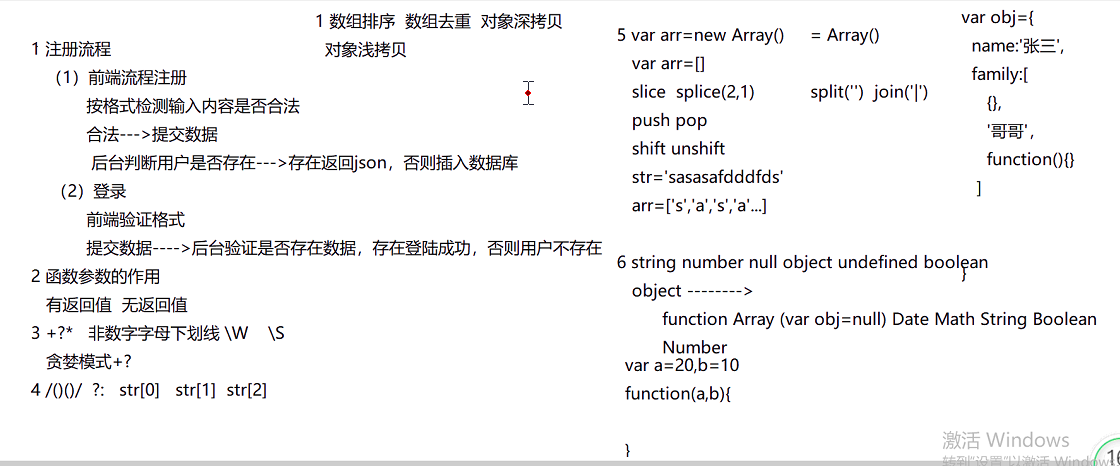


节流：比如输入框(不能一直输入一直**请求接口**)

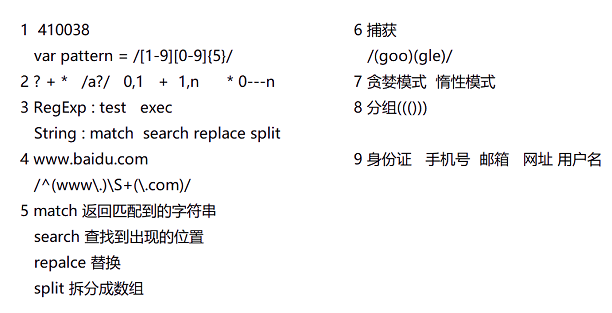
防抖：滑动条(用户体验)

# 2019-12-23

**{ }中必须是键值对**



# 2019-12-20小知识点





# 2019-12-18知识点

