**前端面试题**

### 1.如何评价下面这段代码？是否有优化空间？如果有请写出案例

var sayHello = function( personal ){  
 if ( personal instanceof Li ){  
 console.log( '嘿，你好' );  
 }else if ( personal instanceof Dan ){  
 console.log( '吃早饭了么' );  
 }else if (personal instanceof Arthur){  
 console.log('你今天看起来真漂亮')  
 }  
};  
  
var Li = function(){};  
var Dan = function(){};  
var Arthur = function(){};  
  
sayHello( new Li() ); // 嘿，你好

sayHello( new Dan() ); // 吃早饭了么

sayHello( new Arthur() ); // 你今天看起来真漂亮

答：

该段代码主要的逻辑写在被调用部分，耦合性较强，而且一般不建议instanceof进行比较。

可以通过对象原型方法封装函数的方式进行优化，优化代码如下：

var sayHello = function( personal ){

personal.sayHello();

};

var Li = function(){};

Li.prototype.sayHello = function(){

console.log( '嘿，你好' );

}

var Dan = function(){};

Dan.prototype.sayHello=function(){

console.log('吃早饭了么')

}

var Arthur = function(){};

Arthur.prototype.sayHello = function(){

console.log('你今天看起来真漂亮')

}

sayHello( new Li() ); // 嘿，你好

sayHello( new Dan() ); // 吃早饭了么

sayHello( new Arthur() ); // 你今天看起来真漂亮

如果只是为了调用sayHello方法进行不同类型的打印，修改方法如下：

function sayHello(info){

console.log(info);

}

sayHello('嘿，你好');

sayHello('吃早饭了么');

sayHello('你今天看起来真漂亮');

### 2、平时的一些学习方法？以及如何平衡工作和学习上的关系？目前关注哪些技术方向（技术栈）

1学习方法：网上视频学习（腾讯，网易云课堂），边学边练习。

1. 根据工作需要进行有目的性的有针对性的业余学习，从而提高工作效率。
2. 目前重点关注和学习的是前端的vue.js 和Es6多点，其他流行技术只做简单了解

**3、遇到过哪些技术难题？如何解决的？**

由于本人原来做的是java后台开发，目前转向前端技术学习。在自学前端过程遇到的小问题居多，这里就不一一说明了。

解决办法一般如下：首先如果是报错性的异常问题，解决思路就是先看报错信息，结合提示信息去检查代码，如果解决不了就网查资料，如果接近一个小时还解决不了就请教大牛。

如果是没有接触过的技术难题一般就是先网上查资料，比如官网，百度，技术论坛等。最后就是求助身边丰富经验的人指导一二。

4、按照调用实例，实现下面的Person方法：

Person("Li");  
// 输出： Hi! This is Li!

Person("Dan").sleep(10).eat("dinner");  
// 输出：  
// Hi! This is Dan!  
// 等待10秒..  
// Wake up after 10  
// Eat dinner~

Person("Jerry").eat("dinner").eat("supper");  
// 输出：  
// Hi This is Jerry!  
// Eat dinner~  
// Eat supper~  
  
Person("Smith").sleepFirst(5).eat("supper");  
// 输出：  
// 等待5秒  
// Wake up after 5  
// Hi This is Smith!  
// Eat supper

代码如下：

let TimeTask = function(time,fun){

this.time = time;

this.fun = fun;

}

let Flow = function(){}

Flow.prototype={

\_tasks:[],

\_callTimeTask:function(task,callback){

return function(){

setTimeout(function(){

task.fun();

callback();

},task.time)

}

},

\_finishTask:function(){},

add:function(time,fun){

this.\_tasks.push(new TimeTask(time,fun))

},

addFirst:function(time,fun){

this.\_tasks.unshift(new TimeTask(time,fun))

},

runTask:function(){

let fuse = this.\_finishTask;

while(this.\_tasks.length>0){

let task = this.\_tasks.pop();

fuse = this.\_callTimeTask(task,fuse)

}

fuse.call()

},

run:function(){

let that = this;

setTimeout(function(){

that.runTask();

},0);

}

}

let Person = function(name){

this.sleep = function(time){

this.flow.add(time\*1000,()=>{console.log("Wake up after " + time)});

return this;

}

this.sleepFirst = function(time){

this.flow.addFirst(time\*1000,()=>{console.log("Wake up after "+time)});

return this;

}

this.eat = function(food){

this.flow.add(0,()=>{console.log("Eat " + food + "~")})

return this;

}

this.flow = new Flow();

this.flow.add(0,()=>{console.log("Hi! this is " + name + "!")});

this.flow.run();

return this;

}

// Person("Li");

// Person("Dan").sleep(10).eat("dinner");

// Person("Jerry").eat("dinner").eat("supper");

// Person("Smith").sleepFirst(5).eat("supper");