验证码功能的修改

Login.vue代码：

```
        handlerPopup(captchaObj){
          // 验证码成功的回调
          let _this = this;
          captchaObj.onSuccess(function () {
              var validate = captchaObj.getValidate();
              _this.validateResult = true;
          });
          // 将验证码加到id为captcha的元素里
          captchaObj.appendTo("#geetest");
        }
```

views.py

```
from rest_framework.response import Response
from rest_framework.views import APIView
import random
from luffy.libs.geetest import  GeetestLib
class VerifyCode(APIView):
    gt = None
    """验证码类"""
    def get(self,request):
        """获取验证码"""
        user_id = random.randint(1, 100)
        APP_ID = "884b024377529d6ba4d2f07d227879df"
        APP_KEY = "28e7f92b7c66f718d65ede8feb26f477"
        gt = GeetestLib(APP_ID,APP_KEY)
        status = gt.pre_process(user_id)
        data = gt.get_response_str()
        return Response(data)

from rest_framework.generics import CreateAPIView
from .models import User
from .serializers import UserModelSerializer
class UserAPIView(CreateAPIView):
    """用户管理"""
    queryset = User.objects.all()
    serializer_class = UserModelSerializer

from rest_framework import status
from luffy.libs.yuntongxun.sms import CCP
from django_redis import get_redis_connection
class SMSCodeAPIView(APIView):
    def get(self,request):
        # 1. 通过查询字符串获取手机号码
```

```
        mobile = request.query_params.get("mobile")
        # 2. 发送短信之前验证码验证一下手机号码
        try:
            User.objects.get(mobile=mobile)
            return Response({"message": "当前手机号已经被注册过"},
status=status.HTTP_400_BAD_REQUEST)
        except:
            pass
        redis = get_redis_connection("sms_code")
        if redis.get("times_%s" % mobile):
            return Response({"message": "当前手机号已经在一分钟内发送过短信"},
status=status.HTTP_400_BAD_REQUEST)

        # 3. 使用手机号码发送短信验证码
        # 生成一个短信验证码
        sms_code = "%04d" % random.randint(0, 9999)
        ccp = CCP()
        result = ccp.send_template_sms(mobile,[sms_code,"5分钟"],1)

        if result == 0:
            # 发送短信成功，保存短信验证码到redis数据库中
            # 开启管道操作
            pl = redis.pipeline()
            pl.multi() # 接下来会在管道中执行多条命令
            # setex(变量名,有效期[秒],值 )
            SMS_EXPIRE_TIME = 5 * 60 # 短信验证码的有效期
            SMS_TIMES = 60  # 短信发送的间隔时间
            # 把原来立即执行的命令放置到管道
            pl.setex("sms_%s" % mobile, SMS_EXPIRE_TIME, sms_code)
            pl.setex("times_%s" % mobile, SMS_TIMES, 1)

            # 统一执行管道中的命令
            pl.execute()

        # 4. 响应数据给客户端
        return Response({"message":result},status=status.HTTP_200_OK)
```

订单模型

```
from django.db import models

# Create your models here.
from users.models import User
from courses.models import Course
class Order(models.Model):
    """订单记录"""
    status_choices = (
        (0, '未支付'),
        (1, '已支付'),
        (2, '已取消'),
```

```
    )
    total_price = models.DecimalField(max_digits=6, decimal_places=2, verbose_name="订单
总价", default=0)
    order_number = models.CharField(max_length=16,verbose_name="订单号")
    order_status = models.SmallIntegerField(choices=status_choices, default=0,
verbose_name="订单状态")
    order_desc = models.CharField(max_length=120,verbose_name="订单描述")
    created_time = models.DateTimeField(verbose_name="订单生成时间", auto_now_add=True)
    pay_time = models.DateTimeField(verbose_name="订单支付时间", auto_now_add=True)
    user = models.ForeignKey(User, related_name='user_orders',
on_delete=models.DO_NOTHING,verbose_name="用户ID")
    class Meta:
        db_table="ly_order"
        verbose_name= "订单记录"
        verbose_name_plural= "订单记录"


class OrderDetail(models.Model):
    """订单详情"""
    order = models.ForeignKey("Order", related_name='order_course',
on_delete=models.CASCADE, verbose_name="订单ID")
    course = models.ForeignKey(Course, related_name='course_order',
on_delete=models.CASCADE, verbose_name="课程ID")

    class Meta:
        db_table="ly_order_detail"
        verbose_name= "订单详情"
        verbose_name_plural= "订单详情"
```

后端实现生成订单的api接口

```
from django_redis import get_redis_connection
from rest_framework import status
from rest_framework.views import APIView

from decimal import Decimal
from .models import Order,OrderDetail
from datetime import datetime
from rest_framework.response import Response
import random
class OrderAPIView(APIView):
    def post(self,request):
        # 获取用户ID
        try:
            user_id = request.user.id
        except:
            return Response({"message":"用户不存在! "})

        # 自己生成一个订单号,# 结合时间戳和当前用户ID来生成，才能保证整站唯一
        order_number = datetime.now().strftime("%Y%m%d%H%M%S") + "%07d" % int(user_id)
+ "%04d" % random.randint(0,9999)
```

```
        # 从redis中获取商品信息[先获取勾选集,然后根据勾选集,到购物车中查询对应的商品价格]
        redis = get_redis_connection("cart")
        course_id_list = redis.smembers("cart_select_%s" % user_id )

        # 计算总价格
        total_price = 0
        cart_info = redis.hgetall("cart_%s" % user_id ) # 返回哈希数据中的键值对
        for course_id,course_price in cart_info.items():
            if course_id in course_id_list:
                total_price+= Decimal(course_price.decode())

        # 创建订单数据
        order = Order.objects.create(
            user_id=user_id,
            order_number=order_number,
            order_status=0,    # 订单状态默认为未支付
            order_desc="路飞学成课程购买", # # 订单描述信息
            total_price=total_price
        )

        # 返回响应信息给客户端
        if order:
            # 删除redis中已经生成订单的商品信息
            for course_id in course_id_list:
                # 记录订单相关的课程信息到订单详情

                OrderDetail.objects.create(
                    course_id = course_id,
                    order_id = order.id
                )

                redis.hdel("cart_%s" % user_id, course_id.decode() )
                redis.srem("cart_select_%s" % user_id, course_id.decode() )

            return Response({"message": "ok"},status=status.HTTP_200_OK)
        else:
            return  Response({"message": "生成订单失
败! "},status=status.HTTP_500_INTERNAL_SERVER_ERROR)
```

前端请求生成订单

```
<template>
  <div class="cart">
    <Header/>
    <div class="cart-info">
        <h3 class="cart-top">我的购物车 <span>共1门课程</span></h3>
        <div class="cart-title">
           <el-row>
             <el-col :span="2"> </el-col>
             <el-col :span="10">课程</el-col>
             <el-col :span="4">有效期</el-col>
             <el-col :span="4">单价</el-col>
```

```
              <el-col :span="4">操作</el-col>
            </el-row>
          </div>
          <CartItem v-for="item,course_key in course_list" @change_select="total_price"
@delete_course="del_course" :course_key="course_key" :course="item"/>
          <div class="calc">
              <el-row>
                <el-col :span="2"> </el-col>
                <el-col :span="3">
                    <el-checkbox label="全选" name="type"></el-checkbox></el-col>
                <el-col :span="2" class="del"><i class="el-icon-delete"></i>删除</el-col>
                <el-col :span="12" class="count">总计: ¥{{total}}</el-col>
                <el-col :span="3" class="cart-calc"><span @click="create_order">去结算
</span></el-col>
              </el-row>
          </div>
      </div>
      <Footer/>
    </div>
</template>

<script>
  import Header from "./common/Header"
  import Footer from "./common/Footer"
  import CartItem from "./common/CartItem"
  export default {
    name:"Cart",
    data(){
      return {
        token: localStorage.token || sessionStorage.token,
        id: localStorage.id || sessionStorage.id,
        course_list:[],
        total:0,
      }
    },
    components:{
      Header,
      Footer,
      CartItem,
    },
    created(){
      // 判断用户是否已经登陆了。
      if( !this.token || !this.id ){
        this.$router.push("/login");
      }
      let _this = this;
      // 发起请求获取购物车中的商品信息
      _this.$axios.get("http://127.0.0.1:8000/cart/",{
          headers: {
              'Authorization': 'JWT ' + _this.token
          },
          responseType: 'json',
          withCredentials: true
```

```
          }).then(response=>{
            _this.course_list = response.data;
            this.total_price()
          })
      },
      methods: {
        del_course(course_key) {
          this.course_list.splice(course_key, 1);
          // 重新计算总价格
          this.total_price();
        },
        total_price(msg){
          // 计算总价格
          let cl = this.course_list;
          let total = 0;
          for(let i = 0;i<cl.length;i++){
            if(cl[i].selected){
              total+=parseFloat(cl[i].price);
            }
          }
          total = total.toFixed(2);
          this.total = total;
        },
        create_order(){
          // 生成订单
          this.$axios.post("http://127.0.0.1:8000/buy/orders",{},{
            headers: {
              // 附带已经登录用户的jwt token 提供给后端,一定不能疏忽这个空格
              'Authorization': 'JWT ' + this.token
            },
            responseType: "json",
          }).then(response=>{
            // 跳转到结算页面
            this.$router.push("/order")

          }).catch(error=>{
            // 生成订单失败
          })

        }
      }
    }
  </script>
```

调整结算页中的订单课程信息

目前显示的信息是从redis购物车中提取的，是我们没有购买的课程信息。

所以要从数据库Order模型中提取.

```
```

发起支付

接入支付宝

支付宝开发平台登录

https://open.alipay.com/platform/home.htm

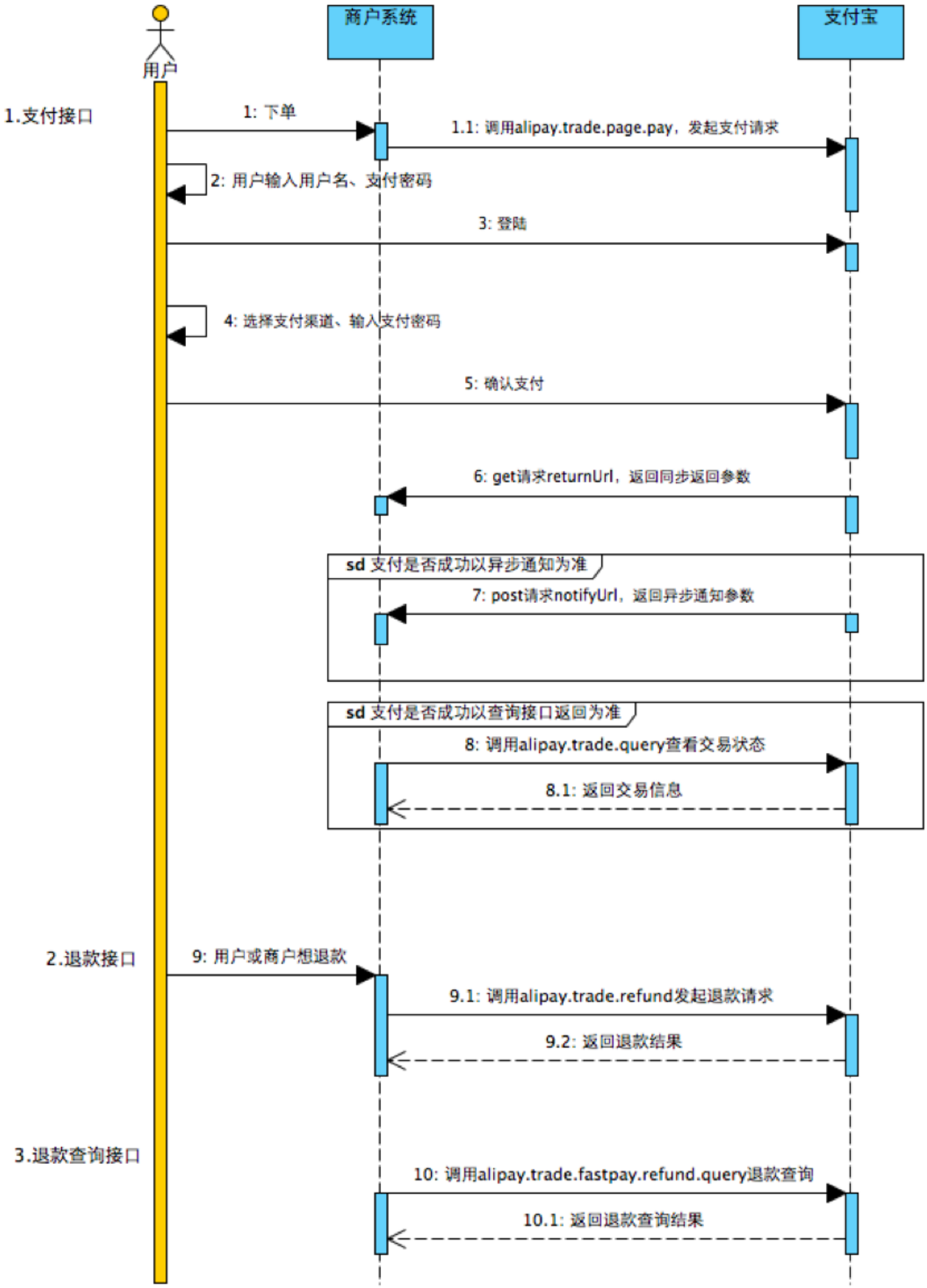
沙箱环境

- 是支付宝提供给开发者的模拟支付的环境
- 跟真实环境是分开的
- **沙箱应用**：https://docs.open.alipay.com/200/105311
- **沙箱账号**：https://openhome.alipay.com/platform/appDaily.htm?tab=account

支付宝开发者文档

- **文档主页**：https://openhome.alipay.com/developmentDocument.htm
- **产品介绍**：https://docs.open.alipay.com/270
- **快速接入**：https://docs.open.alipay.com/270/105899/
- SDK：https://docs.open.alipay.com/270/106291/
  - **python对接支付宝SDK**：https://github.com/fzlee/alipay/blob/master/README.zh-hans.md
  - **python对接支付宝SDK安装**：`pip install python-alipay-sdk --upgrade`
- **API列表**：https://docs.open.alipay.com/270/105900/

电脑网站支付流程


商户系统
支付宝
用户
1.支付接口
1: 下单
1.1: 调用alipay.trade.page.pay，发起支付请求
2: 用户输入用户名、支付密码
3: 登陆
4: 选择支付渠道、输入支付密码
5: 确认支付
6: get请求returnUrl，返回同步返回参数
sd 支付是否成功以异步通知为准
7: post请求notifyUrl，返回异步通知参数
sd 支付是否成功以查询接口返回为准
8: 调用alipay.trade.query查看交易状态
8.1: 返回交易信息
2.退款接口
9: 用户或商户想退款
9.1: 调用alipay.trade.refund发起退款请求
9.2: 返回退款结果
3.退款查询接口
10: 调用alipay.trade.fastpay.refund.query退款查询
10.1: 返回退款查询结果

接入步骤

1. 创建应用
2. 配置密钥
3. 搭建和配置开发环境
4. 接口调用

配置秘钥

1. 生成应用的私钥和公钥

2. 保存应用私钥文件

在payment应用中新建keys目录，用来保存秘钥文件。

将应用私钥文件app_private_key.pem复制到payment/keys目录下。

3. 查看公钥

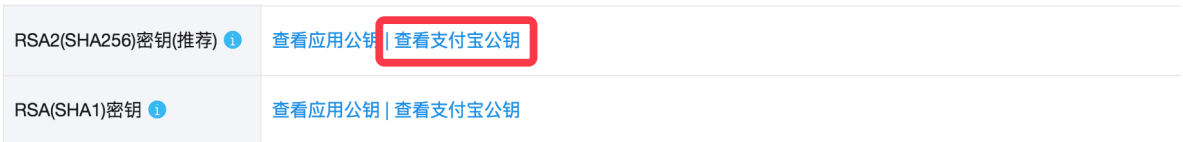
```
cat app_publict_key.pem
```

将公钥内容复制给支付宝

4. 保存支付宝公钥

在payment/keys目录下新建alipay_public_key.pem文件，用于保存支付宝的公钥文件。

将支付宝的公钥内容复制到alipay_public_key.pem文件中



注意，还需要在公钥文件中补充开始与结束标志

```
-----BEGIN PUBLIC KEY-----
此处是公钥内容
-----END PUBLIC KEY-----
```

后端实现发起支付接口

```
class PaymentAPIView(APIView):
    """支付宝"""
    permission_classes = (IsAuthenticated,)
```

```
    def get(self, request, order_id):
        """获取支付链接"""
        # 判断订单信息是否正确
        try:
            order = Order.objects.get(order_id=order_id, user=request.user,
                                      order_status=0,)
        except Order.DoesNotExist:
            return Response({'message': '订单信息有误'},
status=status.HTTP_400_BAD_REQUEST)

        # 构造支付宝支付链接地址
        alipay = AliPay(
            appid=settings.ALIPAY_APPID,
            app_notify_url=None,  # 默认回调url

app_private_key_path=os.path.join(os.path.dirname(os.path.abspath(__file__)),
"keys/app_private_key.pem"),

alipay_public_key_path=os.path.join(os.path.dirname(os.path.abspath(__file__)),
"keys/alipay_public_key.pem"),  # 支付宝的公钥，验证支付宝回传消息使用，不是你自己的公钥,
            sign_type="RSA2",  # RSA 或者 RSA2
            debug=settings.ALIPAY_DEBUG
        )

        order_string = alipay.api_alipay_trade_page_pay(
            out_trade_no=order.id,
            total_amount=str(order.total_price),
            subject=order.order_desc,
            return_url="http://127.0.0.1:8080/pay_success",
        )
        alipay_url = settings.ALIPAY_URL + "?" + order_string
        return Response({'alipay_url': alipay_url}, status=status.HTTP_201_CREATED)
```

在配置文件中编辑支付宝的配置信息[实际的值根据自己的账号而定]

```
# 支付宝
ALIPAY_APPID = "2016091600523592"
ALIPAY_URL = "https://openapi.alipaydev.com/gateway.do"
ALIPAY_DEBUG = True
```

前端点击"支付宝支付",请求后端的发起支付api

```
```

后端实现保存支付

```
class PaymentResultAPIView(APIView):
    """
    支付结果
    """
    def put(self, request):
        data = request.query_params.dict()
        signature = data.pop("sign")

        alipay = AliPay(
            appid=settings.ALIPAY_APPID,
            app_notify_url=None,  # 默认回调url

 app_private_key_path=os.path.join(os.path.dirname(os.path.abspath(__file__)),
"keys/app_private_key.pem"),

 alipay_public_key_path=os.path.join(os.path.dirname(os.path.abspath(__file__)),
                                                    "keys/alipay_public_key.pem"),  # 支付宝
的公钥，验证支付宝回传消息使用，不是你自己的公钥，
            sign_type="RSA2",  # RSA 或者 RSA2
            debug=settings.ALIPAY_DEBUG
        )

        success = alipay.verify(data, signature)
        if success:
            # 订单编号
            order_id = data.get('out_trade_no')
       Order.objects.filter(order_id=order_id).update(status=1)
            return Response({'message': "支付成功"})
        else:
            return Response({'message': '参数错误'}, status=status.HTTP_400_BAD_REQUEST)
```

支付成功的模板

```
<template>
  <div class="success">
    <Header :current_page="current_page"/>
    <div class="main">
        <div class="title">
          <img src="../../static/images/right.svg" alt="">
          <div class="success-tips">
              <p class="tips1">您已成功购买 1 门课程！</p>
              <p class="tips2">你还可以加入QQ群 <span>747556033</span> 学习交流</p>
          </div>
        </div>
        <div class="order-info">
            <p class="info1"><b>付款时间：</b><span>2019/04/02 10:27</span></p>
            <p class="info2"><b>付款金额：</b><span >0</span></p>
```

```
            <p class="info3"><b>课程信息：</b><span><span>《Pycharm使用秘籍》</span></span>
</p>
        </div>
        <div class="wechat-code">
          <img src="../../static/images/server.cf99f78.png" alt="" class="er">
          <p><img src="../../static/images/tan.svg" alt="">重要！微信扫码关注获得学习通知
&amp;课程更新提醒！否则将严重影响学习进度和课程体验！</p>
        </div>
        <div class="study">
          <span>立即学习</span>
        </div>
    </div>
    <Footer/>
  </div>
</template>

<script>
  import Header from "./common/Header"
  import Footer from "./common/Footer"
  export default{
    name:"Success",
    data(){
      return {
        current_page:0,
      };
    },
    components:{
      Header,
      Footer,
    }
  }
</script>

<style scoped>
.success{
  padding-top: 80px;
}
.main{
    height: 100%;
    padding-top: 25px;
    padding-bottom: 25px;
    margin: 0 auto;
    width: 1200px;
    background: #fff;
}
.main .title{
    display: flex;
    -ms-flex-align: center;
    align-items: center;
    padding: 25px 40px;
    border-bottom: 1px solid #f2f2f2;
}
.main .title .success-tips{
```

```
    box-sizing: border-box;
}
.title img{
    vertical-align: middle;
    width: 60px;
    height: 60px;
    margin-right: 40px;
}
.title .success-tips{
    box-sizing: border-box;
}
.title .tips1{
    font-size: 22px;
    color: #000;
}
.title .tips2{
    font-size: 16px;
    color: #4a4a4a;
    letter-spacing: 0;
    text-align: center;
    margin-top: 10px;
}
.title .tips2 span{
    color: #ec6730;
}
.order-info{
    padding: 25px 48px;
    padding-bottom: 15px;
    border-bottom: 1px solid #f2f2f2;
}
.order-info p{
  font-family: PingFangSC-Regular;
    display: -ms-flexbox;
    display: flex;
    margin-bottom: 10px;
    font-size: 16px;
}
.order-info p b{
  font-weight: 400;
  color: #9d9d9d;
  white-space: nowrap;
}
.wechat-code{
    display: flex;
    -ms-flex-align: center;
    align-items: center;
    padding: 25px 40px;
    border-bottom: 1px solid #f2f2f2;
}
.wechat-code>img{
    width: 100px;
    height: 100px;
    margin-right: 15px;
```

```
}
.wechat-code p{
    font-family: PingFangSC-Regular;
    font-size: 14px;
    color: #d0021b;
    display: -ms-flexbox;
    display: flex;
    -ms-flex-align: center;
    align-items: center;
}
.wechat-code p>img{
    width: 16px;
    height: 16px;
    margin-right: 10px;
}
.study{
      padding: 25px 40px;
}
.study span{
  display: block;
  width: 140px;
  height: 42px;
  text-align: center;
  line-height: 42px;
  cursor: pointer;
  background: #ffc210;
  border-radius: 6px;
  font-family: PingFangSC-Regular;
  font-size: 16px;
  color: #fff;
}
</style>
```