优医问诊day06

问诊室

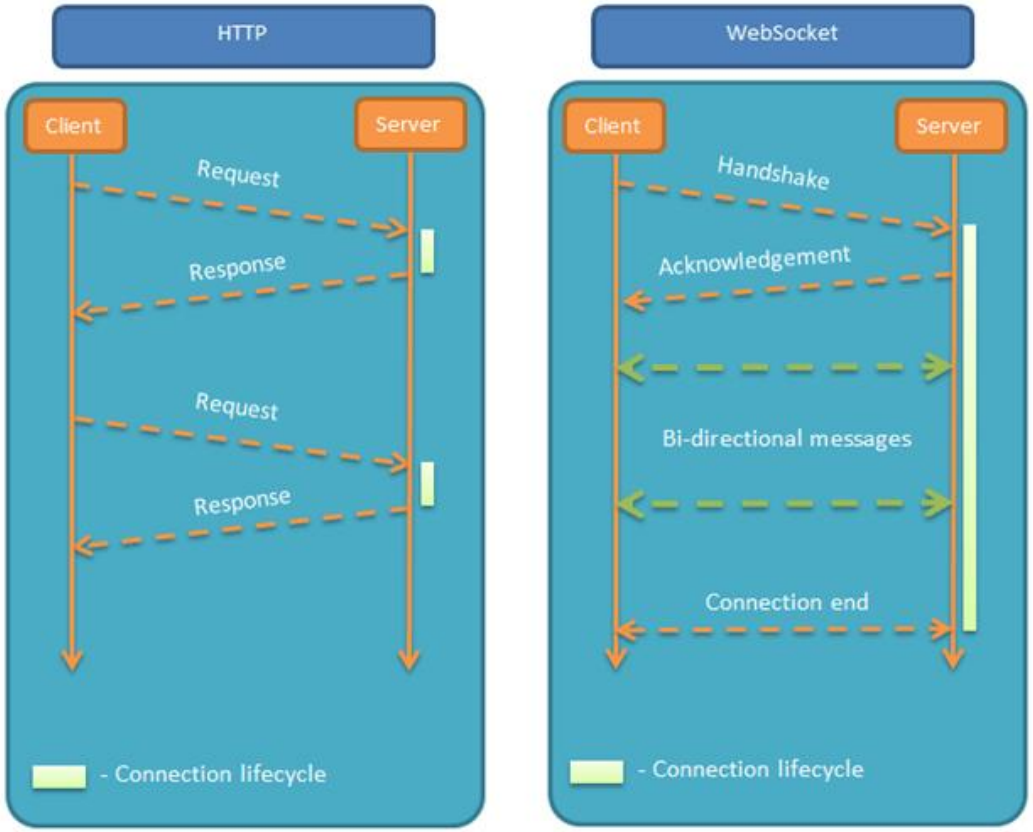
黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

问诊室–websocket介绍

问诊室-websocket介绍

- 什么是 websocket？
  - 网络通信协议，和 http 协议一样。
- 为什么需要 websocket？
  - 因为 http 协议通信只能由客户端发起，不能双向通信。




```
const ws = new WebSocket(
  "wss://javascript.info/article/websocket/demo/hello"
);

ws.onopen = function (evt) {
  console.log("Connection open ...");
  ws.send("Hello WebSockets!");
};
ws.onmessage = function (evt) {
  console.log("Received Message: " + evt.data);
  ws.close();
};
ws.onclose = function (evt) {
  console.log("Connection closed.");
};
```

项目中使用 socket.io-client 来实现客户端功能，它是基于 websocket 的 JS 库

问诊室–socket.io–client 使用

问诊室-socket.io-client 使用

- socket.io 是什么？
  - 基于 websocket 的即时通讯的解决方案，提供后端即时通信服务，前端连接后端 JS 库。
- socket.io 怎么用？
  - 服务端提供服务（忽略）
  - 客户端：怎么建立连接？发消息？收消息？断开连接？

官方demo：

git clone https://github.com/socketio/chat-example.git

```
import io from 'socket.io-client'

const socket = io('http://localhost:3000/')

socket.on('connect', () => {
  console.log('连接成功')
  socket.emit('chat message', '你好')
})
socket.on('chat message', (msg) => {
  console.log(msg)
  socket.close()
})
socket.on('disconnect', () => {
  console.log('断开连接')
})
```

问诊室–建立连接

问诊室-建立连接

初始化Socket → 携带token → 参数订单ID → 绑定常规事件 → 断开连接

```
export const baseURL = 'https://consult-api.itheima.net/'
const instance = axios.create({
  // 1. 基础地址，超时时间
  baseURL,
  timeout: 10000
})
```

```
let socket: Socket
onMounted(() => {
  // 建立连接
  socket = io(baseURL, {
    auth: {
      token: `Bearer ${store.user?.token}`
    },
    query: {
      orderId: route.query.orderId
    }
  })
  socket.on('connect', () => {
    console.log('连接成功')
  })
  socket.on('disconnect', () => {
    console.log('断开连接')
  })
  socket.on('error', (err) => {
    console.log('发生错误', err)
  })
})
```

```
onUnmounted(() => {
  // 关闭连接
  socket.close()
})
```

聊天-发送对话信息

POST https://consult-api.itheima.net/sendChatMsg • 开发中 ▶ 调试

修改时间: 5 个月前

本接口时socketio的方式请求，不能直接是用当前接口进行mock

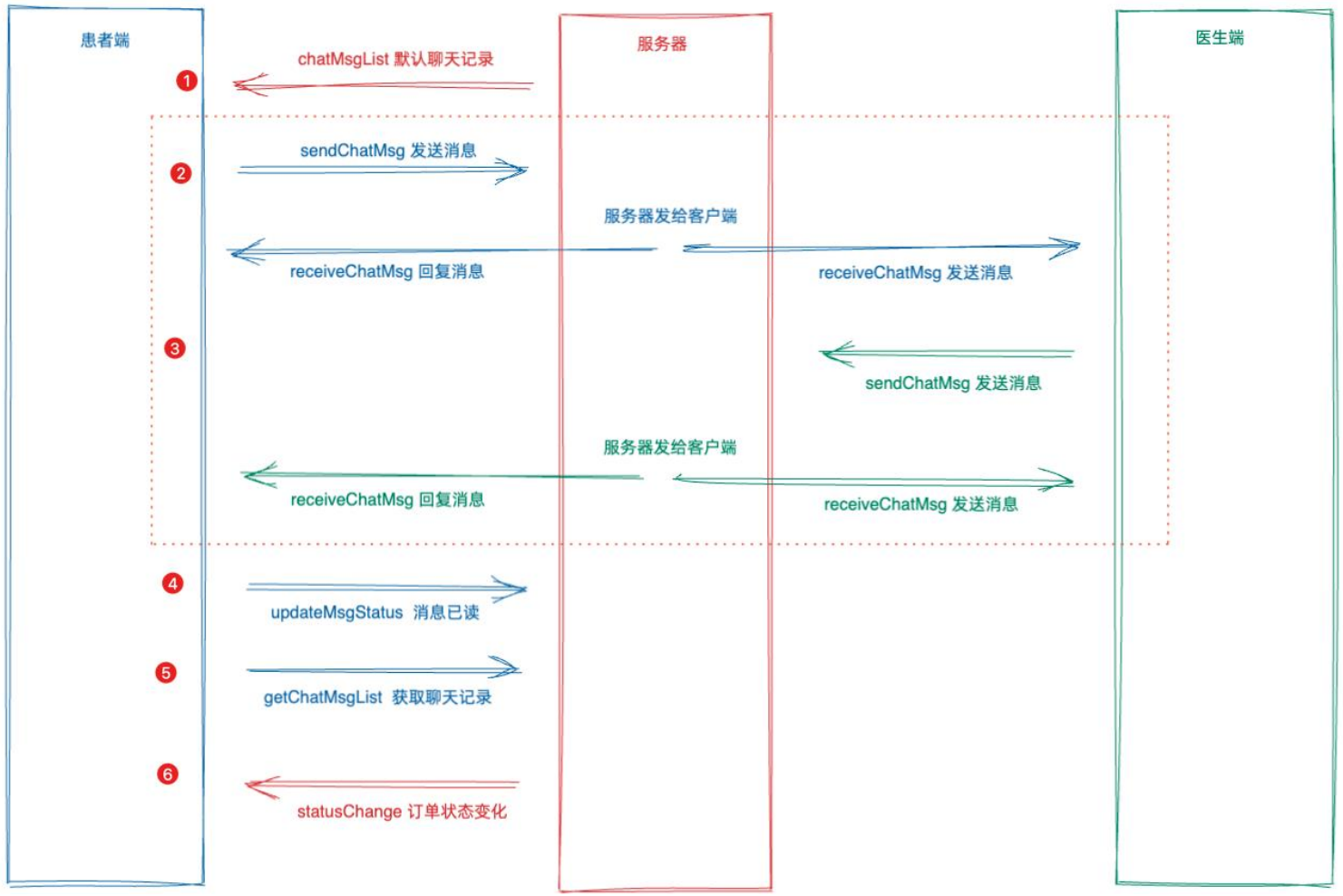
患者和医生创建聊天室 chatType:默认是1，可以不传, (暂时不考虑)患者创建客服聊天，chatType是2

1、客服端在打开聊天窗口时，传token，作为登录认证，传orderid作为参数（作为房间信息，区分不同的聊天室roomid）

```
var socket = io.connect(" http://127.0.0.1:7001 ", {
auth: {
token: "123",
},
query: {
orderId: 1,
},
});
```

04 问诊室–通讯规则与默认消息

问诊室-通讯规则与默认消息



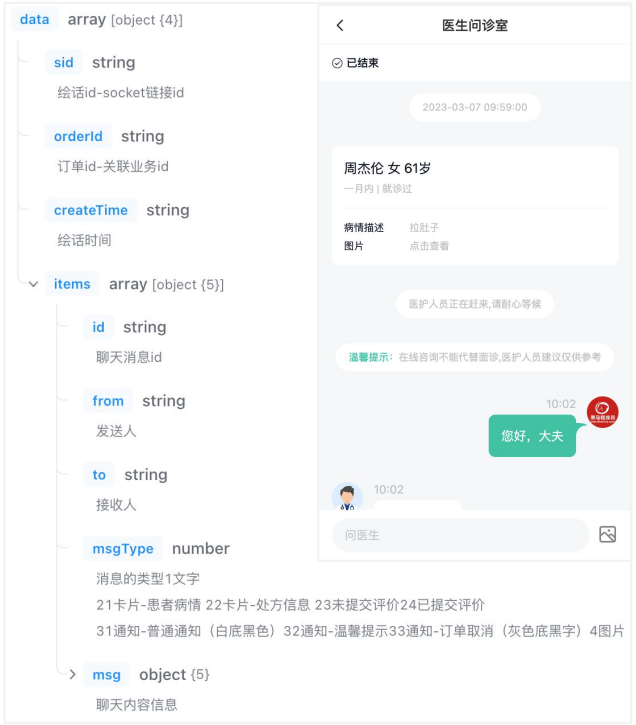

```
// 接收默认消息（聊天记录）
socket.on('chatMsgList', (res) => {
  console.log(res)
})
```

05 问诊室–默认消息–处理数据

问诊室-默认消息-处理数据

```
// 接收默认消息（聊天记录）
socket.on('chatMsgList', ({ data }) => {
  console.log(data)
})
```

分析数据结构 → 添加数据类型 → 转换消息列表




```
export enum MsgType {
  /** 文字聊天 */
  MsgText = 1,
  /** 消息聊天 */
  MsgImage = 4,
  /** 患者信息 */
  CardPat = 21,
  /** 处方信息 */
  CardPre = 22,
  /** 未评价信息 */
  CardEvaForm = 23,
  /** 已评价信息 */
  CardEva = 24,
  /** 通用通知 */
  Notify = 31,
  /** 温馨提示 */
  NotifyTip = 32,
  /** 取消提示 */
  NotifyCancel = 33
}

// 处方状态
export enum PrescriptionStatus {
  /** 未付款 */
  NotPayment = 1,
  /** 已付款 */
  Payment = 2,
  /** 已失效 */
  Invalid = 3
}
```

```
export type Message = {
  /** 消息ID */
  id: string
  /** 消息类型 */
  msgType: MsgType
  /** 发信人 */
  from?: string
  /** 发信人ID */
  fromAvatar?: string
  /** 收信人 */
  to?: string
  /** 收信人头像 */
  toAvatar?: string
  /** 创建时间 */
  createTime: string
  /** 消息主体 */
  msg: {
    /** 文本内容 */
    content?: string
    /** 图片对象 */
    picture?: Image
    /** 问诊记录，患者信息 */
    consultRecord?: Consult & {
      patientInfo: Patient
    }
    /** 处方信息 */
    prescription?: Prescription
    /** 评价信息 */
    evaluateDoc?: EvaluateDoc
  }
}
```

```
// 消息分组列表
export type TimeMessages = {
  /** 分组消息最早时间 */
  createTime: string
  /** 消息数组 */
  items: Message[]
  /** 订单ID */
  orderId: string
  /** 会话ID */
  sid: string
}
```

```
const list = ref<Message[]>([])
// 接收默认消息（聊天记录）
socket.on('chatMsgList', ({ data }: { data: TimeMessages[] }) => {
  const arr: Message[] = []
  data.forEach((item) => {
    arr.push({
      msgType: MsgType.Notify,
      msg: { content: item.createTime },
      createTime: item.createTime,
      id: item.createTime
    })
    arr.push(...item.items)
    // 追加消息列表
    list.value.unshift(...arr)
  })
})
```

10.00-12:00　12.00-14:00

06

问诊室–默认消息–渲染消息

问诊室-默认消息-渲染消息

传递数据给组件 → 展示患者卡片 → 预览图片 → 通用消息和提示

```vue
<!-- 消息 -->
<room-message
  v-for="item in list"
  :key="item.id"
  :item="item"
></room-message>
```

```ts
defineProps<{ item: Message }>()

const getIllnessTimeText = (time: IllnessTime) =>
  timeOptions.find((item) => item.value === time)?.label

const getConsultFlagText = (flag: 0 | 1) =>
  flagOptions.find((item) => item.value === flag)?.label
```

```ts
// 预览图片
const onPreviewImage = (images?: Image[]) => {
  if (images && images.length) showImagePreview(images.map((img) => img.url))
  else showToast('暂无图片')
}
```

```vue
<!-- 患者卡片 -->
<div class="msg msg-illness" v-if="item.msgType === MsgType.CardPat">
  <div class="patient van-hairline--bottom">
    <p>
      {{ item.msg.consultRecord?.patientInfo.name }}
      {{ item.msg.consultRecord?.patientInfo.genderValue }}
      {{ item.msg.consultRecord?.patientInfo.age }}岁
    </p>
    <p v-if="item.msg.consultRecord">
      {{ getIllnessTimeText(item.msg.consultRecord.illnessTime) }} |
      {{ getConsultFlagText(item.msg.consultRecord.consultFlag) }}
    </p>
  </div>
  <van-row>
    <van-col span="6">病情描述</van-col>
    <van-col span="18">{{ item.msg.consultRecord?.illnessDesc }}</van-col>
    <van-col span="6">图片</van-col>
    <van-col
      span="18"
      @click="onPreviewImage(item.msg.consultRecord?.pictures)"
    >
      点击查看
    </van-col>
  </van-row>
</div>
```

```vue
<!-- 通知-通用 -->
<div class="msg msg-tip" v-if="item.msgType === MsgType.Notify">
  <div class="content">
    <span>{{ item.msg.content }}</span>
  </div>
</div>
<!-- 通知-温馨提示 -->
<div class="msg msg-tip" v-if="item.msgType === MsgType.NotifyTip">
  <div class="content">
    <span class="green">温馨提示：</span>
    <span>{{ item.msg.content }}</span>
  </div>
</div>
```

```ts
export const timeOptions = [
  { label: '一周内', value: IllnessTime.Week },
  { label: '一月内', value: IllnessTime.Month },
  { label: '半年内', value: IllnessTime.HalfYear },
  { label: '大于半年', value: IllnessTime.More }
]
export const flagOptions = [
  { label: '就诊过', value: 0 },
  { label: '没就诊过', value: 1 }
]
```

07 问诊室–接诊状态–订单数据

问诊室-接诊状态-订单数据

订单状态枚举 → 订单详情类型 → 订单详情API → 初始化&状态变更

```ts
export enum OrderType {
  // 问诊订单
  /** 待支付 */
  ConsultPay = 1,
  /** 待接诊 */
  ConsultWait = 2,
  /** 问诊中 */
  ConsultChat = 3,
  /** 问诊完成 */
  ConsultComplete = 4,
  /** 取消问诊 */
  ConsultCancel = 5,
  // 药品订单
  /** 待支付 */
  MedicinePay = 10,
  /** 待发货 */
  MedicineSend = 11,
  /** 待收货 */
  MedicineTake = 12,
  /** 已完成 */
  MedicineComplete = 13,
  /** 取消订单 */
  MedicineCancel = 14
}
```

```ts
// 问诊订单单项信息
export type ConsultOrderItem = Consult & {
  /** 创建时间 */
  createTime: string
  /** 医生信息 */
  docInfo?: Doctor
  /** 患者信息 */
  patientInfo: Patient
  /** 订单编号 */
  orderNo: string
  /** 订单状态 */
  status: OrderType
  /** 状态文字 */
  statusValue: string
  /** 类型问诊文字 */
  typeValue: string
  /** 倒计时时间 */
  countdown: number
  /** 处方ID */
  prescriptionId?: string
  /** 评价ID */
  evaluateId: number
  /** 应付款 */
  payment: number
  /** 优惠券抵扣 */
  couponDeduction: number
  /** 积分抵扣 */
  pointDeduction: number
  /** 实付款 */
  actualPayment: number
}
```

```ts
export const getConsultOrderDetail = (orderId: string) =>
  request<ConsultOrderItem>('/patient/consult/order/detail', 'GET', { orderId })
```

```ts
// 获取问诊详情
const consult = ref<ConsultOrderItem>()
const loadConsult = async () => {
  const res = await getConsultOrderDetail(route.query.orderId as string)
  consult.value = res.data
}
```

```ts
onMounted(() => {
  loadConsult()
  // 建立连接
  socket = io(baseURL, {…
  })
  socket.on('connect', () => {…
  })
  socket.on('disconnect', () => {…
  })
  socket.on('error', (err) => {…
  })
  // 接收默认消息（聊天记录）
  socket.on('chatMsgList', ({ data }: { data: TimeMessages[] }) => {…
  })
  // 诊室状态改变
  socket.on('statusChange', () => loadConsult())
})
```

08 问诊室–接诊状态–控制组件

问诊室-接诊状态-控制组件

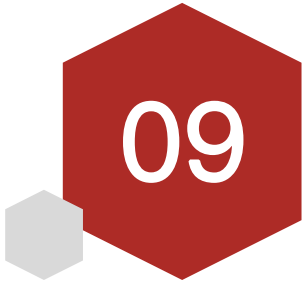
控制状态栏 → 控制操作栏

```vue
<script setup lang="ts">
import { OrderType } from '@/enums'

defineProps<{
  status?: OrderType
  countdown?: number
}>()
</script>

<template>
  <div class="room-status">
    <div v-if="status === OrderType.ConsultWait" class="wait">
      已通知医生尽快接诊，24小时内医生未回复将自动退款
    </div>
    <div v-if="status === OrderType.ConsultChat" class="chat">
      <span>咨询中</span>
      <span>
        剩余时间:
        <van-count-down v-if="countdown" :time="countdown * 1000" />
      </span>
    </div>
    <div
      v-if="
        status === OrderType.ConsultComplete ||
        status === OrderType.ConsultCancel
      "
      class="end"
    >
      <van-icon name="passed" /> 已结束
    </div>
  </div>
</template>
```

```vue
<!-- 状态栏 -->
<room-status
  :status="consult?.status"
  :countdown="consult?.countdown"
></room-status>
```

```vue
<script setup lang="ts">
defineProps<{
  disabled: boolean
}>()
</script>

<template>
  <div class="room-action">
    <van-field
      type="text"
      class="input"
      :border="false"
      placeholder="问医生"
      autocomplete="off"
      :disabled="disabled"
    ></van-field>
    <van-uploader :preview-image="false" :disabled="disabled">
      <cp-icon name="consult-img" />
    </van-uploader>
  </div>
</template>
```

```vue
<!-- 操作栏 -->
<room-action
  :disabled="consult?.status !== OrderType.ConsultChat"
></room-action>
```



问诊室–文字聊天–发送文字

问诊室-文字聊天-发送文字

操作栏传递文字 → 诊室接收文字 → socket发送文字

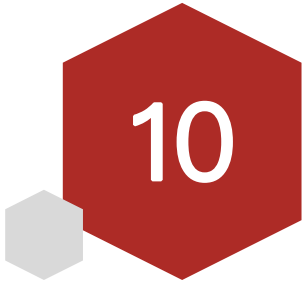
```vue
const emit = defineEmits<{
  (e: 'send-text', text: string): void
}>()

const text = ref('')

const sendText = () => {
  emit('send-text', text.value)
  text.value = ''
}
```

```vue
<van-field
  type="text"
  class="input"
  :border="false"
  placeholder="问医生"
  autocomplete="off"
  :disabled="disabled"
  v-model="text"
  @keyup.enter="sendText"
></van-field>
```

```vue
<room-action
  @send-text="onSendText"
  :disabled="consult?.status !== OrderType.ConsultChat"
></room-action>
```

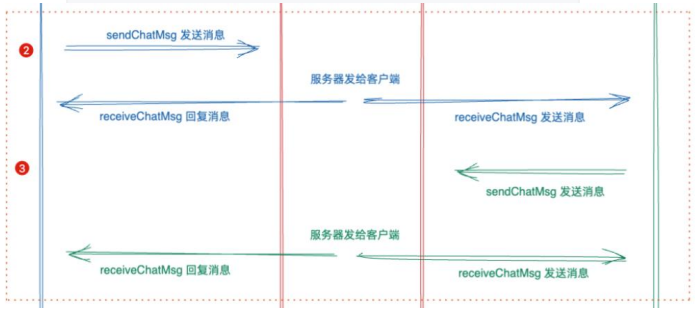
```vue
// 发送文字消息
const onSendText = (text: string) => {
  console.log(text)
}
```

```vue
// 发送文字信息
const onSendText = (text: string) => {
  socket.emit('sendChatMsg', {
    from: store.user?.id,
    to: consult.value?.docInfo?.id,
    msgType: MsgType.MsgText,
    msg: {
      content: text
    }
  })
}
```



问诊室–文字聊天–渲染消息

问诊室-文字聊天-渲染消息



接收消息 → 展示文字消息 → 处理时间

```
// 接收聊天消息
socket.on('receiveChatMsg', async (event) => {
  list.value.push(event)
  await nextTick()
  window.scrollTo(0, document.body.scrollHeight)
})
```

```
import dayjs from 'dayjs'

const formatTime = (time: string) => dayjs(time).format('HH:mm')
```

```
<div
  class="msg msg-to"
  v-if="item.msgType === MsgType.MsgText && store.user?.id === item.from"
>
  <div class="content">
    <div class="time">{{ item.createTime }}</div>
    <div class="pao">{{ item.msg.content }}</div>
  </div>
  <van-image :src="item.fromAvatar" />
</div>
<!-- 发送图片 -->
<!-- <div class="msg msg-to">…
<!-- 接收文字 -->
<div
  class="msg msg-from"
  v-if="item.msgType === MsgType.MsgText && store.user?.id !== item.from"
>
  <van-image :src="item.fromAvatar" />
  <div class="content">
    <div class="time">{{ item.createTime }}</div>
    <div class="pao">{{ item.msg.content }}</div>
  </div>
</div>
```

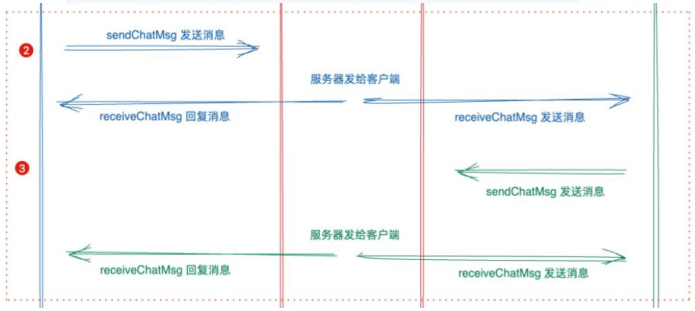
问诊室-文字聊天-渲染消息

接收消息 → 展示文字消息 → 处理时间

```
// 接收聊天消息
socket.on('receiveChatMsg', async (event) => {
  list.value.push(event)
  await nextTick()
  window.scrollTo(0, document.body.scrollHeight)
})
```

```
import dayjs from 'dayjs'

const formatTime = (time: string) => dayjs(time).format('HH:mm')
```

```
<div
  class="msg msg-to"
  v-if="item.msgType === MsgType.MsgText && store.user?.id === item.from"
>
  <div class="content">
    <div class="time">{{ formatTime(item.createTime) }}</div>
    <div class="pao">{{ item.msg.content }}</div>
  </div>
  <van-image :src="item.fromAvatar" />
</div>
<!-- 发送图片 -->
<!-- <div class="msg msg-to">…
<!-- 接收文字 -->
<div
  class="msg msg-from"
  v-if="item.msgType === MsgType.MsgText && store.user?.id !== item.from"
>
  <van-image :src="item.fromAvatar" />
  <div class="content">
    <div class="time">{{ formatTime(item.createTime) }}</div>
    <div class="pao">{{ item.msg.content }}</div>
  </div>
</div>
```



问诊室–图片聊天

问诊室-图片聊天

操作栏提交图片 → 诊室接收并发送 → 渲染图片消息

```
<van-uploader
  :preview-image="false"
  :disabled="disabled"
  :after-read="sendImage"
>
  <cp-icon name="consult-img" />
</van-uploader>
```

```
const emit = defineEmits<{
  (e: 'send-text', text: string): void
  (e: 'send-image', img: Image): void
}>()
```

```
const sendImage: UploaderAfterRead = async (data) => {
  if (Array.isArray(data)) return
  if (!data.file) return
  const t = showLoadingToast({ message: '正在上传', duration: 0 })
  const res = await uploadImage(data.file)
  t.close()
  emit('send-image', res.data)
}
```

```
<room-action
  @send-text="onSendText"
  @send-image="onSendImage"
  :disabled="consult?.status !== OrderType.ConsultChat"
></room-action>
```

```
// 发送图片消息
const onSendImage = (image: Image) => {
  socket.emit('sendChatMsg', {
    from: store.user?.id,
    to: consult.value?.docInfo?.id,
    msgType: MsgType.MsgImage,
    msg: { picture: image }
  })
}
```

```
<!-- 发送图片 -->
<div
  class="msg msg-to"
  v-if="item.msgType === MsgType.MsgImage && store.user?.id === item.from"
>
  <div class="content">
    <div class="time">{{ formatTime(item.createTime) }}</div>
    <van-image fit="contain" :src="item.msg.picture?.url" />
  </div>
  <van-image :src="item.fromAvatar" />
</div>
<!-- 接收文字 -->
<div …
</div>
<!-- 接收图片 -->
<div
  class="msg msg-from"
  v-if="item.msgType === MsgType.MsgImage && store.user?.id !== item.from"
>
  <van-image :src="item.fromAvatar" />
  <div class="content">
    <div class="time">{{ formatTime(item.createTime) }}</div>
    <van-image fit="contain" :src="item.msg.picture?.url" />
  </div>
</div>
```

12 问诊室–聊天记录

问诊室-聊天记录

默认看最新消息 → 实现下拉刷新 → 记录消息时间 → 通知服务器，loading & 提示

```
// 接收默认消息（聊天记录）
socket.on('chatMsgList', ({ data }: { data: TimeMessages[] }) => {
  const arr: Message[] = []
  data.forEach((item) => {…
  })
  // 第一次加载消息列表，看最新消息
  if (initialMsg.value) {
    nextTick(() => {
      window.scrollTo(0, document.body.scrollHeight)
      initialMsg.value = false
    })
  }
})
```

```
// 接收默认消息（聊天记录）
socket.on('chatMsgList', ({ data }: { data: TimeMessages[] }) => {
  const arr: Message[] = []
  data.forEach((item, i) => {
    // 记录下一次聊天记录的时间
    if (i === 0) time.value = item.createTime
    arr.push({
```

- 2023-04-01 14:00:00
- 2023-04-01 12:00:00
- 2023-04-01 10:00:00

```
<!-- 消息 -->
<van-pull-refresh v-model="loading" @refresh="onRefresh">
  <room-message
    v-for="item in list"
    :key="item.id"
    :item="item"
  ></room-message>
</van-pull-refresh>
```

```
// 聊天记录
const loading = ref(false)
const time = ref(dayjs().format('YYYY-MM-DD HH:mm:ss'))
const onRefresh = () => {
  // 触发加载
}
```

```
const onRefresh = () => {
  // 触发加载
  socket.emit('getChatMsgList', 20, time.value, route.query.orderId)
}
```

```
// 接收默认消息（聊天记录）
socket.on('chatMsgList', ({ data }: { data: TimeMessages[] }) => {
  const arr: Message[] = []
  data.forEach((item, i) => {…
  })
  loading.value = false
  if (!arr.length) return showToast('没有聊天记录了')
```

问诊室–消息已读

问诊室-消息已读

展示未读消息 → 已读所有消息 → 收一条读一条

```
// 获取未读消息
export const getUnreadMessageCount = () =>
  request<number>('patient/message/unRead/all')
```

```
const count = ref<number>()
onMounted(async () => {
  const res = await getUnreadMessageCount()
  count.value = res.data
})
```

```
<van-tabbar-item to="/notify" :badge="count || ''">
  消息通知
```

```
// 第一次加载消息列表，看最新消息
if (initialMsg.value) {
  // 已读
  socket.emit('updateMsgStatus', arr[arr.length - 1].id)
  nextTick(() => {
    window.scrollTo(0, document.body.scrollHeight)
    initialMsg.value = false
  })
}
```

```
// 接收聊天消息
socket.on('receiveChatMsg', async (event) => {
  // 已读
  socket.emit('updateMsgStatus', event.id)
  list.value.push(event)
  await nextTick()
  window.scrollTo(0, document.body.scrollHeight)
})
```

13 问诊室–查看处方

问诊室-查看处方

医生开处方&渲染 → 原始处方API → 预览原始处方

```vue
<div class="msg msg-recipe" v-if="item.msgType === MsgType.CardPre">
  <div class="content" v-if="item.msg.prescription">
    <div class="head van-hairline--bottom">
      <div class="head-tit">
        <h3>电子处方</h3>
        <p @click="onShowPrescription(item.msg.prescription?.id)">
          原始处方 <van-icon name="arrow"></van-icon>
        </p>
      </div>
      <p>
        {{ item.msg.prescription.name }}
        {{ item.msg.prescription.genderValue }}
        {{ item.msg.prescription.age }}岁
        {{ item.msg.prescription.diagnosis }}
      </p>
      <p>开方时间：{{ item.msg.prescription.createTime }}</p>
    </div>
    <div class="body">
      <div
        class="body-item"
        v-for="med in item.msg.prescription.medicines"
        :key="med.id"
      >
        <div class="durg">
          <p>{{ med.name }} {{ med.specs }}</p>
          <p>{{ med.usageDosag }}</p>
        </div>
        <div class="num">x{{ med.quantity }}</div>
      </div>
    </div>
    <div class="foot">…
    </div>
  </div>
</div>
```

```ts
// 查看处方
export const getPrescriptionPic = (id: string) =>
  request<{ url: string }>(`/patient/consult/prescription/${id}`)
```

```ts
// 电子处方
const onShowPrescription = async (id?: string) => {
  if (id) {
    const res = await getPrescriptionPic(id)
    showImagePreview([res.data.url])
  }
}
```

传智教育旗下高端IT教育品牌