首先脚手架搭建好项目，搭建的vue2的项目，将一些默认的样式和内容清除掉，只留下home页面

在home页面，这个日历组件中有输入框和日历内容

输入框现在展示的数据是当前的时间，这个时间由外面来控制，所以从父组件将当前时间传过去

HomeView.vue

```
<template>
  <div class="home">
    <DatePicker :value="now"></DatePicker>
  </div>
</template>

<script>
// @ is an alias to /src
import DatePicker from '@/components/DatePicker.vue'

export default {
  name: 'HomeView',
  data() {
    return {
      now: new Date()//获取当前的日期
    }
  },
  components: {
    DatePicker
  }
}
</script>
```

新建一个组件，用来表示日历组件，这个组件内，会有一个输入框，还有点击输入框就会展示的日历面板

component/DatePicker.vue

```
<template>
  <div>
    <!-- 这里不能使用v-model进行双向绑定，因为这个值是由父组件传递过来的，子组件不能去修改父组件的数据 -->
    <input type="text" :value="value">
    <div>content</div>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  props: {
    value: {
      type: Date,
      default: () => new Date()
    }
  }
}
```

```
</script>
```

可以看到在输入框中已经显出今天的日期时间了，但是我们只要年月日就可以了，所以我们还需要对这个值进行转换，通过计算属性去处理

我们去封装一个函数，用来获取当前时间的年月日，因为后面也有其他地方需要去获取日期的年月日，

src/util.js

```
// 放置工具方法
// 处理时间，返回年月日
const getYearMonthDay = (date) => {
  let year = date.getFullYear();
  let month = date.getMonth();
  let day = date.getDate();
  return {
    year,
    month,
    day,
  };
};
export {
    getYearMonthDay
}
```

component/DatePicker.vue,**注意month要加1**

```
  computed: {
    formatDate() {
      let { year, month, day } = utils.getYearMonthDay(this.value)
      return `${year}-${month+1}-${day}`
    }
  }
```

在上面的输入框的绑定也需要修改一下绑定值

```
 <input type="text" :value="formatDate">
```

输入框已经搞定了，接着当我们点击输入框的时候，下面的面板是不是才显示，当我们离开输入框的时候，下面的面板是不是就消失，所以我们去定义一个变量来决定这个面板是应该展示还是取消

component/DatePicker.vue

```
data() {
    return {
      isVisible: false//用来控制面板是否可见
    }
  },
```

```
<div v-if="isVisible">content</div>
```

处理面板取消和消失的效果

我们通过使用事件委托，把点击事件给到document，利用事件委托，那么在点击的时候就可以去做一下判断，如果说你在点击的是这个日历组件中的内容，那么面板不会消失，但是你点日历组件外面的内容，面板就会取消，定义两个方法去控制变量的值，也就是控制面板的显示和消失

```
methods: {
    focus() {
      this.isVisible = true
    },
    blur() {
      this.isVisible = false
    }
  }
```

在这里通过自定义指令去操作，可以去判断当前点击的是否日历组件中的元素，因为指令的生命周期的第一个参数就是当前元素的真实dom吧，我们就使用对应的方法去判断

DatePicker.vue

```
  <div v-click-outside>//给最外层div绑定自定义指令
    <!-- 这里不能使用v-model进行双向绑定，因为这个值是由父组件传递过来的，子组件不能去修改父组件的数据 -->
    <input type="text" :value="formatDate">
    <div v-if="isVisible">content</div>
  </div>
```

```
  directives:{
    clickOutside:{
      bind(el,bingding,vnode){//el当前绑定的dom元素  vnode（虚拟节点）-->context 上下文    //-->只调用一次，指令第一次绑定到元素时调用。
        // 把事件绑定给document上 看一下点击的是否是当前这个元素的内部
        let handler=(e)=>{// e 拿取到目标触发对象
        if(el.contains(e.target)){//判断当前点击的元素是否属于绑定元素中的内容
              console.log('包含');
              vnode.context.focus()//vnode虚拟节点中context表示上下文，可以把它理解为就是this,this中就会有这些方法，调用这些方法去控制下面内容的                显示和销毁
             }else{
              console.log("不包含");
              vnode.context.blur()
             }
        }
        // 将事件触发函数绑定到el上，因为当组件被销毁掉，绑定在document上的事件也要移除掉
        el.handler=handler
        document.addEventListener('click',handler)
      },
      unbind(el){//只调用一次，指令与元素解绑时调用
        // 进行移除事件
        document.removeEventListener('click',el.handler)
      }
    }
  },
```

现在的话，就可以先确定我点击的是不是这个组件内的内容，并且还可以去显示下面的内容，但是这样会有一个问题，就是我们点击第一次的时候，就已经是包含的状态了，显示出来了下面的内容，所以当我们再次点击里面的内容的，就不需要再次触发事件了吧，不管是显示还是销毁都只需要走一次就行了，所以我们还需要再来优化一下，再做一次判断就好了

```
        if (el.contains(e.target)) {//判断当前点击的元素是否属于绑定元素中的内容
            // 判断一下是否当前面板已经显示出来了
            if (!vnode.context.isVisible) {
              console.log('包含');
              vnode.context.focus()
            }
          } else {
            if (vnode.context.isVisible) {
              console.log("不包含");
              vnode.context.blur()
            }
          }
```

这样的话，我们就已经实现了第一步

那么接下来就是去实现这个面板的内容，也就是这个日历的内容样式要给她实现出来

这个日历面板分为三个部分（内容可以自己决定，或者网上搜索都可以），一个可以进行切换时间和展示时间的头部，还有一个展示日期内容的，还有一个展示今天的日期时间，一个粗略的样子大概是这样的

```
    <div class="pannel" v-if="isVisible">
      <div class="pannel-nav">
        <span>&lt;</span>
        <span>&lt;&lt;</span>
        <span>xxx年</span>
        <span>xxx月</span>
        <span>&gt;&gt;</span>
        <span>&gt;</span>
      </div>
      <div class="pannel-content">
        内容
      </div>
      <div class="pannel-footer">今天</div>
    </div>
```

有些点需要注意，如果在这个组件中还有一些其他的内容，我们这个面板显示的时候肯定是不能影响页面的布局,所以去写一下样式，

```
.pannel {
  position: absolute;
  background: #fff;
}
```

我们先把这个日期写出来，也就是日期内容先展示出来

首先先分析一下，{日期内容要怎么展示，日历几行几列，面板总共展示出来的日期会有 42个 ，我们只需要算出日期面板的第一天，再往后循环42天就好，就是当前面板要展示的所有日期，那么日期的最开始的时间怎么算出来呢，首先我们可以得到当前年月的第一天是星期几，算出它与面板的第一天相隔多久，进行相减，就可以得到面板的第一天是哪个日期，就是当前面板要展示的所有日期

```
    computed(){
        visibleDate(){//当前面板要展示的所有日期
          // 先获取当前时间
          // 先生成当前的日期 拿到当前年月，再去获取当前年月的第一天的日期对象
          // 获取当前是周几 再把天数往前移动几天
          // 循环42天
        },
    }
```

```
    visibleDate() {//当前面板要展示的所有日期
      // 先获取当前是周几
      let { year, month } = utils.getYearMonthDay(this.value)
      // 先生成当前的日期 拿到当前年月，再去获取当前年月的第一天
      // 封装了一个函数，用来生成日期对象
      let currentFirstDay = utils.getDate(year, month, 1)
      // 获取当前是周几 再把天数往前移动几天
      // 获取距离面板第一天有多少天
      let week = currentFirstDay.getDay() - 1
      // 拿取当前面板的第一天
      // 如果一个日期格式和一个毫秒相减，会变成一个时间毫秒戳
      let startDay = currentFirstDay - week * 60 * 60 * 1000 * 24
      // 循环42天
      let arr = []
      for (let i = 0; i < 42; i++) {//往后循环42天，将后面的42天的日期保存在arr数组中
        arr.push(new Date(startDay + i * 60 * 60 * 1000 * 24))//第一次循环是0，所以
拿到是最开始的日期，后面每一次循环就添加一天
      }
      return arr//将数组返回出去
    },
```

先将这个数组放入到元素中来看看

```
    <div class="pannel-content">
        <div>
          {{ visibleDate }}
        </div>
      </div>
```

可以看到的是日期会展示出来，但是发现会有点不一样，因为这个是东八区的时间，和我们相差8个小时，所以是16点，加上八个小时，正好是第二天的24点整，所以我们这个数据是正确的,但是我们要展示的只要每个月的多少号就行了，并且这个样式，也就是几行几列也需要展示清楚，所以还需要处理一下

```
<div class="pannel-content">
        <div class="days">
          <!-- 直接列出一个6*7的列表,有点像九九乘法表，一行就是一个div,每一行中放7个span标
签 -->
          <div v-for="i in 6" :key="i">
            <span v-for="j in 7" :key="j" class="cell">
              {{ visibleDate[(i - 1) * 7 + (j - 1)].getDate() }}
            </span>
          </div>
        </div>
      </div>
```

这样就可以实现，中间部分的效果展示

处理后面的样式

```
.pannel {
  width: 32 * 7px;
  position: absolute;
  background: #fff;
  box-shadow: 2px 2px 2px pink, -2px -2px 2px pink;
  .pannel-nav {
    display: flex;
    justify-content: space-around;
    height: 30px;
    span {
      cursor: pointer;
      user-select: none;
    }
  }
  .pannel-content .cell {
    display: inline-flex;
    justify-content: center;
    align-items: center;
    width: 32px;
    height: 32px;
  }
  .pannel-footer {
    height: 30px;
    text-align: center;
  }
}
```

面板内容的具体样式已经出来了，接下来开始处理样式，上个月和下个月与当月的样式是不一样的

所以需要判断是否是当月的，可以去控制class，给循环的每一个日期，添加一个动态的样式，根据返回值去判断是否是当前月的日期来显示对应的样式

```
            <span v-for="j in 7" :key="j" class="cell" :class="[
              // 判断是否是当月，不是当月就变成灰色
              { notCurrentMonth: !isCurrentMonth(visibleDate[(i - 1) * 7 + (j -
1)]) }
            ]">
              {{ visibleDate[(i - 1) * 7 + (j - 1)].getDate() }}
            </span>
```

```
isCurrentMonth(date) {//判断是否是当月
      // 现在知道的当前用户传入的值 this.value,拿取到传入进来的日期的年月
      let { year, month } = utils.getYearMonthDay(this.value)
      //拿取当前循环的日期的年月
      let { year: y, month: m } = utils.getYearMonthDay(date)
      // 和传入进来的年月去做比较，只要年和月是对的，那么就说明是在当前月的
      return y === year && month === m
    }
```

```
.pannel-content .cell {
  display: inline-flex;
  //...
  font-weight: bold;//是当前月的颜色加重显示
}
.notCurrentMonth{//不是当前月，变灰色
  color: gray;
}
```

除了判断是不是当月，还需要把当天的日期的样式也需要突出显示一下，还有一个当鼠标移上某个日期的时候，也要有一个移上的效果

```
          <span v-for="j in 7" :key="j" class="cell" :class="[
              // 判断是否是当月，不是当月就变成灰色
              {
                notCurrentMonth: !isCurrentMonth(
                  visibleDate[(i - 1) * 7 + (j - 1)]
                ),
              },
              // 判断是否是当天
              { today: isToday(visibleDate[(i - 1) * 7 + (j - 1)]) },
            ]">
              {{ visibleDate[(i - 1) * 7 + (j - 1)].getDate() }}
            </span>
```

```
   isToday(date) {
      //判断是否是当天
      // 因为this.value是用户传的，不一定是今天，所以需要使用new Date()去获取当天的日期
      let { year, month, day } = utils.getYearMonthDay(new Date());
      let { year: y, month: m, day: d } = utils.getYearMonthDay(date);
      return year === y && month === m && day === d;
    },
```

```
.today{
  background: red;
  color: #fff;
  border-radius: 4px;
}
```

处理鼠标移上的效果

```
.pannel-content .cell {
//...
  box-sizing: border-box;//因为边框会影响到盒子的大小，所以将盒子设置为怪异盒子
}
.pannel-content .cell:hover{
  border: 1px solid pink;
}
```

再给这个面板加上周几

```
data() {
    return {
      weekDays: ['一', '二', '三', '四', '五', '六', '日'],
      isVisible: false, //用来控制面板是否可见
    };
  },
```

```
    <div class="days">
          <span v-for="i in weekDays" :key="`_` + i" class="cell">
            {{ i }}
          </span>
```

注：周几是不需要有鼠标移上的效果，处理一下鼠标移上的样式就好了

接下来处理点击事件，点击面板中的每一个日期，是可以切换输入框中的日期的，要将点击的日期，传给父组件去修改输入框中的值，因为输入框中的值是父组件传过来的，子组件不能去修改，子组件只能通过自定义事件的方式，将要修改的值传给父组件，由父组件去修改

```
<span v-for="j in 7" :key="j" class="cell cell-days"
      @click="chooseDate(visibleDate[(i - 1) * 7 + (j - 1)])"
```

```
chooseDate(date){
    this.$emit('input',date)//触发自定义事件 input ,将当前点击的日期传给父组件
    this.blur()//关闭面板
    }
```

父组件 HomeView.vue

```
    <!-- :value="now" @input="value=>now=value" 因为v-bind和@input组合就是v-model,
所以可以简写 -->
    <HelloWorld v-model="now"></HelloWorld>
```

可以看到在切换之后，它会自动帮我们找到当前选中的那个的日期的月份进行展示，那么我们也可以给她一个样式，选中了那个日期，可以加一个单独的样式

```
// 判断是否是当天
{ today: isToday(visibleDate[(i - 1) * 7 + (j - 1)]) },
 // 判断是否是被选中的日期
{ select: isSelect(visibleDate[(i - 1) * 7 + (j - 1)]) },
//...
```

```
  isSelect(date) {
     //判断当前选中日期，给她添加样式
     let { year, month, day } = utils.getYearMonthDay(this.value);
     let { year: y, month: m, day: d } = utils.getYearMonthDay(date);
     return y === year && month === m && day === d;
   },
```

和鼠标移上事件，一起使用这个样式

```
.pannel-content .cell-days:hover,
.select {
  border: 1px solid pink;
}
```

接下来就可以去处理一下，面板上面展示的当前的年月，以及点击更换年月的效果

先处理展示年月的效果，这个年月不是一个定死的值，会随着你的点击去发生改变，去定义一个值，接收初始传入的年月和修改之后的年月

```
data() {
    //拿到最初传过去的年和月
    let { year, month } = utils.getYearMonthDay(this.value);
    return {
      weekDays: ["一", "二", "三", "四", "五", "六", "日"],
      time: { year, month },//用来接受初始的年和月
      isVisible: false, //用来控制面板是否可见
    };
  },
```

```
        <span>&lt;&lt;</span>
        <span>{{ time.year }}年</span>
        <span>{{ time.month + 1 }}月</span>
        <span>&gt;&gt;</span>
```

当点击切换日期的时候也需要更改面板上面的月份

```
chooseDate(date) {
      this.time = utils.getYearMonthDay(date); //当点击面板中的日期时，面板上面的日期也需要随之一起修改
      this.$emit("input", date);
      this.blur(); //关闭面板
    },
```

还有一个对当前的年和月进行切换，上一年和下一年，上一个月和下一个月

```
        <span @click="pervMonth">&lt;&lt;</span>
        <span>{{ time.year }}年</span>
        <span>{{ time.month + 1 }}月</span>
        <span @click="nextMonth">&gt;&gt;</span>
```

当前面板展示的日期要和修改的之后的年月是一致的

```
 visibleDate() {
      //当前面板要展示的所有日期
      // 先获取当前是周几
      let { year, month } = utils.getYearMonthDay(
        utils.getDate(this.time.year, this.time.month, 1)
      );
```

还有展示是否是当月的效果也需要修改一下,不能通过this.value去判断是否是当月，要用点击切换之后的日期去对比

```
isCurrentMonth(date) {
      //判断是否是当月
      let { year, month } = utils.getYearMonthDay(utils.getDate(this.time.year,
this.time.month, 1));
      let { year: y, month: m } = utils.getYearMonthDay(date);
      return y === year && month === m;
    },
```

```
    pervMonth() {//上一个月
      // 先拿到当前的年月
      let d = utils.getDate(this.time.year, this.time.month, 1);
      // 使用这个日期对象的方法，去设置月份
      d.setMonth(d.getMonth() - 1);//值为 -1 将导致上一年的最后一个月 12 将导致明年的第
一个月
      // 将更改之后日期重新赋值给this.time
      this.time = utils.getYearMonthDay(d);
    },
    nextMonth() {//下一个月
      // 先拿到当前的年月
      let d = utils.getDate(this.time.year, this.time.month, 1);
      // 使用这个日期对象的方法，去设置月份
      d.setMonth(d.getMonth() + 1);
      this.time = utils.getYearMonthDay(d);
    },
```

那么年份也是一样的，会写月份了，年份也就简单了，整个的效果我们就实现到这里，后面大家也可以根据的情况去优化这个

学习到这，有兴趣的可以来参加我们系统课程，系统课程有更多的精彩的实用的内容等你，现在找助教老师获取优惠，原价**7999**，现在预交**100元**，领取优惠名额，只要**5999**，预交**100**，剩下的**5899**分期免息，买一期送一期，一期学不会，下期接着来，很划算的哟

心动不如行动，想提升自己的技术，拿取一份高薪满意的薪资就来加入我们吧，**金渡不会让你失望**