字节青训营 - 移动端专题

讲师介绍

莫涛（一个脑门倍儿亮的不正经程序员）

原妙味课堂联合创始人 ，妙味上海教研负责人

《React 工程师修炼指南》作者

《JavaScript 修炼之道》,《HTML+CSS 修炼之道》书籍内容提供者

中华女子学院特学前端讲师

课程安排

- 移动端适配
- touch 事件
- 移动端手势处理
- 项目实战

课程大纲

一、移动端适配

视口设置

- viewport

属性名	取值	描述
width	正整数或 device-width	定义视口的宽度，单位为像素
height	正整数或 device-height	定义视口的高度，单位为像素，一般不用
initial-scale	[0.0-10.0]	定义初始缩放值
minimum-scale	[0.0-10.0]	定义放大最大比例，它必须小于或等于 maximum-scale 设置
maximum-scale	[0.0-10.0]	定义缩小最小比例，它必须大于或等于 minimum-scale 设置
user-scalable	yes / no	定义是否允许用户手动缩放页面，默认值 yes

- dpr

```typescript
(function () {
      var metaEl = document.createElement('meta');
      var scale = devicePixelRatio;
      metaEl.setAttribute('name', 'viewport');
      metaEl.setAttribute('content', 'initial-scale=' + (1/scale) + ', maximum-scale=' + (1/scale) + ', minimum-scal
      document.documentElement.firstElementChild.appendChild(metaEl);
})();
```

TypeScript

  - 物理像素：(设备像素，device pixels)
  - CSS 像素：(css pixels)

页面适配方案

- 百分比
- rem
- vw
- media:https://developer.mozilla.org/zh-CN/docs/Web/CSS/@media
  - width
  - min-width
  - max-width

二、移动端事件

touch 事件

- touchstart
- touchmove
- touchend

Mac 下调试移动端

- 在 iOS 设备上打开允许调试：设置→Safari→高级→打开”web 检查器“
- 在 MAC 上打开 Safari 的开发菜单：顶部菜单栏“Safari”→偏好设置→高级→打开”在菜单栏中显示“开发”菜单
- 在 iOS 设备上的 Safari 浏览器中打开要调试的页面，然后切换到 MAC 的 Safari，在顶部菜单栏选择“开发”→找到你的 iOS 设备名称→右边二级菜单选择需要调试的对应标签页，即可开始远程调试
- 小工具推荐

  anywhere 基于 node 的本地服务器

  - 安装 `npm i anywhere -g`
  - 启动 `anywhere` `anywhere -p 8888`

移动端默认事件阻止

- 阻止 touchstart 默认事件
- 阻止 touchmove 默认事件
- 阻止 touchend 默认事件

TouchEvent

- changedTouches
- targetTouches
- touches

拖拽原理

Mouse 事件机制和 Touch 事件机制的差异

- Mouse 事件拖拽实现
- Touch 事件拖拽实现

三、手势库封装

Mouse 事件与 Touch 事件的兼容处理

常用事件封装

- 常用事件
  - start、move、end

  - pressstart、pressend
  - tap
  - panstart、pan、panend
- 自定义事件
  - new CustomEvent
  - elemnt.dispatchEvent

```
function enableGesture(element) {
    let contexts = [];
    const mouse_type = Symbol("mouse");
    if (!("ontouchstart" in document)) {
        // PC
        element.addEventListener("mousedown", (event) => {
            let move = (event) => {
                onMove(event, contexts[mouse_type]);
            };
            let end = (event) => {
                onEnd(event, contexts[mouse_type]);
                document.removeEventListener("mousemove", move);
            }
            document.addEventListener("mousemove", move);
            contexts[mouse_type] = {};
            onStart(event, contexts[mouse_type]);
            document.addEventListener("mouseup", end, { once: true });
        });
    }
    element.addEventListener("touchstart", (event) => {
        for (let touch of event.changedTouches) {
            contexts[touch.identifier] = {};
            onStart(touch, contexts[touch.identifier]);
        }
    });
    element.addEventListener("touchmove", (event) => {
        const stop = ()=>{
            event.preventDefault();
        }
        for (let touch of event.changedTouches) {
            touch.stop = stop;
            onMove(touch, contexts[touch.identifier]);
        }
    });
    element.addEventListener("touchend", (event) => {
        for (let touch of event.changedTouches) {
            onEnd(touch, contexts[touch.identifier]);
            delete contexts[touch.identifier];
        }
    });

    let onStart = (point, context) => {
        element.dispatchEvent(Object.assign(new CustomEvent('start'), {
            startX: point.clientX,
            startY: point.clientY,
            clientX: point.clientX,
            ClientY: point.clientY
        }));
        context.startX = point.clientX;
        context.startY = point.clientY;
        context.isTap = true; // 点击
        context.isPan = false; // 滑屏
        context.isPress = false; // 长按
        context.timoutHandler = setTimeout(() => {
            if (context.isPan) return;
            context.isTap = false;
            context.isPress = true;
            element.dispatchEvent(Object.assign(new CustomEvent('pressstart'), {
                clientX: point.clientX,
                ClientY: point.clientY
            }))
        }, 300);
    };
    let onMove = (point, context) => {
        let dx = point.clientX - context.startX;
        let dy = point.clientY - context.startY;
        if (!context.isPan && dx ** 2 + dy ** 2 > 100) {
            clearTimeout(context.timoutHandler);
            context.isTap = false;
            context.isPan = true;
            context.isPress = false;
            element.dispatchEvent(Object.assign(new CustomEvent("panstart"), {
                startX: context.startX,
                startY: context.startY,
                clientX: point.clientX,
                clientY: point.clientY,
                stop: point.stop
            }));
            if(context.isPress){
                element.dispatchEvent(new CustomEvent('presscancel'))
            }
            return ;
        }
        if (context.isPan) {
            element.dispatchEvent(Object.assign(new CustomEvent("pan"), {
                startX: context.startX,
                startY: context.startY,
                clientX: point.clientX,
                clientY: point.clientY,
                stop: point.stop
            }));
        }
        element.dispatchEvent(Object.assign(new CustomEvent("move"), {
            clientX: point.clientX,
            clientY: point.clientY
        }))
    };
    let onEnd = (point, context) => {
        clearTimeout(context.timoutHandler);
        if (context.isPan) {
            element.dispatchEvent(Object.assign(new CustomEvent('panend'), {
                startX: context.startX,
                startY: context.startY,
                clientX: point.clientX,
                clientY: point.clientY
            }))
        }
        if (context.isTap) {
            element.dispatchEvent(Object.assign(new CustomEvent("tap"), {
                clientX: point.clientX,
                clientY: point.clientY
            }));
        }
        if (context.isPress) {
            element.dispatchEvent(Object.assign(new CustomEvent("pressend"), {
                clientX: point.clientX,
                clientY: point.clientY
            }));
        }
        element.dispatchEvent(Object.assign(new CustomEvent("end"), {
            clientX: point.clientX,
            clientY: point.clientY
        }))
    }
}
```

TypeScript

四、移动端轮播图实战

1. 布局

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Document</title>
    <script>
        (function () {
            var metaEl = document.createElement('meta');
            var scale = devicePixelRatio;
            metaEl.setAttribute('name', 'viewport');
            metaEl.setAttribute('content', 'initial-scale=' + (1/scale) + ', maximum-scale=' + (1/scale) + ', minimu
            document.documentElement.firstElementChild.appendChild(metaEl);
        })();
    </script>
    <style>
        body {
            margin: 0;
            font-size: 4vw;
            line-height: 1.5;
        }
        ul {
            margin: 0;
            padding: 0;
            list-style: none;
        }
        #banner {
            position: relative;
            width: 100vw;
            overflow: hidden;
            text-align: center;
        }
        #banner_pic {
            display: flex;
            transform: translateX(0);
        }
        #banner_pic li {
            flex: none;
            width: 100vw;
        }
        #banner_pic img {
            display: block;
            width: 100%;
        }
        #banner_nav {
            position: absolute;
            left: 10vw;
            bottom: 2vw;
        }
        #banner_nav span {
            float: left;
            margin: 0 .5vw;
            width: 4vw;
            height: 1vw;
            background: #fff;
        }
        #banner_nav span.active {
            background: blue;
        }
    </style>
</head>
<body>
    <div id="banner">
        <ul id="banner_pic">
            <li>
                <img src="https://static001.geekbang.org/resource/image/bb/21/bb38fb7c1073eaee1755f81131f11d21.jpg"/
            </li>
            <li>
                <img src="https://static001.geekbang.org/resource/image/1b/21/1b809d9a2bdf3ecc481322d7c9223c21.jpg"
            </li>
            <li>
                <img src="https://static001.geekbang.org/resource/image/b6/4f/b6d65b2f12646a9fd6b8cb2b020d754f.jpg"
            </li>
            <li>
                <img src="https://static001.geekbang.org/resource/image/73/e4/730ea9c393def7975deceb48b3eb6fe4.jpg"
            </li>
        </ul>
        <nav id="banner_nav">
            <span class="active"></span>
```

```
                <span></span>
                <span></span>
                <span></span>
            </nav>
        </div>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
        <h3>占位1</h3>
    </body>
    </html>
```
TypeScript

2. 幻灯片拖拽实现

3. 判断用户滑屏方向

4. 动画实现

5. 无缝轮动

6. 触碰悬停

7. 自动播放

```
class Carousel {
    constructor(opt) {
        for(let s in opt){
            this[s] = opt[s];
        }
        let {wrap} = opt;
        this.parent = wrap.parentNode;
        this.viewWidth = this.parent.clientWidth;
        this.isAnimate = false;
        this.isMove = false;
        this.animateTime = 0;
        this.isBreak = false;
        this.initLayout();
        this.imgsLen = wrap.children.length;
        enableGesture(wrap);
        wrap.addEventListener("start", this.start);
        wrap.addEventListener("panstart", this.panstart);
        wrap.addEventListener("pan", this.move);
        wrap.addEventListener("panend", this.end);
        wrap.addEventListener("end", ()=>{
            if(this.isBreak){
                this.isBreak = false;
                this.end();
            }
            this.autoPlay();
        });
        wrap.querySelectorAll("img").forEach(item => {
            item.addEventListener("dragstart", event => event.preventDefault());
        });
        this.autoPlay();
    }
    start = ()=>{
        if(this.animateTime){
            clearInterval(this.animateTime);
            this.isBreak = true;
        } else {
            this.isBreak = false;
        }
        clearInterval(this.autoTimer);
    }
    panstart = (e) => {
        let dx = e.clientX - e.startX,dy = e.clientY - e.startY;
        if(Math.abs(dx) > Math.abs(dy)){
            this.isMove = true;
        }
        if(this.isMove){
            this.init();
            this.offsetX = this.x;
            e.stop();
        }
    };
    move = (e) => {
        if (this.isMove) {
            let disX = e.clientX - e.startX;
            this.x = this.offsetX + disX;
            this.setTransform();
            e.stop();
        }
    };
    end = (e) => {
        this.isMove = false;
        this.index = Math.round(-this.x/this.viewWidth);
        let targetX = -this.index*this.viewWidth;
        if(Math.abs(targetX - this.x)>20){
            this.animate(targetX);
        } else {
            this.x = targetX;
            this.setTransform();
        }
        this.setNavs();
    };
    initLayout(){
        const imgs = this.wrap.children;
        const fastChild  = imgs[0];
        const lastChild  = imgs[imgs.length-1];
        this.wrap.insertBefore(lastChild.cloneNode(true),fastChild);
        this.wrap.appendChild(fastChild.cloneNode(true));
        this.x = -this.viewWidth;
        this.index = 1;
        this.setTransform();
    }
    init() {
        if(this.index === 0||this.index===this.imgsLen-1){
            this.resetLayout();
        }
    }
    resetLayout(){
        let targetIndex = -this.index*this.viewWidth;
        let disX = targetIndex - this.x;
        if(this.index === 0){
            this.index = this.imgsLen - 2;
        } else if(this.index === this.imgsLen - 1){
            this.index = 1;
        }
        this.x = -this.index*this.viewWidth + disX;
        this.setTransform();
    }
    autoPlay(){
        this.autoTimer = setInterval(()=>{
            if(this.index === this.imgsLen-1){
                this.resetLayout();
            }
            this.index++;
            this.animate(-this.index*this.viewWidth);
            this.setNavs();
        },3000);
    }
    animate(targetX) {
        const time = Math.abs(targetX - this.x);
        let t = 0;
        let b = this.x;
        let c = targetX - this.x;
        let d = Math.ceil(time/(1000/60));
        clearTimeout(this.animateTime);
        this.animateTime = setInterval(()=>{
            t++;
            if(t === d){
                clearInterval(this.animateTime);
                this.animateTime = 0;
            }
            this.x = this.easeOut(t,b,c,d);
            this.setTransform();
        },1000/60);
    }
    /*
    t: current time（当前时间）；
    b: beginning value（初始值）；
    c: change in value（变化量）；
    d: duration（持续时间）。
    */
    easeOut (t, b, c, d) {
        return -c *(t/=d)*(t-2) + b;
    }
    setTransform(){
        this.wrap.style.transform = `translate3d(${this.x}px,0,0)`;
    }
    setNavs() {
        if (!this.navs.length) {
            return;
        }
        this.navs.forEach(nav => {
            nav.className = ""
        });
        const nowIndex = this.index>0?(this.index - 1)%this.navs.length:this.navs.length-1;
        this.navs[nowIndex].className = "active";
    }
}
```

TypeScript