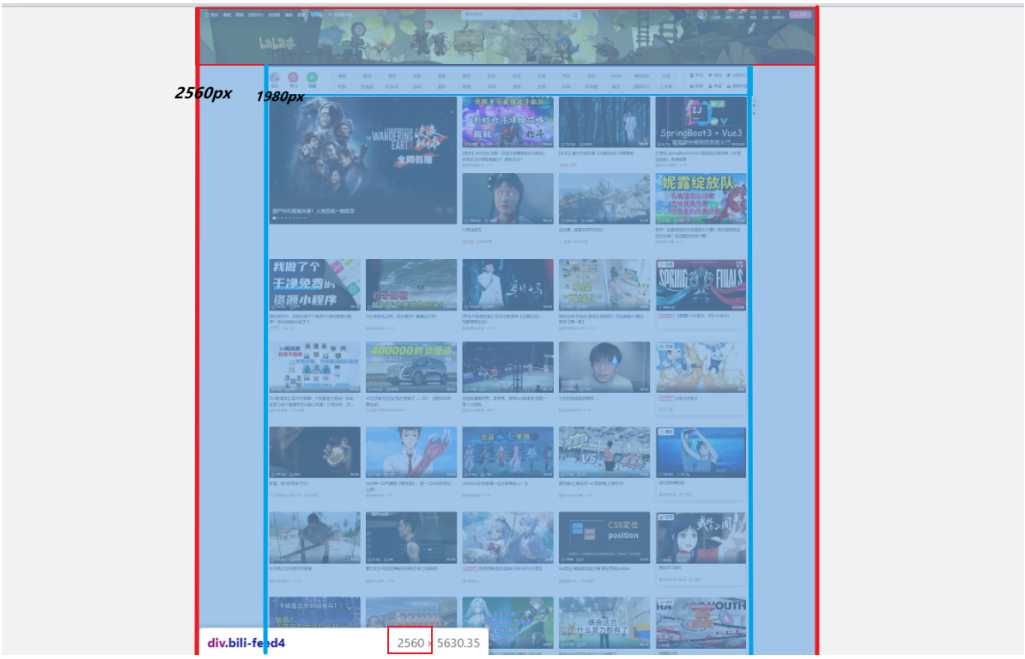
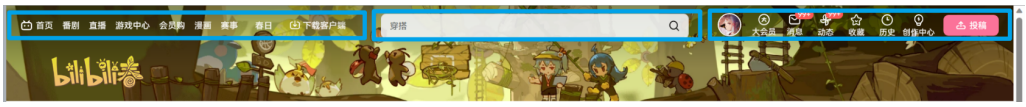
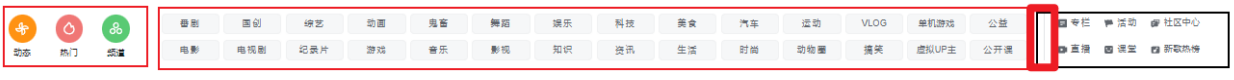
首页

页面布局

> 使用**aspect-ratio固定盒子的宽高**







瀑布流无限加载效果

> 监听scroll 当页面滑到下面的时候就创建新的视频然后再实现懒加载效果

注意节流

```
window.addEventListener("scroll", throttle(() => {
    //回调函数
    //如果页面划到了距离最下面有一定距离的地方，就渲染新的数据
    const scrollTop = document.documentElement.scrollTop || document.body.scrollTop//文档的滚动高度
    const scrollHeight = document.documentElement.scrollHeight || document.body.scrollHeight//文档的总高度
    const clientHeight = document.documentElement.clientHeight || window.innerHeight//当前视窗的高度
    //逻辑或是为了兼容性
    if (scrollTop + clientHeight >= scrollHeight - 100) {//视窗高度加滚动高度接近总高度了
          //渲染新的html
        addVideo()
    }
}, 500))
```

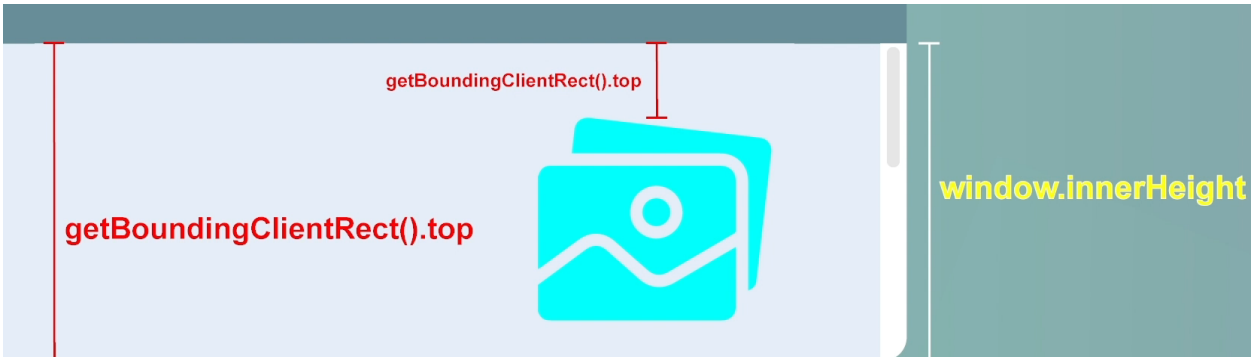
每一个视频的标签对应如下

```
<section><video src="">视频</video>
    <p class="videoTitle">标题</p>
    <p class="info">info</p>
</section>
```

> 💡 注意，当页面比较长的时候我们也要判断并用视频将页面填满

```
(function checking() {
        const scrollTop = document.documentElement.scrollTop || document.body.scrollTop
        const scrollHeight = document.documentElement.scrollHeight || document.body.scrollHeight
        const clientHeight = document.documentElement.clientHeight || window.innerHeight
        if (scrollHeight <= clientHeight) {
            addVideo()
            return checking()
        }
        else return
    }());//立刻执行函数记得加分号
```

懒加载效果



IntersectionObserver交叉观察

盒子类名变化

懒加载创建盒子：loading → 用ajax获取 → reading → 渲染完成

用ajax： fetch获取视频

```
function askForVideo(sections) {//参数是要请求加载的视频列表
    let observer = new IntersectionObserver((entries) => {
        //先发送ajax请求然后再forEach渲染
        fetch('https://frontend.exam.aliyun.topviewclub.cn/api/getHomePageVideo',
            {
                method: "GET"
            }
        ).then(response => {
            if (response.status >= 200 && response.status <= 300) {
                if (response.ok = true) {
                    return response.json()
                }
            }
        }).then(response => {
            //得到数据了可以渲染了
            let i = 0
            entries.forEach(entry => {
                const src = response.videos[i].videoSrc
                const title = response.videos[i].title
                const authour = response.videos[i].authour
                const describe = response.videos[i].describe
                const uuid = response.videos[i].uuid
                const authorAvatarSrc = response.videos[i].authorAvatarSrc
                entry.target.innerHTML = `
                <p class="videoTitle">${title}</p>
                <p class="discribe">${describe}</p>
                `
                //测试的时候这里战术没有写视频进去
                entry.target.classList.remove('loading')
                entry.target.classList.add('ready')
                observer.unobserve(entry.target)
                i = (i === 9) ? 0 : i + 1
            })
            // 加载完之后把类名改ready，并取消观察
        })
    }, { threshold: 1 })
```

```
    //创建交叉观察对象

    //观察放视频的section
    sections.forEach((section) => {
        observer.observe(section)
    })
}
```

悬停播放视频

> [html video API]

伪代码：

首先添加一个伪元素{

绝对定位在下方，渐变颜色}，显示的是视频的时长和播放，弹幕等

男单决赛-第1场

UP 乌拉拉拉拉拉啦 · 15小时前

男单决赛-第1场

UP 乌拉拉拉拉拉啦 · 15小时前

hover之后（mouseenter）视频开始播放，而且下方的信息消失，出现一个加入待会看列表的的图标（不实现）

> **⚠️如何实现伪元素里面的布局**

💡 Uncaught (in promise) DOMException: play() failed because the user didn't interact with the document first：浏览器阻止了没有用户交互的自动播放，除非设置muted = true不然就不会播放

点击进入详情页

> 实现点击跳转

```
window.location.href = 'url'//给当前的url重新分配，当然就会取代这个页面
window.location.replace("url")//直接取代
window.open('url')//会产生新的页面，相当于a标签的target="_blank" /用这个
```

> url传参Params