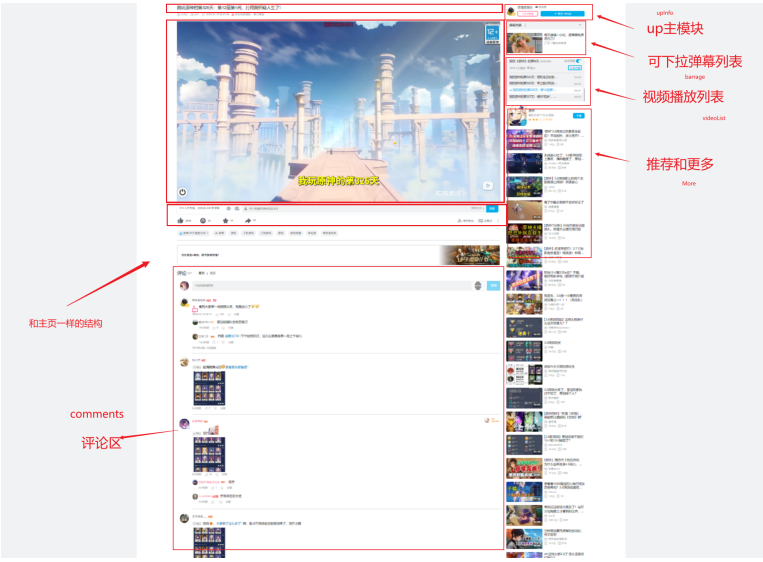
视频详情页

页面布局



数据来源

⚠️ 点击跳转的url在这里被定义，通过urlParam传参，在这里我直接传入了：`replaceUrl(entry.target.firstElementChild, 'video.html?' + "src=" + `${JSON.stringify(response)}` + '&current=' + `${i}`)` 在video详情界面跳转链接会显得有一些冗长。每一次的ajax请求获得的视频列表的顺序时不同的，为了点击之后在详情界面的视频列表中获得的数据是本次ajax的结果，保证视频列表的正确性，所以才出此下策。当有更好的解决方法。

```
const urlParams = new URLSearchParams(window.location.search);
const response = JSON.parse(urlParams.get('src'));
let i = +urlParams.get('current')
```

这个i在video.js中作为全局变量，表示当前播放的视频在这个视频列表中的位置，同时也用于切换上一个和下一个视频

功能

HTMLMediaElement API

区分登录状态

- 通过localStroage中的login的真值，来判断是否登录
  - 未登录状态执行signin.js中的创建模态框函数（刷新页面，页面开始就会加载登录模态框）
- 未登录和已登录的样式通过修改类名来区分
- 点击需要登陆的功能时，也是跳转到登录界面

控件

暂停播放

调用`PoP(videoObj.src)`暂停视频

> 剩余参数

```
function pOp(src, force = false, isToPasue = true) {
    //false表示要主动控制而不是通过检测视频状态
    if (force) {
        if (isToPasue) {
```

```
            src.pause()
            playIcon.style.opacity = '1'
            popBtn.querySelector('.onPlay').style.display = 'none'
            popBtn.querySelector('.onPause').style.display = 'block'

            barPause(true)
        }
        else {
            src.play()
            playIcon.style.opacity = '0'
            popBtn.querySelector('.onPlay').style.display = 'block'
            popBtn.querySelector('.onPause').style.display = 'none'
            barPause(false)
        }
    }
    else {
        if (src.paused) {
            src.play()
            playIcon.style.opacity = '0'
            popBtn.querySelector('.onPlay').style.display = 'block'
            popBtn.querySelector('.onPause').style.display = 'none'
            barPause(false)
        }
        else if (!src.paused) {
            src.pause()
            playIcon.style.opacity = '1'
            popBtn.querySelector('.onPlay').style.display = 'none'
            popBtn.querySelector('.onPause').style.display = 'block'
            barPause(true)
        }
    }

}
```

调用`barPause(isToPasue = true)`暂停弹幕

```
function barPause(isToPasue = true) {
    const bar = document.querySelectorAll('.content .left .video .runningBarrage p')
    bar.forEach(
        (element) => {
            if (isToPasue) element.style.animationPlayState = 'paused'
            else { element.style.animationPlayState = 'running' }
        }
    )
}
```

倍速播放

`audio.playbackRate = value` 比如(1,2…)

音量

`video.volume = 0.0 - 1.0`

开启或者关闭弹幕

- 直接控制所有弹幕的display即可

```
//开启和关闭弹幕
const barBtn = document.querySelector('.content .left .sendBarrage #barrageToggle')
barBtn.addEventListener('click', (e) => {
    runningBarrage.classList.toggle('close')
    barBtn.querySelector('.barOn').classList.toggle("active")
    barBtn.querySelector('.barOff').classList.toggle("active")
})
```

可拖动进度条

> 这个远离和可拖动的音量大同小异

`video.current = time`

其实要做的事把拖动得到的长度换算为时间给到接口

- 获取鼠标的坐标： `e.pageX/e.pageY`
- 获取元素的坐标：`box.offsetLeft/box.offsetTop` (都是相对页面的)

可点击进度条

- 点击事件获取到坐标
- 换算后给视频`currentTime`
- 刷新播放进度条`refreshProcessingBar()`

```
//可点击进度条实现
videoObj.progressBar.addEventListener('click', (mouse) => {
    const deltaX = mouse.clientX - videoObj.barChild.getBoundingClientRect().left
    const length = videoObj.progressBar.offsetWidth
    videoObj.src.currentTime = deltaX / length * videoObj.src.duration
    refreshProcessingBar()
})
```

切换视频

通过切换全部变量i，将i传入刷新视频的函数，加载新的视频

自动连播

开关

全局变量`let isAutoContinue = true`

- 点击按钮换样式
- 点击按钮修改`isAutoContinue`

连播`addEventListener('ended', () => {})`

```
videoObj.src.addEventListener('ended', () => {
    if (isAutoContinue) {
        if (i === response.videos.length - 1) {
            //这里显示播放完成的界面
            return;
        }
        else {
            i++
            if (i === response.videos.length - 1) {
                //改变按键颜色
            }
        }
        refreshVideo(i)
        upInfo(i)
        refreshWatermark(i)
        refreshVideoList(i)
        barClear()
        barSending(i)
        refreshComments(i)
        refreshCondition(i)
    }

})
```

弹幕模块

> 需要用到的知识：

> 本地存储
>
> 构造函数和原型
>
> class 类
>
> 函数
>
> Intersection observer 交叉观察

伪代码：

找到对应弹幕

> 将所有的时间拿出来放到一个数组里，时间对上了就找出来 需要用到内置的方法

用interval不断获取视频的时间，如果发现时间对上了，就拿出对应的弹幕对象来发送

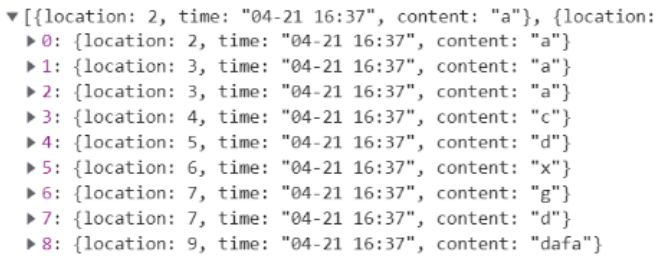
给对应弹幕添加动画效果

直接写好动画，然后用js给动画dom添加绑定动画就好

弹幕离开屏幕之后删除弹幕

通过在添加弹幕时，延时给弹幕添加一个交叉观察，如果弹幕与是视频没有交叉了，就删除这个弹幕

数据结构



存储数据的应当是一个以名字加barrage定义的localStorage数组，按顺序包含了许多个弹幕对象，弹幕对象里面储存了弹幕的时间，内容，和发送时间；传送弹幕的应当是一个函

数，接受一个弹幕对象

这是一个构造函数，每一次产生新的弹幕都会用这个对象，然后放到barrage数组里，然后再放到localStorage里面

```
function BarrageInfo(location, time, content)//在视频中的定位的时间，time是现实时间，content就是内容
{
    this.location = location
    this.time = time
    this.content = content
}
```

发送弹幕函数 `creatBarrage(content)`

```
function creatBarrage(content) {
    const bar = document.createElement('p')
    bar.innerText = content
    runningBarrage.appendChild(bar)
    const setTime = Math.floor(Math.random() * 3 + 6) + 's'
    bar.style.animationDuration = setTime
    bar.style.animationPlayState = 'running'
    setTimeout(() => {
        observer.observe(bar)
    }, 2000);
    //何时删掉这个弹幕？
}
```

匹配弹幕函数 `barSending()`

利用这个一秒一次的定时器顺便修改了视频的时间

```
//拿出弹幕并准备发送的函数
//顺便改视频时间
let interval = null;
function barSending(n) {
    if (interval) { clearInterval(interval) }
    let barObj = JSON.parse(localStorage.getItem(`${response.videos[n].title}`))
    barAddToList(barObj)
    interval = setInterval(() => {//全局变量

        //更改视频播放时间和进度条
        videoObj.timer.innerText = `
    ${formation(videoObj.src.currentTime)}/${formation(videoObj.src.duration)}`
        refreshProcessingBar()
        if (!videoObj.src.paused) {
```

```
            if (barObj) {
                const _bar = barObj.filter((target) => {
                    return target.location + 1 === Math.floor(videoObj.src.currentTime)
                }).forEach(element => {
                    creatBarrage(element.content)
                })
            }
        }
    }, 1000);
}
```

⚠️ **出现问题，删除弹幕的时候，屏幕中的弹幕会出现位移，非常影响观感**

解决方法：尝试脱标，让每个盒子是属于绝对定位，这样的话就需要计算和随机每次生成弹幕的位置，

连播

> 判断已经播放完毕可以用ended事件

`videoObj.src.addEventListener('ended',()=>{执行下一个})`

切换

> 要做到连带切换标题，视频列表，弹幕列表，评论和up信息

⚠️ 切换弹幕列表的时候出现问题：
在调用sort()方法时报错，认为我对null调用的sort()方法，但显然不是，
就其原因，sort()方法内部报错是TypeError: 'caller', 'callee', and 'arguments' properties may not be accessed on strict mode functions or the arguments objects for calls to them at Function，严格模式下不能访问caller,callee 不懂。
将sort的箭头函数改为具名函数之后，问题出现了变化，左右切换一次没有问题，但是第二次开始又出现原来的问题

原来是没有判断传入的barObj是不是空的，因为没有任何弹幕的时候，localStorage里也没有这个对象，所以是null

点赞投币收藏

> 用到css动画的clippath，以及在css文件中使用变量

水印

绝对定位字体图标

通过视频的宽度判断位置

评论模块

> `async function addComment(target, father, isReply = false, btn) 添加回复`
> `async function replying(target, isReply = false, fatherBtn) 添加回复框`
> `async function refreshComments(i)刷新`

添加回复框函数 `async function replying(target, isReply = false, fatherBtn)`

- target是放评论输入框的盒子，fatherBtn是点击事件的target，用来定位
- 渲染一个输入框，用于输入你的回复
- 为发送按钮绑定事件，执行addComment函数

添加评论函数 `async function addComment(target, father, isReply = false, btn)`

- target是input框，father是放comment的地方，isReply表示这是一个评论还是一个回复，btn是点击的target
- 获取`target.value`
- 渲染评论，如果回复的不是楼主，添加@谁谁谁

- 根据isReply的真值，绑定按键的回复和删除，以及保存数据的方式有略微不同
  - 绑定评论中按键功能：回复，点赞，踩，删除
  - 结构化数据，保存到localStorage

刷新函数 `async function refreshComments(i)`

- 从localStroage中拿出数据，渲染到评论区
- 绑定按键，这里要判断这个发送者是不是登录了，登陆了才渲染删除按钮，如果不是本人登录，就没有这个删除按钮

评论数据结构

> 类比弹幕，但是评论要区分是评论还是回复

- 首先要用视频的名字来命名comment的本地存储的值，只有产生了发送事件才会创建这样一个数据，所以这个数据的创建和添加放在评论发布的函数里，同时在回复的函数里也要保存这个回复
- 每个视频的comment存储里有一个数组，数组是母评论，在这些元素里面，还有一些属性和数组，这些数组是回复，回复和评论套用同一个构造函数，构造函数结构如下

```
//保存评论的JSON对象
function commentInfo(avata, uname, time, index, reply) {
    this.avata = avata
    this.uname = uname
    this.time = time
    this.index = index
    this.reply = reply//储存reply
}
```

🚫评论计数n

用于标志每个评论的位置，让回复正确插入回复的那个框框，所以，我们要将n存储而且每次使用应该是最新的

🚫 这个在存储的时候很不方便，优化掉，直接变成元素的下标不是更好？

修改：创建这个n的目的是让回复的窗口正确出现，那么我们就在没有n的情况下实现这个框框正确显示，直接用parentNode就行了

🚫 **bug 在切换视频之后再次评论会显示reply is not a function**

💡 reply有同名对象属性，在同一个作用域中出现同名，reply是函数的事实可能被覆盖，所以有reply is not a function 只需要把reply() 的调用和声明改成replying()就行了

刷新

```
refreshVideo(i)//加载视频
upInfo(i)//加载up信息
refreshWatermark(i)//加载水印
refreshVideoList(i)//加载视频列表
barClear()//清除当前弹幕
barSending(i)//重新加载弹幕
refreshComments(i)//重新加载评论
refreshCondition(i)//重新加载视频状态
```

视频数据

> 以名字+condition为键保存在localStroage中

```
// 弹幕数量,评论数量，点赞和播放数量
function Condition(subAmount, viewAmount, barAmount) {
    this.subAmount = subAmount
    this.viewAmount = viewAmount
    this.barAmount = barAmount
}
```