同济大学

TONGJI UNIVERSITY

《WEB 技术》

实验报告

实验名称	播放器实现
小组成员	
学院（系）	电子与信息工程学院
专　　业	计算机科学与技术专业
任课教师	郭玉臣
日　　期	2023 年　4 月 13 日

1. 实验原理

本次实验的主要目标是用 js 设计一个在线媒体播放器,其中涉及到 html 中 video 元素的相关控制、视频基本控制模块 js 实现：时间跳转、音量控制、倍速控制以及视频列表选择等。

2. 实验基本要求

基本功能：

2.1 实现媒体播放器的基本功能，包括播放、暂停、音量控制、全屏等。

2.2 播放和暂停功能：可以通过获取媒体元素（audio 或 video）并使用 play()和 pause()方法来实现。可以在媒体元素上设置事件监听器，例如点击播放按钮时触发 play()方法，在点击暂停按钮时触发 pause()方法。

2.3 音量控制功能：可以通过获取媒体元素的 volume 属性并设置其值来实现音量控制。可以在页面上添加音量控制滑块，当用户滑动滑块时，可以通过事件监听器将音量值更新到媒体元素的 volume 属性中。

2.4 全屏功能：可以通过获取媒体元素的 requestFullscreen()方法来实现全屏功能。当用户点击全屏按钮时，可以触发该方法，并将媒体元素全屏显示。

扩展功能：

2.5 倍速功能：可以通过获取媒体元素的 playbackRate 属性并设置其值来实现倍速功能。可以在页面上添加倍速控制按钮，当用户点击按钮时，可以通过事件监听器将播放速度更新到媒体元素的 playbackRate 属性中。

2.6 进度条功能：可以在页面上添加进度条，并通过获取媒体元素的 currentTime 属性和 duration 属性来计算出当前播放进度百分比，并将其显示在进度条上。当用户点击进度条时，可以通过事件监听器获取点击位置的百分比，并将其设置为媒体元素的 currentTime 属性值，以实现跳转播放。

2.7 播放列表功能：可以使用数组存储多个媒体文件信息，例如媒体文件的 URL、标题、作者等。可以在页面上添加播放列表，并使用循环遍历数组，并根据用户选择的媒体文件播放对应的文件。当用户点击播放列表中的某个媒体文件时，可以通过事件监听器获取该文件的 URL，并将其设置为媒体元素的 src 属性值，以实现播放该文件。

3. 实验内容

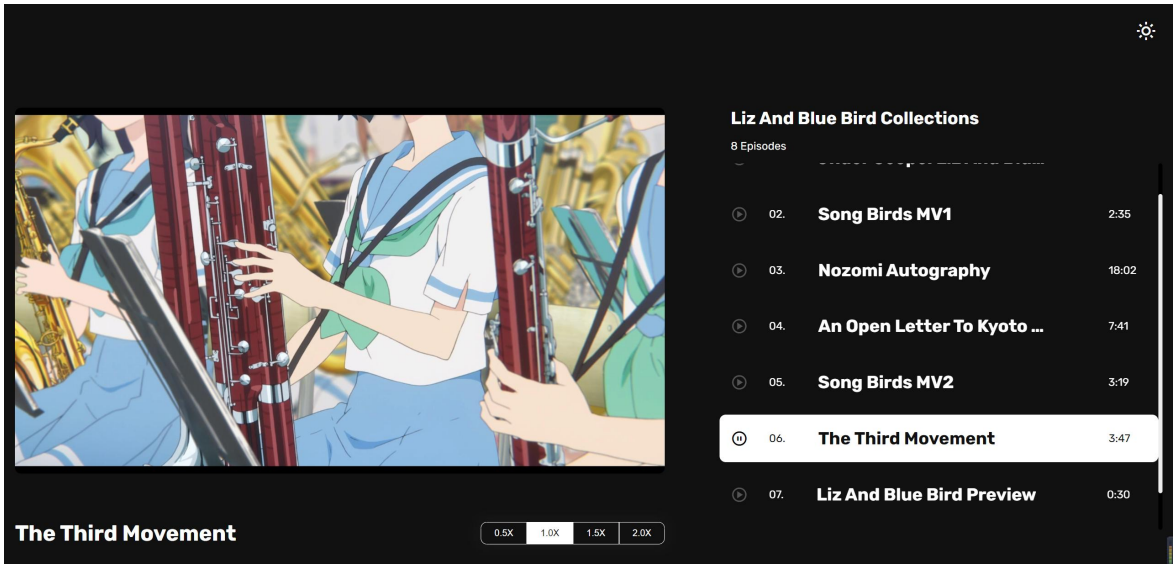
小节目录：

3.1 整体结构

页面整体分为两部分，分别是播放器视频主体部分和列表选择部分，用 grid 布局统一进行管理

其中左侧分为视频主体和视频标题，扩展功能进度条、音量等悬浮在视频播放器内部，倍速功能在播放器外部。

右侧包括视频集合标题，视频选择列表。列表可进行滚动显示，其中每一条视频选择项包括图标标识、视频名称、视频时长。可供用户进行视频选择



全局变量设置；

包括颜色、间距、阴影等参数，保证各个部分之间的一致性以及方便后续夜间模式对整体颜色的修改

```
:root {
  --background-color: #fff;
  --text: #000;
  --video-hover: #eee;
  --video-selected: #000;
  --video-selected-text: #fff;
  --border-radius-sm: 9px;
  --margin-ssm: 1rem;
  --margin-sm: 2rem;
  --margin-md: 6rem;
  --margin-mx: 9rem;
  --fontsize-sm: 1.6rem;
  --fontsize-md: 2rem;
  --scrollbar-backgroundcolor: #fff;
  --scrollbar-thumbcolor: #000;
```

```
    --play-icon-normal: #555;
    --play-icon-hover: #fff;
    --play-icon-selected: #fff;
    --shadow: rgb(0, 0, 0, 0.02);
    --transition-time: 0.3s;
}
```

3.2 播放器基本样式实现

视频：在 html 中利用 video 标签指定视频播放，其中具有 autoplay 控制自动播放、muted 控制起始静音等属性，注意 control 属性可以利用浏览器中的内置组件对视频进行控制，但其基本样式很难自主调节，故此处并不使用

（扩展功能介绍在 3.4 扩展功能实现部分）

<Html>:

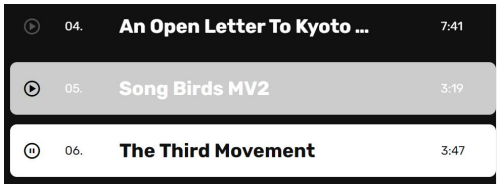
```
<video
 class="video-onlive"
 src="videos/video1.mp4"
 autoplay
 muted
 ></video>
```

<css>

```
.main-video video {
  width: 100%;
  border-radius: var(--border-radius-sm);
  margin-bottom: var(--margin-sm);
}
```

视频选择单元:视频选择单元包括左侧显示播放小图标、视频编号、视频标题、视频时长

视频选择栏主体、滚动 thumb：



<html>

```
<div class="video video-2">
    <i class="bx bx-play-circle"></i>
    <i class="bx bx-pause-circle pause"></i>
    <p class="numericalNumber">01.</p>
    <h3 class="title">title title title</h3>
    <p class="time">3:46</p>
</div>
```

<css>(关键部分)

控制高度

```
.video-playlist .videos {
  height: 70%;
  overflow-y: auto;
}
```

控制视频选择项样式

```
.video-playlist .videos .video {
  position: relative;
  width: 100%;
  padding: 1.2rem var(--margin-sm);
  margin-top: 0.8rem;
  cursor: pointer;
  border-radius: var(--border-radius-sm);

  display: flex;
  gap: var(--margin-sm);
  align-items: center;
}
```

悬浮以及图标变化

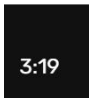
```
.video-playlist .videos .video:last-child {
  margin-right: 0;
}
.video-playlist .videos .video:hover {
  background-color: var(--video-hover);
}
.video-playlist .videos .video:hover .bx {
  color: var(--play-icon-hover);
}
.video-playlist .videos .video.active {
  background-color: var(--video-selected);
  color: var(--video-selected-text);
}
```

<js>

视频名称装填以及高亮选择，此处通过 active 标签进行选择，利用 css 样式改变被选择视频选项样式

```
let fileName = [
  "Under Scope: Liz and Blue Bird Review",
  "Song Birds MV1",
  "Nozomi Autography",
  " An Open letter to Kyoto Animation",
  "Song Birds MV2",
  "the Third Movement",
  "Liz and Blue Bird Preview",
  "Liz and Blue Bird Snippet",
];

liveTitle.innerHTML = fileName[0];

for (i = 0; i < video.length; i++) {
  (function (n) {
    video[n].onclick = function () {
      for (var j = 0; j < video.length; j++) {
        if (video[j].classList.contains("active")) {
          video[j].classList.remove("active");
        }
      }
      console.log("aaa");
      video[n].classList.add("active");
      console.log(videoOnlive);
      let e = n + 1;
      let videoName = "videos/video" + e + ".mp4";
      videoOnlive.src = videoName;
      liveTitle.innerHTML = fileName[n];
    };
  })(i);
}
```

7:41



获取并计算和装填视频时长以及视频序列号，此处使用 video 对象的 durations 属性，获取时长并进行计算转化，

```
const titles = document.querySelectorAll(".video .title");

for (let i = 0; i < fileName.length; i++) {
  titles[i].innerHTML = fileName[i];
}

const durations = document.querySelectorAll(".time");
const listSummary = document.querySelector(".list-summary");
const numericalNumbers = document.querySelectorAll(".numericalNumber");

let timeSum = 0;

for (let i = 1; i <= video.length; i++) {
  let videoName = "videos/video" + i + ".mp4";
  let videoOB = new Audio(videoName);
  numericalNumbers[i - 1].innerHTML = i < 10 ? "0" + i + "." : "" + i + ".";
  videoOB.oncanplay = function () {
    let timeLength = parseInt(videoOB.duration);
    timeSum += timeLength;
    // console.log(timeSum);
    let timeText = "";
    timeText += timeLength / 3600 < 1 ? "" : timeLength / 3600 + ":";
    timeText +=
      (timeLength / 60) % 60 < 1
        ? "0:"
        : parseInt((timeLength / 60) % 60) + ":";
    timeText +=
      timeLength % 60 < 1
        ? "00"
        : timeLength % 60 >= 10
        ? timeLength % 60
        : "0" + (timeLength % 60);
    durations[i - 1].innerHTML = timeText;
  };
}
```

其中 icon 的变化是由 css 进行控制，当 js 添加 active 标签后，可以通过选择器将其是否可视的 visibility 属性进行改变

```
.pause {
  visibility: hidden;
}

.video-playlist .videos .video.active .bx-play-circle {
  visibility: hidden;
}

.video-playlist .videos .video.active .pause {
  visibility: visible;
}

```


装 订 线

```
.video-playlist .videos .video.active .bx-play-circle {
  visibility: hidden;
}
```

3.3 扩展功能实现

3.3.1 倍速功能

倍速功能通过控制 video 的 playbackrate 属性可以进行调节，用 js 接受 button 传递参数，实现视频减慢播放或者加速播放

<html>

```
<div class="speed-control">
    <button class="speed-button sp1">0.5X</button>
    <button class="speed-button sp2 sp-active">1.0X</button>
    <button class="speed-button sp3">1.5X</button>
    <button class="speed-button sp4">2.0X</button>
</div>
```

<css>

```
.speed-control {
  display: flex;
  border-radius: var(--border-radius-sm);
  box-shadow: 0 0 1rem 0.5rem var(--shadow);
}

.speed-control .speed-button {
  height: 50%;
  padding: 0.5rem 1.2rem;
  background-color: var(--background-color);
  border: 1px solid var(--text);
  color: var(--text);
}

.speed-button:hover {
  cursor: pointer;
}

```

```
.speed-button.sp-active {
  background-color: var(--text);
  color: var(--background-color);
}
```

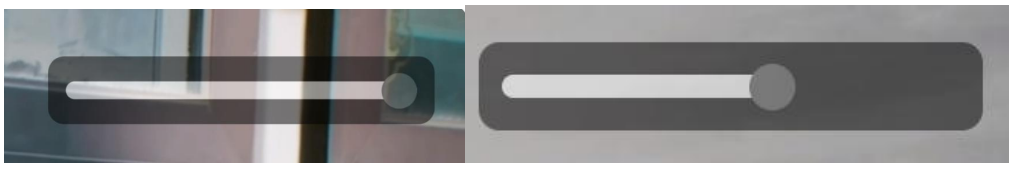
<js>

通过为 button 添加 active 标签，改变按钮样式，改变视频 playbackrate 属性，并利用 button 的特性获取对应值

```
let speed_list = [0.5, 1.0, 1.5, 2.0];

var speed_buttons = document.querySelectorAll(".speed-button");
for (let i = 0; i < speed_buttons.length; i++) {
  speed_buttons[i].addEventListener("click", function () {
    for (let j = 0; j < speed_buttons.length; j++) {
      if (speed_buttons[j].classList.contains("sp-active")) {
        speed_buttons[j].classList.remove("sp-active");
        break;
      }
    }

    videoObj.playbackRate = speed_list[i];
    speed_buttons[i].classList.add("sp-active");
  });
}
```

装 订 线

3.3.2 音量功能



通过 html 中 input 输入标签制作音量滑块，js 获取相应数值后调整视频的 volumn 属性来实现音量控制

<html>

```
<div class="vol-control">
    <input type="range" class="btnsound" value="100" max="100" />
</div>
```

<css>

基本样式

```
.vol-control {
  position: absolute;
  width: 23%;
  height: 5%;
  top: 5%;
  right: 2%;
  padding: 1.2rem 0.6rem;
  border-radius: var(--border-radius-sm);
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: var(--text);
  opacity: 0.5;
  visibility: hidden;
  transition: 0.3s;
}
```

关于 css 控制滑块样式和进度条样式：在控制进度条样式时，通过在 Input 进度条左侧添加伪元素，并通过 js 获取 input 数值对其长度进行调整，保证左侧进度与滑块长度保持相等。由于获取伪元素长度信息并不方便，我们可以使用全局变量进行控制，js 通过控制改变该全局变量对伪元素长度进行修改

```
:root {
  --vol-len: 100%;
}

input[type="range"]::before {
  position: absolute;
  content: "";
  left: 0;
  height: 100%;
  width: var(--vol-len);
  border-radius: 5px;
  background-color: var(--background-color);
  border: none;
}
```

相应 js 修改部分：

```
v = vol_input.value;
document.documentElement.style.setProperty("--vol-len", v + "%");
videoObj.volume = v / 100;
```

〈js〉

为 dom 对象添加鼠标事件检测，保证鼠标点击并松开后对滑块 input 数据进行读取。读取 input 数据后可对视频 volumn 属性进行修改

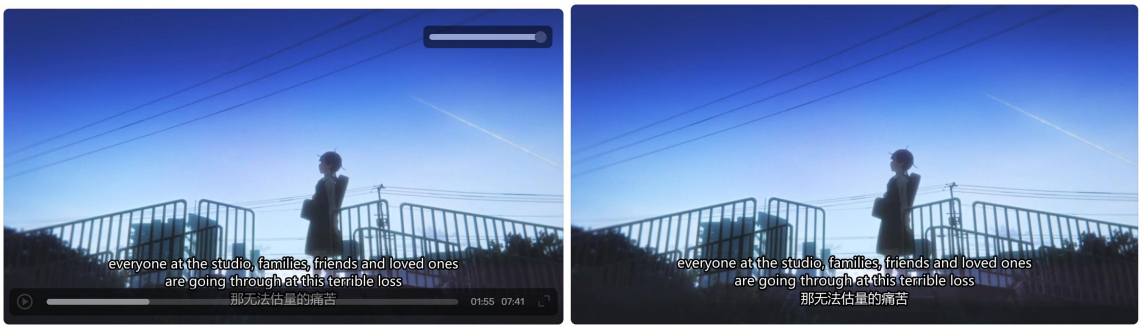
```
var vol_input = document.querySelector(".btnsound");
console.log(vol_input);
var vol_length = vol_input.clientWidth;
var vol_processbar = document.querySelector('.input[type="range"]::before');
var event = window.event;

vol_input.onmousedown = function (event) {
  document.onmousemove = function (event) {
    v = vol_input.value;
    document.documentElement.style.setProperty("--vol-len", v + "%");
    videoObj.volume = v / 100;
  };

  document.onmouseup = function (event) {
    event = event || window.event;
    document.onmousemove = null;
    document.onmouseup = null;
  };
};
```

装

订

线

3.3.3 悬停显示控制坞功能

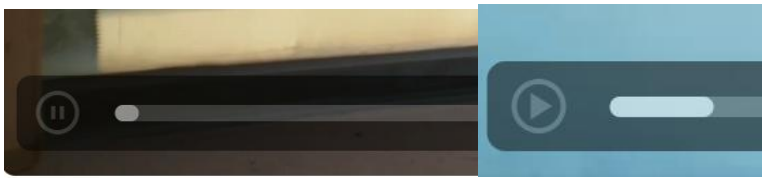


当鼠标悬停在视频上方时，控制条会显示可供选择，否则经过 0.3s 后进度条自动消失，不会影响用户交互体验

〈js〉

```
.video-play:hover .control,
.video-play:hover .vol-control {
  visibility: visible;
  cursor: pointer;
}
```

3.3.4 播放暂停功能



播放暂停主要通过控制视频的 pause()和 play()，并对外以两种方式展示：

点击视频主体进行播放暂停控制

点击左下角小图标进行播放暂停控制

<html>

```
<div class="fa fa-play play_pause">
    <i class="bx bx-play-circle v-start"></i>
    <i class="bx bx-pause-circle v-stop"></i>
</div>
```

<js>

此处需要为这个图标以及视频主体绑定点击事件，

并且通过变量取反进行控制，且考虑到操作的连贯性，此处的需要为同一个变量，

且在暂停播放过程中要注意左下角小图标进行相应变化

为视频绑定事件：

```
let temp = 0;
var v_start = document.querySelector(".v-start");
var v_stop = document.querySelector(".v-stop");
videoObj.addEventListener("click", function () {
  temp = ~temp;
  if (!temp) {
    videoObj.pause();
    v_start.style.display = "block";
    v_stop.style.display = "none";
  } else {
    videoObj.play();
    v_start.style.display = "none";
    v_stop.style.display = "block";
  }
});
```

为图标绑定事件以及修改鼠标样式

```
v_start.addEventListener("click", function () {
  videoObj.play();
  temp = ~temp;
  v_start.style.display = "none";
  v_stop.style.display = "block";
});

v_start.addEventListener("hover", function () {
  v_start.style.cursor = "pointer";
});

v_stop.addEventListener("click", function () {
  videoObj.pause();
  temp = ~temp;
  v_start.style.display = "block";
  v_stop.style.display = "none";
});

v_stop.addEventListener("hover", function () {
  v_stop.style.cursor = "pointer";
});
```

注意:此处为 icon 绑定事件时使用 display 属性对其显示进行控制,display 为 none 时相应事件不可见。可以通过 visibility 属性控制，后者 hidden 时点击事件等仍然存在，利用 display 对事件相应更优

3.3.5 进度条功能

进度条功能与音量控制外观相似，但此处使用 js 控制播放进度的方式来实现进度条自动前进，完成点击跳转、拉动跳转的功能

<html>

```
<div class="control">
    <div class="fa fa-play play_pause">
        <i class="bx bx-play-circle v-start"></i>
        <i class="bx bx-pause-circle v-stop"></i>
    </div>
    <div class="progress">
        <div class="progress-bar"><div class="tips-go"></div></div>
    </div>
    <div class="timer">
```

```
            <span class="progress_timer">00:00</span>/
            <span class="duration_timer">00:00</span>
        </div>
        <div class="fa fa-expand expand">
            <i class="bx bx-expand-alt expander"></i>
        </div>
    </div>
```

<js>

1. 进度条长度变化：

```
// 视频
var videoObj = document.querySelector("video");
var playBtn = document.querySelector(".play_pause");
videoObj.muted = false;
videoObj.autoplay = true;
// 时间
var progressTimer = document.querySelector(".progress_timer");
var durationTimer = document.querySelector(".duration_timer");

let { totalT, presentT } = { totalT: 0, presentT: 0 };
videoObj.addEventListener("canplay", function () {
  totalT = this.duration;
  var videoDuration = formatTime(totalT);
  durationTimer.innerHTML = videoDuration;
});

// 进度条
var progress = document.querySelector(".tips-go");
console.log(progress);
var percent = (presentT / totalT) * 100;
progress.style.width = percent + "%";

videoObj.addEventListener("timeupdate", function () {
  presentT = this.currentTime;
  var videoCurrent = formatTime(presentT);
  progressTimer.innerHTML = videoCurrent;
  var percent = (presentT / totalT) * 100;
  progress.style.width = percent + "%";
});
```

利用 currentTime 获取当前视频播放进度，通过其与总时间的占比确定进度条的宽度。

2. 进度条点击和拖动事件：

```
var main_bar = document.querySelector(".progress-bar");

window.onload = function () {
```


装 订 线

```
    videoObj.play();
    videoObj.autoplay = true;
  };

  main_bar.addEventListener("click", function (e) {
    let perc = e.offsetX / main_bar.clientWidth;
    videoObj.currentTime = parseInt(perc * totalT);
  });

  main_bar.onmousedown = function (event) {
    document.onmousemove = function (event) {
      let perc = event.offsetX / main_bar.clientWidth;
      videoObj.currentTime = parseInt(perc * totalT);
    };

    // 为右侧圆点绑定鼠标抬起事件
    document.onmouseup = function (event) {
      event = event || window.event;
      // 取消鼠标移动事件
      document.onmousemove = null;
      document.onmouseup = null;
    };
  };
```

为进度条添加点击事件，保证进度条能够进行点击跳转。

为进度条添加拖动效果：在鼠标按下事件中添加鼠标移动事件，在移动事件中监听当前鼠标的位置离左侧进度条边界的距离，并与进度条总长度做比给出进度百分比，利用进度百分比确定视频对应百分比的秒数并利用 currentTime 对视频的当前时间进行修改。

3. 视频时间动态显示

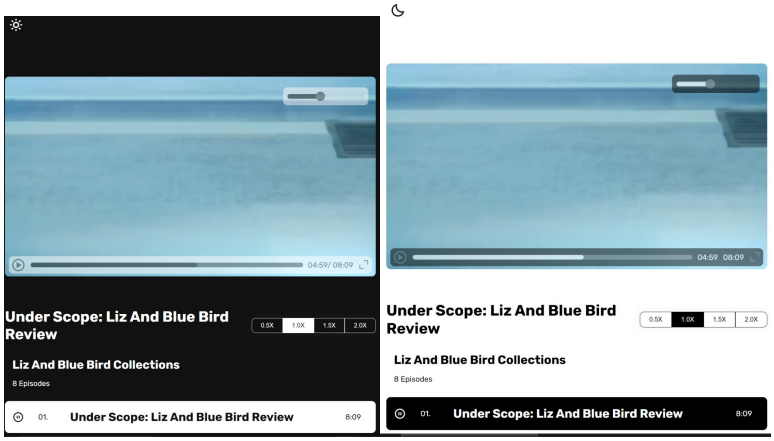
〈js〉

```
const durations = document.querySelectorAll(".time");
const listSummary = document.querySelector(".list-summary");
const numericalNumbers = document.querySelectorAll(".numericalNumber");

let timeSum = 0;

for (let i = 1; i <= video.length; i++) {
  let videoName = "videos/video" + i + ".mp4";
  let videoOB = new Audio(videoName);
  numericalNumbers[i - 1].innerHTML = i < 10 ? "0" + i + "." : "" + i + ".";
  videoOB.oncanplay = function () {
```

```
        let timeLength = parseInt(videoOB.duration);
        timeSum += timeLength;
        // console.log(timeSum);
        let timeText = "";
        timeText += timeLength / 3600 < 1 ? "" : timeLength / 3600 + ":";
        timeText +=
          (timeLength / 60) % 60 < 1
            ? "0:"
            : parseInt((timeLength / 60) % 60) + ":";
        timeText +=
          timeLength % 60 < 1
            ? "00"
            : timeLength % 60 >= 10
            ? timeLength % 60
            : "0" + (timeLength % 60);
        durations[i - 1].innerHTML = timeText;
      };
    }
```

通过获取当前视频播放的 currentTime，进行时间格式转化后显示在进度条右侧

3.3.6 夜间模式实现



夜间模式主要通过为左上角的切换小图标绑定点击事件，js 改变 css 颜色的全局变量进行实现

<css>

```
body.dark {
  --background-color: #111;
  --text: #fff;
  --video-hover: #ccc;
```

```
    --video-selected: #fff;
    --video-selected-text: #000;

    --border-radius-sm: 9px;

    --margin-ssm: 1rem;
    --margin-sm: 2rem;
    --margin-md: 6rem;

    --fontsize-sm: 1.6rem;
    --fontsize-md: 2rem;

    --scrollbar-backgroundcolor: #000;
    --scrollbar-thumbcolor: #eee;
    --play-icon-normal: #555;
    --play-icon-hover: #000;
    --play-icon-selected: #000;
    --shadow: rgb(255, 255, 255, 0.02);
}
```

此处通过 js 为 body 标签添加 dark 属性来对全局颜色进行统一控制

<js>

```
const darkMode = document.querySelectorAll(".icon-change");
const bodyTag = document.getElementsByTagName("body");

(function () {
  for (let i = 0; i < 2; i++) {
    darkMode[i].onclick = function () {
      if (!bodyTag[0].classList.contains("dark")) {
        bodyTag[0].classList.add("dark");
      } else {
        bodyTag[0].classList.remove("dark");
      }
    };
  }
})();
```

小图标的显示为 css 进行控制，当 body 标签具有 dark 属性时，小图标的 visibility 属性发生相应变化

```
.icon-change {
  position: absolute;
  top: 1.5rem;
  right: 1.5rem;
}

.icon-change:hover {
  cursor: pointer;
}
```

```
.icon-change .bx {
  font-size: var(--fontsize-md);
  color: var(--text);
}

body .sun {
  visibility: hidden;
}
body .moon {
  visibility: visible;
}

body.dark .moon {
  visibility: hidden;
}
body.dark .sun {
  visibility: visible;
}
```

装 订 线

3.3.7 全屏功能

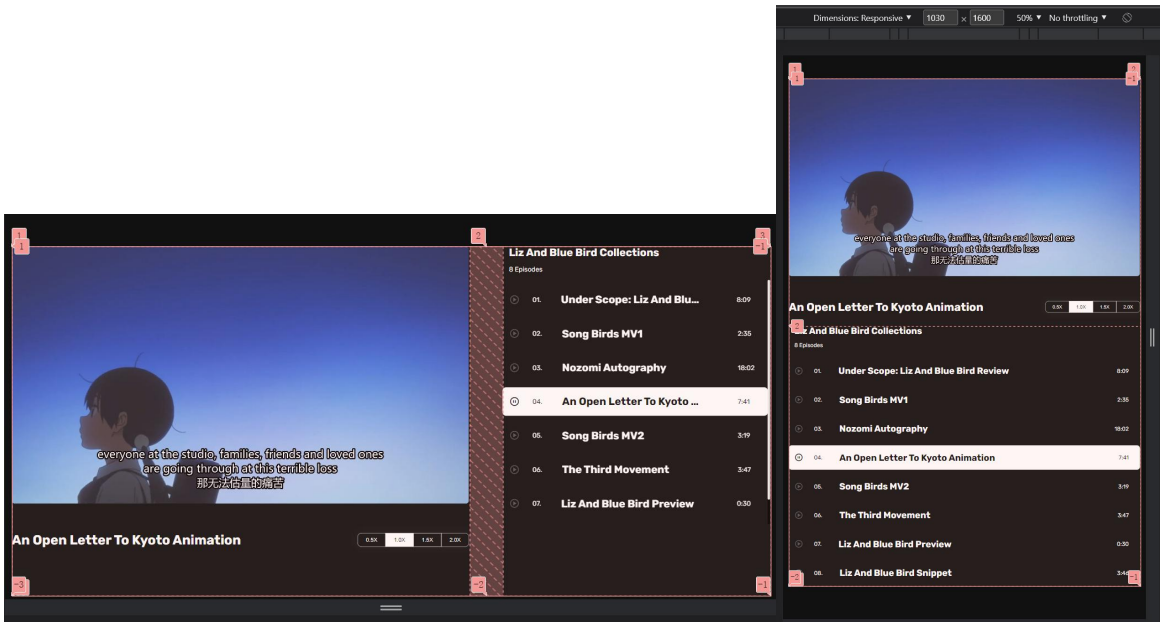
全屏功能由 web 组件视频对象属性 webkitEqquestFullScreen()控制

<js>

```
var expand = document.querySelector(".expand");
expand.addEventListener("click", function () {
  videoObj.webkitRequestFullScreen();
});
```

3.3.8 响应式设计

2560*1600 屏幕显示、1030*1600 屏幕显示



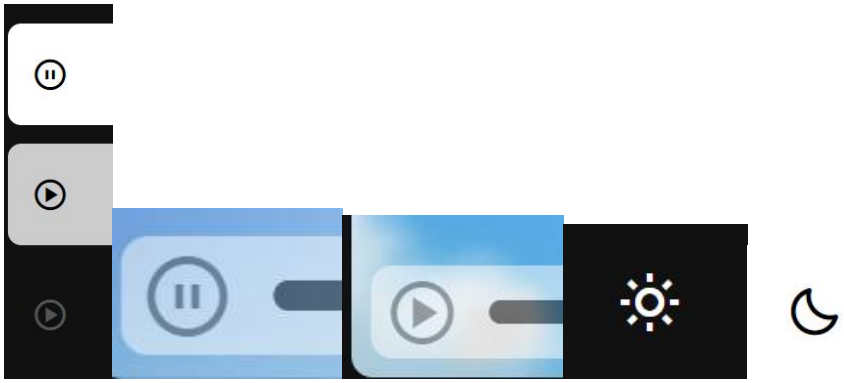
响应式设计通过 css 中的媒体请求进行变化

```
@media screen and (max-width: 1590px) {
  .container {
    grid-template-columns: 1fr;
    /* grid-template-rows: 5fr 1fr; */
  }

  .icon-change {
    position: absolute;
    top: 1.5rem;
    left: 1.5rem;
  }

  .video-playlist .videos {
    height: 100%;
    overflow-y: visible;
  }

  .video h3 {
    width: 100%;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
  }
```

```
24.  }
```

3.4 细节调整

3.4.1 图标变换



图标变化主要体现在视频左下角暂停/播放控制、左上角夜间模式切换设计、播放列表左侧小标识设计。此处展示的是视频列表中图标变换设计。

```
1.  let temp = 0;
2.  var v_start = document.querySelector(".v-start");
3.  var v_stop = document.querySelector(".v-stop");
4.  videoObj.addEventListener("click", function () {
5.    temp = ~temp;
6.    if (!temp) {
7.      videoObj.pause();
8.      v_start.style.display = "block";
9.      v_stop.style.display = "none";
10.   } else {
11.     videoObj.play();
12.     v_start.style.display = "none";
13.     v_stop.style.display = "block";
14.   }
15. });
```

通过 js 控制其显示与否，主要控制 display 和 visibility 属性控制显示。前者设置 none 则该图标不存在于网页流中，无法被页面检测到。后者设置 hidden 则该图标不显示但其仍然存在于网页中，仍可以实现交互行为等。

4. 心得体会

在完成该实验的过程中，我进一步了解了 JavaScript 脚本语言的重要性和使用方法，加深了我对 HTML 和 CSS 的理解。

实现基本要求后，我按照扩展要求进一步优化了媒体播放器，添加了倍速、音量控制、全屏、进度条功能以及播放列表等功能。其中还添加了浮动显示、夜间模式和响应式设计，完善了用户体验