同济大学

TONGJI UNIVERSITY

《WEB 技术》

实验报告

实验名称	实验 3 播放器实现
小组成员	郑博远（2154312）
学院（系）	电子与信息工程学院计算机科学与技术系
专　　业	计算机科学与技术
任课教师	郭玉臣
日　　期	2023 年 4 月 9 日

一、实验原理

本次实验使用 HTML、CSS、JavaScript 实现一个媒体播放器，能够播放预定的音视频，实现开始、暂停、倍速、音量控制、全屏、进度条的功能，并能够在播放列表中选取媒体播放。本实验的实验原理如下：

1.1 HTML5 <video>标签

HTML5 中提供了 <video> 标签，可以将视频（或音频）嵌入网页中。 <video> 标签有很多属性，如 src、controls、autoplay、loop 等等。其中，src 属性用于指定视频文件的 URL，controls 属性用于在页面中显示默认的控制条，autoplay 属性用于指定视频自动播放，loop 属性用于指定循环播放。本次实验中也通过了设置这些属性，从而方便地控制视频的播放。

此外，<video> 标签还可以使用 <source> 标签来指定多个视频文件，以便在不同的浏览器中播放不同的格式。如果浏览器不支持 <video> 标签指定的视频格式，就会去找 <source> 标签指定的其他格式，直到寻找到可播放的音视频源为止。

1.2 CSS 添加样式和布局

CSS 可以为媒体播放器添加样式和布局。通过为 <video> 标签设置样式和布局，可以美化媒体播放器，并使其更符合网页的整体风格。

在本次实验中，我使用 CSS 来设置 <video> 标签的宽度、高度、边框、背景色等属性，以及设置控制条的样式、位置、颜色等属性。此外，还可以通过 CSS 实现响应式布局，使播放器在不同的设备上都能够正确显示。

1.3 JavaScript 处理交互与控制

JavaScript 是一种高级的、解释型的脚本语言，被广泛应用于 web 前端开发中。通过 JavaScript，我们可以实现对网页中元素的动态操作，以及对用户输入事件的响应。在本实验中将使用 JavaScript 实现一个媒体播放器，并掌握 JavaScript 基础语法和应用程序设计的过程。JavaScript 控制媒体播放器的功能，如播放、暂停、

倍速、音量、全屏和进度条等。通过 DOM 操作来获取媒体元素，并通过调用它们的方法和属性来实现播放器的功能。具体来说可以通过 JavaScript 实现如下功能：

· 播放和暂停功能：通过获取视频元素，在点击播放按钮或视频本体时调用 play() 和 pause() 方法来实现播放和暂停；

· 倍速功能：通过获取视频元素，设置 playbackRate 属性来实现调整播放速度。此外，倍速选择列表上的每个倍速选择选项都能够触发 JavaScript 中的函数，从而实现倍速选择；

· 音量控制：通过获取视频元素，设置 volume 属性来实现调整音量。通过监听鼠标的按下和拖拽事件来实现音量条的拖动控制功能。

· 全屏功能：通过获取视频元素，调用 requestFullScreen() 方法来实现全屏。

· 进度条功能：通过获取视频元素的 currentTime 和 duration 属性来计算视频的当前进度和总时长，并在进度条上显示当前进度。同时通过监听鼠标的按下和拖拽事件来实现进度条的拖动控制功能。

装

订

线

二、实验步骤

在本次实验中，我独立地从零开始搭建了一个功能较为齐全的音视频播放器。过程主要分为 HTML、CSS 的设计以及 JavaScript 的编写。

2.1 HTML 设计

1. 创建一个 HTML 文档，并设置文档类型为 <!DOCTYPE html>。这里的 <!DOCTYPE html> 是 HTML5 文档类型声明，指定了当前文档的 HTML 版本为 HTML5。在 HTML5 中，文档类型声明只需要写这一个声明即可；

2. 在 HTML 头部设置如编码格式，文档标题等信息。将字符集设置为“UTF-8”，并在头部中添加一个标题，将标题设置为“视频播放器”，此外还需要链接外部 CSS 文件“player.css”；

3. 在 HTML 主体中创建一个 ID 为“video-player”的 元素，作为整个视频播放器的容器。通过一个具有唯一 ID 的 元素，将整个视频播放器包含在一个容器中，便于后续的样式设置和 JavaScript 操作；

4. 在“video-player”容器内部创建一个 ID 为“video-container”的 <figure> 元素。<figure> 元素表示独立的内容块，通常用于嵌入多媒体内容。在这个播放器中，将整个视频播放器的内容放在一个 <figure> 元素中，以便于后续布局和样式设置；

5. 在“video-container”内部创建一个 ID 为“video-header-bar”的 元素。这个 元素是视频播放器的头部栏，用于显示视频的标题等信息。将这个元素设置为 <figure> 元素的子元素。在“video-header-bar”内部创建一个 ID 为“video-name”的 元素。这个 元素用于显示视频的标题；

6. 继续在“video-container”内部创建 ID 为“video”的 <video> 元素。这个 <video> 元素是视频播放器的主体部分。在“video”元素内部创建一个 ID 为“source”的 <source> 元素，用于指定视频文件的地址和格式（src 部分可以通过 JavaScript 更改从而实现媒体源的切换）。将文件地址设置为视频文件的 URL，格式设置为“video/mp4”，这样浏览器就可以根据视频文件的格式进行解析和播放；

7. 同样作为“video-container”的子元素，与“video”并列地创建一个 ID 为“video-control-bar”的 元素。这个元素用于包含视频播放器的控制面板，例如播放/暂停按钮、音量控制条、倍速、设置等。所有的相应按钮与悬浮弹出菜单项都是“video-control-bar”这个 的子元素；

8. 下面在“video-control-bar”中创建控制栏部分的各个子元素。首先创建一个 ID 为“progress-wrap”的 ，其中包含了进度条的相关部分。该 包含两个部分，其中第一个 <progress> 元素是 HTML5 提供的标签，通过 JavaScript 控制来实现进度的实时更新；其二是一个 ID 为“progress-dot”的 ，表示进度条上的白色圆点。二者的布局设置将在 CSS 部分进行说明；

9. 其次，在“video-control-bar”内部创建一个 ID 为“video-control-buttons”的 ，用来包含包括播放/暂停、音量、设置、画中画、全屏等按钮。“video-controls-buttons”中包含 ID 为“video-controls-buttons-left”和

“video-controls-buttons-right”的两个 ，分别包含左侧和右侧的按钮；

10. 在“video-controls-button-left”中创建两个部分。其中第一个元素是播放/暂停按钮的 <img>，由 JavaScript 控制不同状态下的图标；第二个 中含三个 ，用于显示视频的播放时间信息，也由 JavaScript 实时更新；

11. 在“video-control-buttons-right”这个 中创建一个 ID 为“multiple-speed”的 ，用于显示倍速按钮和相应的选项栏。在其中建立一个 用于在控制栏中显示“倍速”二字的按钮；与之并列地创建一个 ，包含多个 <li> 对应从 0.5 倍速到 2.0 倍速的不同倍速选项；

12. 继续在“video-control-buttons-right”这个 中创建一个 ID 为“volume”的 ，用于显示音量按钮和拖动条。首先在其中创建一个表示音量图标的 <img>， 然后创建一个 ID 为“volume-bar-holder”的 ，其包含了音量拖动条部分的组件；

13. 在“volume-bar-holder”中创建一个 ID 为“volume-value”的 用于显示音量数值，然后再创建一个 ID 为“volume-bar”的 表示音量条。再在“volume-bar”这一 中创建一个 ID 为“volume-bar-done”的 用于显示当前音量覆盖的音量条部分，并在其中创建一个 ID 为“volume-dot”的 表示音量条上方的圆点，通过 CSS 的相应布局使其保持在“volume-bar-done”的最上方；

14. 继续在“video-control-buttons-right”中创建一个 ID 为“settings-block”的 ，包含设置内容（循环播放和镜像翻转）。以循环播放为例，在“settings-block”内部创建一个 表示这一部分的内容，建立一个 以在其左侧显示解释说明文字，同时建立一个 class 为“switch-wrapper”的 来表示按钮。按钮部分通过 <input> 与 <label>来实现左右移动的 switch 效果；

15. 在“video-control-buttons-right”中创建两个<img>对应画中画以及全屏按钮；

16. 在“video-Splayer”中与“video-container”并列建立一个 ID 为“video-chooser”的 ，这一部分代表着右侧的媒体源选择栏。由于使用原生 JS，此部分的内容都由 JavaScript 通过 innerHtml 进行填充。

2.2 CSS 设计

CSS 部分的编写与 HTML 的设计并非割裂的前后关系，而是共同进行的，因此难以列举具体的实验步骤。下面主要介绍 CSS 设计中比较重要的部分：

2.2.1 视频进度条颜色设置

使用 CSS 实现如下图效果所示的视频进度条：

其中进度条圆点的部分是一个单独的 ，其位置由 JavaScript 不断计算并调整 margin-left 从而动态确定，将在下一部分具体说明。此处说明如何通过 CSS 实现进度条样式的设置。进度条的部分是使用 HTML5 中的 <progress> 标签实现的，在此基础上要通过 CSS 改变的样式有：进度条始末的圆角、已经播放部分的颜色、未播放部分的背景颜色三个部分。

通过设置进度条的“border-radius”属性可以实现始末的圆角样式。需要注意的是，还要添加“overflow:hidden”使得超出的部分隐藏，否则无法正确显示。

关于 <progress> 已完成部分与总长度背景色颜色的设置，不同的浏览器有不同的 CSS 写法，为实现兼容性，要在 CSS 中尽可能详细的进行涵盖。比如在 Chrome 浏览器下，已完成部分的颜色与背景色分别通过“#progress::-webkit-progress-bar”和“#progress::-webkit-progress-value”的“background-color”来实现；在 IE10 以上，则通过“#progress”中的“color”与“background”来实现。为兼容性而书写的 CSS 代码具体如下：

```
#progress {
    height: 6px;
    width: 100%;
    border-radius: 5px;
    overflow: hidden;
    flex: 0 0 auto;
    cursor: pointer;
```

```
    /* 兼容 IE 已完成进度背景色 */
    color: mediumaquamarine;
    /* 兼容 IE FireFox 总长度背景色 */
    background: lightgray;
}

/* Chrome 表示总长度背景色 */
#progress::-webkit-progress-bar {
    background-color: lightgray;
}

/* Chrome 表示已完成进度背景色 */
#progress::-webkit-progress-value {
    background-color: mediumaquamarine;
}

/* 兼容 FireFox 表示已完成进度背景色 */
#progress::-moz-progress-bar {
    background-color: mediumaquamarine
}
```

2.2.2 音量条的实现

使用 CSS 实现如下图效果的鼠标悬停时出现的音量条：

“volume-bar-holder”作为“volume”（音量按钮）的子元素，默认设置为“visibility: hidden;”不可见。设置“#volume:hover”的伪类，使得鼠标悬停在音量按钮上（包括音量条本身，因为是 volume 的子元素）时音量条显示。

在音量条的实现上，通过“volume-bar-done”对应的 表示当前音量覆盖的进度条，“volume-dot”对应的 表示上方的白色圆点。将二者的“position”

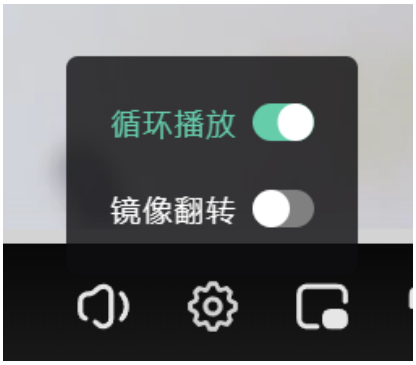
属性设置为“absolute”，并设置前者为“bottom:0”从而实现已覆盖的部分居于底部；白色圆点的 设置“left:50%; top:0”实现其在已覆盖部分的顶部并居中。此外，由于 left 属性定位的是元素的左上角，还需要通过“transform: translate(-50%, -50%);”来修正白色圆点的位置。

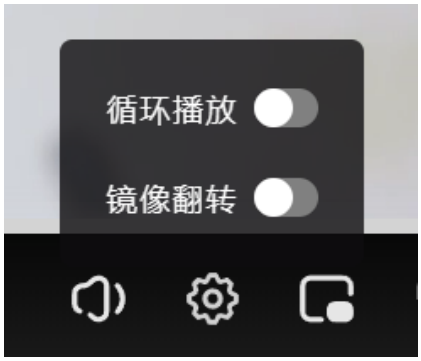
```
#volume{
    position: relative;
    display: flex;
    flex-direction: column;
}

#volume:hover #volume-bar-holder{
    visibility: visible;
}

#volume-bar-holder{
    position: absolute;
    display: flex;
    flex-direction: column;
    justify-content: space-evenly;
    align-items: center;
    left: 50%;
    bottom: 25px;
    width: 40px;
    height: 120px;
    border-radius: 5px;
    transform:translate(-50%, 0);

    background-color: rgba(14, 12, 15,
0.8);

    visibility: hidden;
}

#volume-value{
    color: white;
    font-size: 10px;
    font-family:'Century Gothic';
}
```

```
#volume-bar{
    position: relative;
    height: 70px;
    width: 2px;
    background-color: #ffffff;
    cursor: pointer;
}


#volume-bar-done{
    position: absolute;
    height: 70px;
    width: 2px;
    background-color:
mediumaquamarine;
    bottom: 0;
}


#volume-dot{
    position: absolute;
    width: 10px;
    height: 10px;
    border-radius: 100%;
    background-color: #fff;
    top: 0;
    left: 50%;
    transform: translate(-50%,
-50%);
}
```

2.2.3 开关按钮的实现

使用 CSS 实现如下图的带动画效果的开关按钮：





简单的开关按钮可以通过 HTML 中的 <input> 标签来实现“勾选/未勾选”的功能。但是可视化的原点移动等则需要自行实现。通过设置 label 的各种属性作为底部的灰色背景，又通过“::before”伪类的设计来作为上方白色的可移动圆形。

动画方面，通过“input:checked”设置伪类来实现选中时的按钮背景变色和白色圆点移动时的动画效果。通过“transition”属性可以设置动画的变化时间等细节，“cubic-bezier”能设置动画曲线使得移动变化更加丝滑。

```
.switch-wrapper input{
    display: none;
}

.switch-wrapper label{
    position: relative;
    background: grey;
    cursor: pointer;
    width: 25px;
    height: 15px;
    border-radius: 100px;
    display: block;
}

.switch-wrapper label::before {
    content:'';
    position: absolute;
    width: 15px;
    height: 15px;
    border-radius: 100%;
    background:white;
    transition: all 0.36s;
    top: 0px;
    left: 0px;
}

.switch-wrapper input:checked + label
{
    background: mediumaquamarine;
    transition: all 0.36s
cubic-bezier(.78, .14, .15, .86);
}


.switch-wrapper input:checked +
label::before {
    left: 100%;
    transform: translateX(-100%);
    transition: all 0.36s
cubic-bezier(.78, .14, .15, .86);
}
```

2.3 JavaScript 编写

2.3.1 播放与暂停

首先，我在实现播放/暂停功能时使用了 HTML5 Video 元素的 play() 和 pause() 方法来实现，这些方法可以很方便地控制视频的播放和暂停。在实现时，我监听了一个按钮的点击事件，并通过 JavaScipt 控制按钮的图标来表示当前视频的状态（播放或暂停）以达到更好的交互效果。

2.3.2 进度条拖动与更新

在实现进度条的同步更新和拖动时，我使用了 HTML5 Video 元素的 currentTime 属性，该属性表示视频当前播放的时间（以秒为单位）。我监听了进度条的 mouseDown 事件，同时监听鼠标的拖动，在 mouseUp 之前将鼠标在屏幕的位置转换为进度条长度以及视频的播放时间，并通过设置 currentTime 属性来实现视频播放位置的更新。同时，我还监听了视频的 timeupdate 事件，该事件会在视频播放位置改变时触发，通过该事件可以实现进度条的同步更新。

2.3.3 音量条的拖动与更新

在实现音量条的拖动时，我使用了 HTML5 Video 元素的 volume 属性，该属性表示视频的音量大小（范围从 0 到 1）。与进度条的拖动类似，我监听了音量条的 mouseDown 事件，将鼠标的位置转换为音量条的值和视频的音量大小，并通过设置 volume 属性来实现视频音量的调节。此外，还设置了点击音量按钮调用 mute() 方法将视频静音，并根据状态切换音量按钮的图标。

2.3.4 全屏与画中画

在实现全屏时，我使用了 HTML5 Video 元素的 requestFullscreen() 和 exitFullscreen() 方法，这些方法可以控制视频的全屏显示和退出全屏。同时，我还使用了 HTML5 Video 元素的 requestPictureInPicture() 方法，该方法可以将视频显示在画中画模式下。在实现时，我监听了两个按钮的点击事件，并根据当前视频的状态（全屏、非全屏、画中画）来判断采取何种操作。

2.3.5 循环播放与镜像翻转

当监听到循环播放的 <input> 为checked时，可以通过JavaScript将video的loop属性设置为 true，则能够实现视频播放到末尾后的自动播放。另外，监听镜像翻转按钮对应 <input> 是否 checked，通过 JavaScript 设置 CSS3 的 transform 属性来实现视频的水平镜像翻转，以满足用户的需求。

2.3.6 倍速选择

在实现倍速设置时，我使用了 HTML5 Video 元素的 playbackRate 属性，该属性表示视频的播放速度（默认值为 1）。我监听了一个下拉菜单的改变事件，将下拉菜单的值转换为视频的播放速度，并通过设置 playbackRate 属性来实现视频播放速度的调节。此外，还通过 JavaScript 对当前选中的倍速进行高亮显示。

2.3.7 媒体源切换

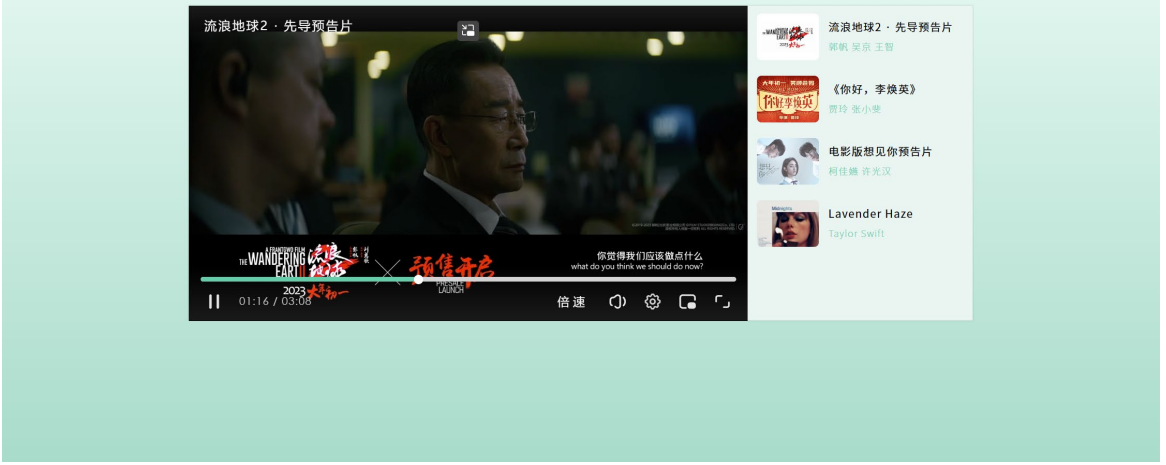
在实现视频切换列表时，我使用 JavaScript 将存储有视频 URL、缩略图、视频名称、视频创作者的信息的常量列表填充了“video-chooser”这个 。对于每一个视频对应的 ，我都添加了监听事件，即在它们被点击时会触发对视频源的替换（其中音频还要在 <video> 区域显示封面图），供选择的视频列表视频顺序也会发生变化，让正在播放的视频居于最高位。

三、实验内容

3.1 效果展示

当鼠标悬停在视频上方时，能够显示控制栏与上方的标题栏如下：

点击播放按钮时视频能够播放并切换图标状态，进度条与下方的时间显示自动随视频进度而更新。鼠标点击进度条后可以拖动（不仅仅是点击）时间轴实现视频时间点切换，如下图所示：



鼠标悬停在“倍速”按钮上时会弹出倍速选项悬浮窗，可以选择不同的倍速比例。当前选中的倍速选项会高亮显示，鼠标悬停的选项也有对应高亮：

鼠标悬停在音量按钮上可以弹出音量调整悬浮窗。音量悬浮窗上方会显示当前音量的数字，下方的音量条也可视化的表明当前的音量大小。可以上下拖动音量条来改变当前音量，也可以点击音量键进行静音：

鼠标悬停在设置按钮上可以弹出设置悬浮窗。悬浮窗中分为“循环播放”以及“镜像翻转”两个选项。点击对应的按钮可以实现相应效果。如下图中展示了点击“镜像翻转”按钮后视频呈镜像播放效果。

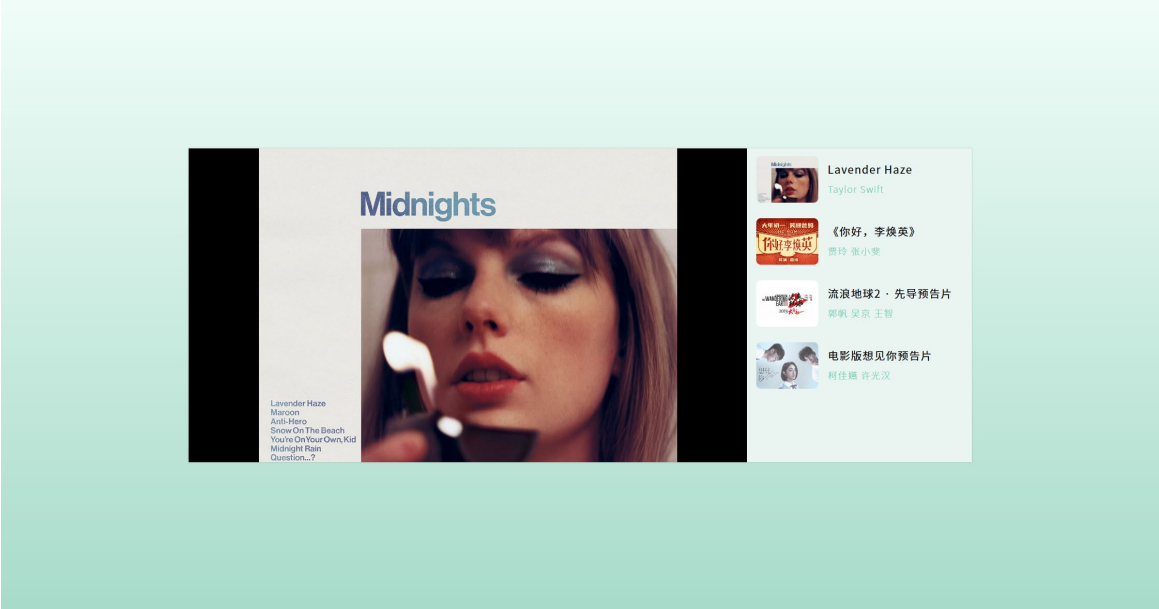
装 订 线

点击画中画按钮可以实现画中画模式：

点击全屏按钮可以进入全屏模式，再次点击可以退出：

装 订 线

点击右侧媒体源列表可以切换当前播放的媒体。当当前播放的媒体是音频时，将在视频窗口部分显示视频的封面图。右侧的视频列表通过 JavaScript 装填，动态调整以保证第一个是当前正在播放的媒体源。



3.2 HTML 源代码

```
<!DOCTYPE html>

<html>
    <head>
        <meta charset="utf-8">
        <title>视频播放器</title>
        <link rel="stylesheet" href="player.css">
    </head>

    <body>
        <div id="video-player">
            <figure id="video-container">
                <div id="video-header-bar">
                    <span id="video-name">《你好·李焕英》预告片</span>
                </div>

                <video id="video">
                    <source src="video1.mp4" type="video/mp4">
                </video>
```

```
                <div id="video-control-bar">
                    <div id="progress-wrap">
                        <progress id="progress" value = "0" min = "0"></pro
gress>
                        <div id="progress-dot"></div>
                    </div>
                    <div id="video-control-buttons">
                        <div id="video-control-buttons-left">
                            <img src="./icon/play.svg" id="playpause" class
="button-img">
                            <div class="time">
                                <span id="current-time">00:00</span>
                                <span>/</span>
                                <span id="total-time">00:00</span>
                            </div>
                        </div>
                        <div id="video-control-buttons-right">
                            <div id="multiple-speed">
                                <span id="multiple-speed-text">倍速</span>
                                <ul id="multiple-speed-choices">
                                    <li class="multiple-speed-choice">2.0x<
/li>
                                    <li class="multiple-speed-choice">1.75x
</li>
                                    <li class="multiple-speed-choice">1.25x
</li>
                                    <li class="multiple-speed-choice">1.0x<
/li>
                                    <li class="multiple-speed-choice">0.75x
</li>
                                    <li class="multiple-speed-choice">0.5x<
/li>
                                </ul>
                            </div>
                            <div id="volume">
                                <img src="./icon/volume.svg" id="mute" clas
s="button-img">
                                <div id="volume-bar-holder">
                                    <div id="volume-value">
                                        100
                                    </div>
                                    <div id="volume-bar">
                                        <div id="volume-bar-done">
                                            <div id="volume-dot"></div>
```

装 订 线

```
                                    </div>
                                </div>
                            </div>
                        </div>
                        <div id="settings-block">
                            <img src="./icon/settings.svg" id="settings
" class="button-img">
                            <div id="settings-holder">
                                <div class="setting-choice">
                                    <span class="prompt">循环播放</span>
                                    <div class='switch-wrapper'>
                                        <input type="checkbox" id="cycl
e" onclick="openVideoCycle(this)">
                                        <label for="cycle"></label>
                                    </div>

                                </div>
                                <div class="setting-choice">
                                    <span class="prompt">镜像翻转</span>
                                    <div class='switch-wrapper'>
                                        <input type="checkbox" id="mirr
or" onclick="openVideoMirror(this)">
                                        <label for="mirror"></label>
                                    </div>
                                </div>
                            </div>
                        </div>
                        <img src="./icon/picinpic.svg" id="picinpic" cl
ass="button-img">
                        <img src="./icon/fullscreen.svg" id="fs" class=
"button-img">
                    </div>
                </div>
            </div>
        </figure>
        <div id ="video-chooser">

        </div>
    </div>

    <script type="text/javascript" src="player.js"></script>
</body>
</html>
```

3.3 CSS 源代码

```
html{
    height: 100%;
}

body{
    width: 100%;
    height: 100%;
    margin: auto;
    display: flex;
    justify-content: center;
    align-items: center;
    background-image: linear-gradient(0deg, rgb(168, 219, 202), rgb(240, 253,
249));
}


#video-player{
    display: flex;
    justify-content: space-around;
    align-items: center;
    position: relative;
    height: 405px;
    box-shadow: 0 0 2px rgba(0, 0, 0, 0.2);
}

figure {
    position: relative;
    display: flex;
    height: 405px;
    width: 720px;
    margin: auto;
    background: no-repeat;
    background-size: contain;
    background-color: black;
    background-position: center;
}

video {
    display: block;
    margin: auto;
    width: 100%;
    user-select: none;
```

```
}

/* 视频选择栏 */
#video-chooser {
    height: 100%;
    width: 290px;
    overflow-y: auto;
    background-color: rgb(234, 245, 242);
    user-select: none;
}

.video-alternative{
    height: 80px;
    width: 100%;
    display: flex;
    justify-content: space-evenly;
    align-items: center;
    cursor: pointer;
}

.video-alternative:hover .title{
    font-weight: bolder;
}

.video-alternative:hover .subtitle{
    font-weight: normal;
}


.video-alternative img{
    width: 80px;
    border-radius: 6px;
}

.video-alternative .title{
    font-size: 14px;
    letter-spacing: 1px;
    margin: 0;
    font-family: '思源黑体';
    color: black;
}

.video-alternative .subtitle{
    font-size: 10px;
    letter-spacing: 1px;
```

装 订 线

```
    margin: 0;
    font-family: '思源黑体';
    font-weight: lighter;
    color: mediumaquamarine;
}

.video-alternative .detail{
    width: 60%;
    height: 70%;
    display: flex;
    flex-direction: column;
    justify-content: space-evenly;
    align-items: start;
}

/* 进度条 */
#progress-wrap {
    padding-left: 15px;
    padding-right: 15px;
}

#progress {
    height: 6px;
    width: 100%;
    border-radius: 5px;
    overflow: hidden;
    flex: 0 0 auto;
    cursor: pointer;
    /* 兼容 IE 已完成进度背景色 */
    color: mediumaquamarine;
    /* 兼容 IE FireFox 总长度背景色 */
    background: lightgray;
}

/* Chrome 表示总长度背景色 */
#progress::-webkit-progress-bar {
    background-color: lightgray;
}

/* Chrome 表示已完成进度背景色 */
#progress::-webkit-progress-value {
    background-color: mediumaquamarine;
}

/* 兼容 FireFox 表示已完成进度背景色 */
```

装 订 线

```
#progress::-moz-progress-bar {
    background-color: mediumaquamarine
}


/* 进度条圆点 */
#progress-dot {
    margin-left: -6px;
    top: 11px;
    width: 12px;
    height: 12px;
    background: white;
    box-shadow: 0 0 2px 0 rgba(0, 0, 0, 0.2);
    border-radius: 50%;
    position: absolute;
    cursor: pointer;
}


/* 按钮图标 */
.button-img {
    width: 25px;
    padding-left: 10px;
    padding-right: 10px;
    cursor: pointer;
}

/* 时间显示 */
.time{
    padding-left: 10px;
    color:lightgray;
    font-size: 14px;
    letter-spacing: 0.75px;
    font-family:'Century Gothic';
}


#video-container:hover #video-header-bar{
    visibility: visible;
}

/* 顶部视频名栏 */
#video-header-bar {
    height: 50px;
    width: 100%;
```

装 订 线

```
    position: absolute;

    left: 0;
    top: 0;
    user-select: none;
    background-image: linear-gradient(0deg, rgba(0, 0, 0, 0), rgba(255, 255,
255, 0.1));

    display: flex;
    flex-flow: column nowrap;
    align-items: stretch;
    justify-content: center;

    visibility: hidden;
}

#video-header-bar span{
    padding-left: 20px;
    color: white;
    letter-spacing: 2px;
}

#video-container:hover #video-control-bar{
    visibility: visible;
}

/* 下端控制进度条 */
#video-control-bar {
    height: 70px;
    width: 100%;

    position: absolute;

    left: 0;
    bottom: 0;
    user-select: none;
    background-image: linear-gradient(0deg, rgba(255, 255, 255, 0.1),
rgba(0,0,0,0));

    display: flex;
    flex-flow: column nowrap;
    align-items: stretch;

    visibility: hidden;
```

```
}

#video-control-buttons {
    flex: 1 1 100%;
    padding-left: 10px;
    padding-right: 10px;
    display: flex;
    flex-flow: row nowrap;
    align-items: center;
    justify-content: space-between;
}

#video-control-buttons-left {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

#video-control-buttons-right{
    display: flex;
    justify-content: space-between;
    align-items: center;
}

#multiple-speed{
    color: white;
    margin-left: 15px;
    margin-right: 15px;
}

#volume{
    position: relative;
    display: flex;
    flex-direction: column;
}

#volume:hover #volume-bar-holder{
    visibility: visible;
}

#volume-bar-holder{
    position: absolute;
    display: flex;
    flex-direction: column;
    justify-content: space-evenly;
```

```
    align-items: center;

    left: 50%;
    bottom: 25px;

    width: 40px;
    height: 120px;
    border-radius: 5px;
    transform:translate(-50%, 0);

    background-color: rgba(14, 12, 15, 0.8);

    visibility: hidden;
}

#volume-value{
    color: white;
    font-size: 10px;
    font-family:'Century Gothic';
}

#volume-bar{
    position: relative;
    height: 70px;
    width: 2px;
    background-color: #ffffff;
    cursor: pointer;
}

#volume-bar-done{
    position: absolute;
    height: 70px;
    width: 2px;
    background-color: mediumaquamarine;
    bottom: 0;
}

#volume-dot{
    position: absolute;
    width: 10px;
    height: 10px;
    border-radius: 100%;
    background-color: #fff;
    top: 0;
    left: 50%;
```

装 订 线

```
    transform: translate(-50%, -50%);
}


#multiple-speed{
    position: relative;
    display: flex;
    flex-direction: column;
    font-size: 15px;
    cursor: pointer;
}

#multiple-speed:hover #multiple-speed-choices{
    visibility: visible;
}

#multiple-speed-choices{
    position: absolute;
    display: flex;
    flex-direction: column;
    justify-content: space-evenly;
    align-items: center;

    left: 50%;
    bottom: 7px;

    width: 60px;
    height: 180px;
    padding: 0px;
    border-radius: 8px;
    transform:translate(-50%, 0);

    background-color: rgba(14, 12, 15, 0.8);
    overflow:hidden;
    visibility: hidden;
}

#multiple-speed-text{
    font-size: 15px;
    letter-spacing: 5px;
}

.multiple-speed-choice{
    list-style: none;
    width: 100%;
```

```
    padding-top: 5px;
    padding-bottom: 5px;
    font-size: 13px;
    font-family:'Century Gothic';
    letter-spacing: 0.75px;
    text-align: center;
}

.multiple-speed-choice:hover{
    background-color: rgba(255, 255, 255, 0.3)
}

#settings-block{
    position: relative;
    display: flex;
    flex-direction: column;
}

#settings-block:hover #settings-holder{
    visibility: visible;
}

#settings-holder{
    position: absolute;
    display: flex;
    flex-direction: column;
    justify-content: space-evenly;
    align-items: center;
    left: 50%;
    bottom: 25px;

    width: 120px;
    height: 90px;
    border-radius: 5px;
    transform:translate(-50%, 0);

    background-color: rgba(14, 12, 15, 0.8);

    visibility: hidden;
}

.setting-choice{
    display: flex;
    width: 80%;
    flex-direction: row;
```

```
    align-items: center;
    justify-content: space-evenly;
    cursor: pointer;
}

.prompt{
    color: white;
    font-size: 12px;
    letter-spacing: 1px;
}

.setting-choice:hover .prompt{
    color: mediumaquamarine;
}


.switch-wrapper input{
    display: none;
}

.switch-wrapper label{
    position: relative;
    background: grey;
    cursor: pointer;
    width: 25px;
    height: 15px;
    border-radius: 100px;
    display: block;
}

.switch-wrapper label::before {
    content:'';
    position: absolute;
    width: 15px;
    height: 15px;
    border-radius: 100%;
    background:white;
    transition: all 0.36s;
    top: 0px;
    left: 0px;
}

.switch-wrapper input:checked + label {
    background: mediumaquamarine;
    transition: all 0.36s cubic-bezier(.78, .14, .15, .86);
```

```
}

.switch-wrapper input:checked + label::before {
    left: 100%;
    transform: translateX(-100%);
    transition: all 0.36s cubic-bezier(.78, .14, .15, .86);
}

/* 解决 switch 消失时的延缓问题 */
#settings-holder:not(:hover) .switch-wrapper label::before{
    transition: all 0s;
}

#settings-holder:not(:hover) .switch-wrapper label{
    transition: all 0s;
}
```

装 订 线

3.4 JavaScript 源代码

```
const video = document.getElementById("video")
const videoName = document.getElementById("video-name")
const videoContainer = document.getElementById("video-container")
const videoControls = document.getElementById("video-control-bar")

const playpause = document.getElementById("playpause");
const mute = document.getElementById("mute");
const picInPic = document.getElementById("picinpic");
const fullscreen = document.getElementById("fs");
const currentTime = document.getElementById("current-time");
const totalTime = document.getElementById("total-time");

const progress = document.getElementById("progress");
const progressDot = document.getElementById("progress-dot");
const progressBar = document.getElementById("progress-bar");

const volumeValue = document.getElementById("volume-value");
const volumeBar = document.getElementById("volume-bar");
const volumeBarDone = document.getElementById("volume-bar-done");

const multipleSpeedList = document.getElementById("multiple-speed-choices
").getElementsByTagName('li');

const videoAlternativeBar = document.getElementById("video-chooser")
```

```
var mouseDragForProgress = false;           // 鼠标拖拽进度条
var mouseDragForVolume = false;             // 鼠标拖拽音量条

video.controls = false;          // 禁用自带的 controls


// 视频选择列表
let focusAlterOrder = 0
const videoAlternativeList = [
    { src : "./video1.mp4", pic : "./pic1.jpg", title : "《你好，李焕英》", s
ubtitle : "贾玲 张小斐" },
    { src : "./video2.mp4", pic : "./pic2.jpg", title : "流浪地球 2 · 先导预
告片", subtitle : "郭帆 吴京 王智" },
    { src : "./video3.mp4", pic : "./pic3.jpg", title : "电影版想见你预告片
", subtitle : "柯佳嬿 许光汉" },
    { src : "./music1.m4a", pic : "./pic4.jpg", title : "Lavender Haze", su
btitle : "Taylor Swift"}
]


// 填充视频列表某元素的 Html 属性
function fillVideoAlter(item, order){
    const title = "<p class = \"title\">" + item.title + "</p>";
    const subtitle = "<p class = \"subtitle\">" + item.subtitle + "</p>";
    const detail = "<div class = \"detail\">" + title + subtitle + "</div>
";
    const videopic = "<img src=\"" + item.pic +"\">";
    videoAlternativeBar.innerHTML += "<div class = \"video-alternative\" i
d = \"videoAlter"+ String(order) + "\">" + videopic +  detail + "</div>";
}

// 改变视频列表顺序
function changeVideoAlter(){
    videoAlternativeBar.innerHTML = "";
    // 当前选中的视频在列表首位
    fillVideoAlter(videoAlternativeList[focusAlterOrder], focusAlterOrde
r);
    for(const order in videoAlternativeList){
        if(videoAlternativeList[order] != videoAlternativeList[focusAlterO
rder])
            fillVideoAlter(videoAlternativeList[order], order);
    }

}

// 加载视频选择列表
```

```
function loadVideoAlter(){
    changeVideoAlter();
    for(let order in videoAlternativeList){
        document.getElementById("videoAlter"+ String(order)).addEventListe
ner("click", (e) => {
            video.src = videoAlternativeList[order]['src'];
            videoName.innerHTML = videoAlternativeList[order]['title'];
            focusAlterOrder = order;
            changeVideoAlter();
            totalTime.innerHTML = getFormatTime(video.duration, video.durat
ion);    // 初始化视频长度
            progressDot.style.marginLeft = 0;
                                        // 初始化圆点位置
            // 音频用封面图
            if(videoAlternativeList[order]['src'].slice(-4) == ".m4a")
                document.getElementById("video-container").style.backgroun
dImage = "url(" + videoAlternativeList[order]['pic'] + ")";
            // 其他的为黑色底色（background 设置为 None 的话会影响其他 css 属性，
再次设置图片要重设，因此此处也用 Image）
            else
                document.getElementById("video-container").style.backgroun
dImage = "linear-gradient(to bottom, black, black)";
            loadVideoAlter();
        });
    }
}

loadVideoAlter();

// 释放拖动进度条
document.body.addEventListener("mouseup", (e) => {
    mouseDragForProgress = false;
    mouseDragForVolume = false;
});


// 监听鼠标移动
document.body.addEventListener("mousemove", (e) => {
    if(mouseDragForProgress){
        const barRec = progress.getBoundingClientRect();
        const pos = (e.pageX - barRec.left) / progress.offsetWidth;
        video.currentTime = pos * video.duration;
    }
    else if(mouseDragForVolume){
        const barRec = volumeBar.getBoundingClientRect();
        var pos = (barRec.bottom - e.pageY) / volumeBar.offsetHeight;
```

```
        if(pos > 1)
            pos = 1;
        else if(pos < 0)
            pos = 0;
        video.volume = pos;
    }
});

// 监听全屏
window.addEventListener('fullscreenchange', () => {
    const isFullScreen = document.fullScreen || document.mozFullScreen || d
ocument.webkitIsFullScreen
    if(isFullScreen) {
        fullscreen.src = "./icon/fullscreenexit.svg";
    }
    else{
        fullscreen.src = "./icon/fullscreen.svg";
    }

    // 防止进度条圆点错位
    progressDot.style.marginLeft = (progress.value / video.duration) * pro
gress.offsetWidth - progressDot.offsetWidth / 2 + "px";
})


// 播放按钮
playpause.addEventListener("click", (e) => {
    if (video.paused || video.ended) {
        video.play();
        playpause.src = "./icon/pause.svg"
    }
    else {
        video.pause();
        playpause.src = "./icon/play.svg"
    }
});


// 点击视频本身同理暂停/播放
video.addEventListener("click", (e) => {
    if (video.paused || video.ended) {
        video.play();
        playpause.src = "./icon/pause.svg"
    }
    else {
        video.pause();
```

```
        playpause.src = "./icon/play.svg"
    }
});


// 监听音量变化
video.addEventListener("volumechange", (e)=>{
    volumeValue.innerHTML = Math.floor(video.volume * 100);
    volumeBarDone.style.height = video.volume * volumeBar.offsetHeight + '
px';
});


// 点击音量条
volumeBar.addEventListener("click", (e) => {
    const barRec = volumeBar.getBoundingClientRect();
    var pos = (barRec.bottom - e.pageY) / volumeBar.offsetHeight;
    if(pos > 1)
        pos = 1;
    else if(pos < 0)
        pos = 0;
    video.volume = pos;
});

// 拖动音量条
volumeBar.addEventListener("mousedown", (e) => {
    mouseDragForVolume = true;
})


// 静音键
mute.addEventListener("click", (e) => {
    video.muted = !video.muted;
    if(video.muted)
        mute.src = "./icon/mute.svg";
    else
        mute.src = "./icon/volume.svg";
});

// 画中画
picInPic.addEventListener("click", () => {
    video.requestPictureInPicture();
})


// 同步进度条时间
```

```
video.addEventListener("timeupdate", () => {
    progress.setAttribute("max", video.duration);
    progress.value = video.currentTime;

    // 控制圆点
    progressDot.style.marginLeft = (progress.value / video.duration) * pro
gress.offsetWidth - progressDot.offsetWidth / 2 + "px";

    // 时间显示
    currentTime.innerHTML = getFormatTime(video.currentTime, video.duratio
n);
    totalTime.innerHTML = getFormatTime(video.duration, video.duration);
});


// 点击进度条
progress.addEventListener("click", (e) => {
    const barRec = progress.getBoundingClientRect();
    const pos = (e.pageX - barRec.left) / progress.offsetWidth;
    video.currentTime = pos * video.duration;
});


// 拖动进度条
progress.addEventListener("mousedown", (e) => {
    mouseDragForProgress = true;
});


// 全屏
fullscreen.addEventListener("click", (e) => {
    if (document.fullscreenElement !== null) {
        // 当下处于全屏模式
        document.exitFullscreen();
    }
    else {
        // 当下并非全屏模式
        videoContainer.requestFullscreen();
    }
    videoContainer.setAttribute("data-fullscreen", !!document.fullscreenEl
ement);
});


// 根据总时长格式化为 00:00 或 00:00:00 的格式
function getFormatTime(time, duration) {
```

装 订 线

```
    var time = time || 0;

    var h = parseInt(time / 3600),
        m = parseInt(time % 3600 / 60),
        s = parseInt(time % 60);

    s = s < 10 ? "0" + s : s;

    if (duration >= 3600){
        m = m < 10 ? "0" + m : m;
        h = h < 10 ? "0" + h : h;
        return h + ":" + m + ":" + s;
    }
    else{
        if(m == 0)
            m = "00";
        else
            m = m < 10 ? "0" + m : m;
        return m + ":" + s;
    }
}

// 监听倍速列表中的元素
var speedSelected = multipleSpeedList[3];
speedSelected.style.color = "mediumaquamarine";
for(var i = 0; i < multipleSpeedList.length; i++){
    multipleSpeedList[i].addEventListener('click', function() {
        video.playbackRate = parseFloat(this.innerHTML);
        speedSelected.style.color = "white";
        speedSelected = this;
        speedSelected.style.color = "mediumaquamarine";
    })
}


// 开启视频循环
function openVideoCycle(obj){
    if(obj.checked){
        video.loop = true;
    }
    else
        video.loop = false;
}

// 开启视频镜像
function openVideoMirror(obj){
```

```
    if(obj.checked){
        video.style.transform = "rotateY(180deg)";
    }
    else
        video.style.transform = "none";
}
```

四、心得体会

在这次的网页设计实验中，我成功地实现了一个媒体播放器，巩固了已经掌握的 HTML、CSS 和 JavaScript 相关知识。通过嵌入 JavaScript 脚本语言，我成功地设计了应用程序，并掌握了基本的 JavaScript 语法。在上一个实验中，我便实践了 HTML 与 CSS 的布局设计；在此次的媒体播放器实验中我又一次更加熟练地运用了 HTML 与 CSS 的相关知识。在实现媒体播放器的过程中，我使用了 HTML5 的<video>标签和 JavaScript 的媒体 API，实现了播放预定的音频或视频、开始和暂停功能。除此之外，我还扩展了倍速、音量控制、全屏和进度条等功能；同时，我还添加了播放列表功能，可以从右侧的列表中选取不同的媒体源进行播放。在实现这些功能的过程中，我更加深入地了解了 JavaScript 语言的特性和应用场景，并熟练掌握了 JavaScript 语言的基本语法、流程控制和函数定义等知识点。同时，我也发现了 JavaScript 语言的灵活性和高效性，通过编写少量的代码即可实现丰富的功能。这也让我更加有信心在未来的工作中使用 JavaScript 来进行网页开发和应用程序设计。

在实现媒体播放器的过程中，我也注意到了代码规范性和可读性的重要性。由于 JavaScript 语言的灵活性，代码的可读性和维护性往往容易受到影响。因此，在编写 JavaScript 代码时，我尽量使用规范的语法和命名方式，同时注重代码的可读性和可维护性。这样可以能够方便代码的理解与修改，提高工作效率和代码质量，也希望在之后的实践中我也能继续培养良好的代码习惯。

总的来说，这次实验让我更加深入地了解了前端开发中 JavaScript 语言的应用和重要性。同时，我也更加认识到代码规范性和可读性的重要性，这对提高代码质量和工作效率都非常关键。在未来的学习和实践中，我会继续努力，将所学到的知识学以致用，从而不断提高自己的技术水平和网页设计能力。