1. <video> 태그와 JavaScript API 개요

HTML의 <video> 태그는 브라우저 내에서 동영상 재생을 가능하게 해줍니다. 기본적으로 controls 속성을 추가하면 브라우저 기본 재생 컨트롤(재생, 일시정지, 볼륨 조절 등)을 제공하지만, JavaScript를 사용하면 이 태그를 보다 세밀하게 제어할 수 있습니다.

주요 API 및 속성

- **메서드**
  - play(): 동영상 재생을 시작합니다.
  - pause(): 동영상 재생을 일시정지합니다.
  - load(): 소스 파일을 다시 로드합니다.
- **속성**
  - currentTime: 현재 재생 위치(초 단위)를 반환하거나 설정할 수 있습니다.
  - duration: 동영상의 전체 길이(초 단위)를 반환합니다.
  - paused: 동영상이 일시정지 상태인지 여부를 반환합니다.
  - volume: 볼륨을 0.0(무음)부터 1.0(최대)까지 조절합니다.
  - muted: 동영상의 음소거 여부를 설정하거나 반환합니다.
  - playbackRate: 재생 속도를 설정하거나 반환합니다.
- **이벤트**
  - play: 동영상이 재생될 때 발생합니다.
  - pause: 동영상이 일시정지될 때 발생합니다.
  - timeupdate: 재생 시간이 변경될 때마다 발생합니다.
  - ended: 동영상 재생이 완료되었을 때 발생합니다.

이 API들은 기본 컨트롤 외에도 학생들이 자신만의 사용자 정의 인터페이스(예, 재생/일시정지 버튼, 현재 시간 표시, 건너뛰기 버튼 등)를 만들 수 있도록 도와줍니다.

---

2. HTML 예제: 기본 video 컨트롤 만들기

아래 예제는 학생들이 HTML 파일 하나만으로 실행해 볼 수 있는 코드입니다. 이 예제에서는 <video> 태그를 사용하여 동영상을 삽입하고, JavaScript를 통해 재생, 일시정지, 그리고 10초 건너뛰기 기능을 구현합니다.

```
<!DOCTYPE html>
<html lang="ko">
```

```
<head>
    <meta charset="UTF-8">
    <title>Video 태그 JavaScript API 예제</title>
    <style>
        body {
            font-family: sans-serif;
            max-width: 700px;
            margin: 2em auto;
            line-height: 1.6;
        }

        button {
            margin: 0.5em;
            padding: 0.5em 1em;
            font-size: 16px;
        }

        #currentTime {
            font-weight: bold;
        }
    </style>
</head>

<body>
    <h1>Video 태그와 JavaScript API 예제</h1>

    <!-- video 태그: 재생할 동영상 파일(sample.mp4)을 본인 환경에 맞게 수정하세요. -->
    <video id="myVideo" width="640" height="360" controls>
        <source src="quiz/videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>

    <!-- 사용자 정의 컨트롤 -->
    <div>
        <button onclick="playVideo()">Play</button>
        <button onclick="pauseVideo()">Pause</button>
        <button onclick="skip()">10 초 건너뛰기</button>
    </div>

    <div>
        <span>현재 시간: </span>
        <span id="currentTime">0.00</span> 초
    </div>

    <!-- JavaScript 코드 -->
    <script>
        // video 태그 요소를 가져옵니다.
        const video = document.getElementById('myVideo');

        // 영상의 재생 시간이 변경될 때마다 실행되어 현재 시간을 업데이트합니다.
        video.addEventListener('timeupdate', () => {
            document.getElementById('currentTime').innerText =
video.currentTime.toFixed(2);
        });
```

```
        // video API를 활용하여 동영상을 재생합니다.
        function playVideo() {
            // 동영상이 끝나면 처음부터 재생하도록 설정할 수도 있습니다.
            if (video.ended) {
                video.currentTime = 0;
            }
            video.play();
        }

        // video API를 활용하여 동영상을 일시정지합니다.
        function pauseVideo() {
            video.pause();
        }

        // 10초 앞으로 건너뛰기. 단, 총 길이를 초과하지 않도록 합니다.
        function skip() {
            // 만약 건너뛴 시간이 총 길이를 초과하면 마지막으로 이동합니다.
            if ((video.currentTime + 10) < video.duration) {
                video.currentTime += 10;
            } else {
                video.currentTime = video.duration;
            }
        }
    </script>
</body>

</html>
```

예제 설명

1. **HTML 구조**
    - <video> 태그에 id="myVideo"를 부여하여 JavaScript에서 쉽게 제어할 수 있도록 했습니다.
    - controls 속성을 추가해서 브라우저가 기본 제공하는 재생 컨트롤을 표시합니다.
    - <source> 태그를 이용해 동영상 파일을 지정합니다.
2. **사용자 정의 컨트롤**
    - Play, Pause, 10초 건너뛰기 버튼을 만들어 각각의 기능을 호출합니다.
3. **JavaScript 코드**
    - document.getElementById로 <video> 요소를 선택한 후, 해당 요소에 이벤트 리스너(timeupdate)를 추가하여 현재 재생 시간을 실시간으로 업데이트합니다.
    - playVideo(), pauseVideo(), skip() 함수는 <video> 태그의 API 메서드를 사용하여 동작을 제어합니다.

연습문제

아래는 **video 태그의 Web API**를 활용하는 10개의 문제와, 각 문제에 대해 하나의 HTML 파일에서 바로 실행 가능한 정답 예제입니다. 각 문제는 요구사항을 명확하게 작성했으니, 학생들이 어떤 기능을 구현해야 하는지 명확하게 이해할 수 있습니다.

문제 1: Custom Play/Pause Toggle 버튼 만들기

요구사항:

- <video> 태그가 포함된 HTML 파일을 작성합니다.
- 하나의 버튼을 만들어서, 버튼을 클릭할 때마다 동영상이 재생 중이면 일시정지, 정지 상태이면 재생하도록 합니다.
- 버튼의 텍스트는 현재 상태("Play" 혹은 "Pause")에 맞게 변해야 합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 1: Custom Play/Pause Toggle</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
            margin-top: 2em;
        }

        button {
            padding: 0.5em 1em;
            font-size: 16px;
        }
    </style>
</head>

<body>
    <h1>문제 1: Custom Play/Pause Toggle</h1>
    <video id="myVideo" width="640" height="360">
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <br>
    <button id="toggleBtn">Play</button>

    <script>
        const video = document.getElementById('myVideo');
        const toggleBtn = document.getElementById('toggleBtn');
```

```
        toggleBtn.addEventListener('click', function () {
            if (video.paused) {
                video.play();
                toggleBtn.innerText = 'Pause';
            } else {
                video.pause();
                toggleBtn.innerText = 'Play';
            }
        });

        // 동영상이 끝나면 버튼을 'Play'로 변경합니다.
        video.addEventListener('ended', function () {
            toggleBtn.innerText = 'Play';
        });
    </script>
</body>

</html>
```

문제 2: 커스텀 진행바(Progress Bar) 만들기

요구사항:

- HTML 파일 내에 <video> 태그와 input type="range"를 이용한 진행바를 만듭니다.
- 동영상이 재생될 때마다 timeupdate 이벤트를 활용하여 진행바의 값이 업데이트되도록 합니다.
- 진행바를 직접 조작하면 해당 위치로 동영상이 이동해야 합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 2: Custom Progress Bar</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
            margin-top: 2em;
        }

        input[type="range"] {
            width: 640px;
        }
    </style>
</head>
```

```
<body>
    <h1>문제 2: Custom Progress Bar</h1>
    <video id="myVideo" width="640" height="360">
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <br>
    <input type="range" id="progressBar" value="0" step="0.1">

    <script>
        const video = document.getElementById('myVideo');
        const progressBar = document.getElementById('progressBar');

        video.addEventListener('loadedmetadata', function () {
            progressBar.max = video.duration;
        });

        video.addEventListener('timeupdate', function () {
            progressBar.value = video.currentTime;
        });

        progressBar.addEventListener('input', function () {
            video.currentTime = progressBar.value;
        });
    </script>
</body>

</html>
```

문제 3: 10초 앞으로/뒤로 건너뛰기 기능 구현

요구사항:

- HTML 파일에 <video> 태그와 두 개의 버튼("10초 뒤로", "10초 앞으로")을 만듭니다.
- 버튼을 클릭하면 동영상이 각각 10초 뒤로 또는 앞으로 이동해야 합니다.
- 동영상의 시작(0초)과 종료(동영상 길이)를 넘어가지 않도록 합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 3: Skip Forward/Backward</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
```

```
            margin-top: 2em;
        }

        button {
            padding: 0.5em 1em;
            margin: 0.5em;
            font-size: 16px;
        }
    </style>
</head>

<body>
    <h1>문제 3: Skip Forward and Backward</h1>
    <video id="myVideo" width="640" height="360" controls>
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <br>
    <button onclick="skipBackward()">10 초 뒤로</button>
    <button onclick="skipForward()">10 초 앞으로</button>

    <script>
        const video = document.getElementById('myVideo');

        function skipForward() {
            video.currentTime = Math.min(video.currentTime + 10, video.duration);
        }

        function skipBackward() {
            video.currentTime = Math.max(video.currentTime - 10, 0);
        }
    </script>
</body>

</html>
```

문제 4: 사용자 정의 볼륨 컨트롤(slider)과 음소거 버튼 구현

요구사항:

- HTML 파일 내에 <video> 태그와 input type="range"를 사용한 볼륨 슬라이더, 그리고 음소거를 토글할 수 있는 버튼을 추가합니다.
- 슬라이더의 값에 따라 동영상의 volume 프로퍼티가 변경되어야 합니다.
- 버튼을 클릭할 때마다 동영상의 muted 상태가 토글되고, 버튼 텍스트도 변경되어야 합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">
```

```html
<head>
    <meta charset="UTF-8">
    <title>문제 4: Volume Slider and Mute Toggle</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
            margin-top: 2em;
        }

        input[type="range"] {
            width: 300px;
        }

        button {
            padding: 0.5em 1em;
            margin-left: 1em;
            font-size: 16px;
        }
    </style>
</head>

<body>
    <h1>문제 4: Volume Slider and Mute Toggle</h1>
    <video id="myVideo" width="640" height="360" controls>
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <br>
    <input type="range" id="volumeSlider" min="0" max="1" step="0.01" value="1">
    <button id="muteToggle">Mute</button>

    <script>
        const video = document.getElementById('myVideo');
        const volumeSlider = document.getElementById('volumeSlider');
        const muteToggle = document.getElementById('muteToggle');

        volumeSlider.addEventListener('input', function () {
            video.volume = volumeSlider.value;
            if (video.volume == 0) {
                video.muted = true;
                muteToggle.innerText = "Unmute";
            } else {
                video.muted = false;
                muteToggle.innerText = "Mute";
            }
        });

        muteToggle.addEventListener('click', function () {
            video.muted = !video.muted;
            muteToggle.innerText = video.muted ? "Unmute" : "Mute";
            // 슬라이더 값 동기화
            if (video.muted) {
```

```
                volumeSlider.value = 0;
            } else {
                volumeSlider.value = video.volume;
            }
        });
    </script>
</body>

</html>
```

문제 5: 재생 속도 제어 기능 구현

요구사항:

- HTML 파일에 <video> 태그와 "속도 증가", "속도 감소" 버튼을 추가합니다.
- 버튼을 클릭하면 동영상의 playbackRate 속도가 일정 값(예, 0.25) 만큼 증가 혹은 감소해야 합니다.
- 재생 속도의 변경 결과를 화면에 표시합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 5: Playback Rate Controller</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
            margin-top: 2em;
        }

        button {
            padding: 0.5em 1em;
            margin: 0.5em;
            font-size: 16px;
        }

        #rateDisplay {
            font-weight: bold;
        }
    </style>
</head>

<body>
    <h1>문제 5: Playback Rate Controller</h1>
    <video id="myVideo" width="640" height="360" controls>
```

```
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <br>
    <button onclick="decreaseRate()">속도 감소</button>
    <button onclick="increaseRate()">속도 증가</button>
    <p>현재 재생 속도: <span id="rateDisplay">1.0</span>x</p>

    <script>
        const video = document.getElementById('myVideo');
        const rateDisplay = document.getElementById('rateDisplay');

        function updateRateDisplay() {
            rateDisplay.innerText = video.playbackRate.toFixed(1);
        }

        function increaseRate() {
            video.playbackRate += 0.25;
            updateRateDisplay();
        }

        function decreaseRate() {
            video.playbackRate = Math.max(0.25, video.playbackRate - 0.25);
            updateRateDisplay();
        }
    </script>
</body>

</html>
```

문제 6: 전체 화면(Fullscreen) 토글 버튼 구현

요구사항:

- HTML 파일에 <video> 태그와 '전체 화면 토글' 버튼을 추가합니다.
- 버튼을 클릭하면 Fullscreen API를 사용해 동영상 요소가 전체 화면 모드로 전환되고, 전체 화면 상태일 때 다시 클릭하면 원래 상태로 돌아와야 합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 6: Fullscreen Toggle</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
```

```
            margin-top: 2em;
        }

        button {
            padding: 0.5em 1em;
            font-size: 16px;
        }
    </style>
</head>

<body>
    <h1>문제 6: Fullscreen Toggle</h1>
    <video id="myVideo" width="640" height="360" controls>
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <br>
    <button id="fullscreenBtn">전체 화면 토글</button>

    <script>
        const video = document.getElementById('myVideo');
        const fullscreenBtn = document.getElementById('fullscreenBtn');

        fullscreenBtn.addEventListener('click', function () {
            if (!document.fullscreenElement) {
                if (video.requestFullscreen) {
                    video.requestFullscreen();
                } else if (video.webkitRequestFullscreen) { /* Safari */
                    video.webkitRequestFullscreen();
                } else if (video.msRequestFullscreen) { /* IE11 */
                    video.msRequestFullscreen();
                }
            } else {
                if (document.exitFullscreen) {
                    document.exitFullscreen();
                }
            }
        });
    </script>
</body>

</html>
```

문제 7: 동영상 종료시 메시지 표시 및 재생 리셋 버튼 구현

요구사항:

- HTML 파일 내 <video> 태그를 사용하여 동영상을 재생하고, 동영상이 끝났을 때 ended 이벤트를 활용해 "재생이 종료되었습니다." 메시지를 화면에 표시합니다.

- 메시지와 함께 "처음부터 다시 재생" 버튼이 나타나며, 버튼 클릭 시 동영상이 처음부터 재생되어야 합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 7: End-of-Video Message and Reset</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
            margin-top: 2em;
        }

        #message {
            margin-top: 1em;
            font-size: 18px;
            color: red;
        }

        button {
            padding: 0.5em 1em;
            font-size: 16px;
        }
    </style>
</head>

<body>
    <h1>문제 7: End-of-Video Message and Reset</h1>
    <video id="myVideo" width="640" height="360" controls>
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <br>
    <button id="resetBtn" style="display:none;">처음부터 다시 재생</button>
    <div id="message"></div>

    <script>
        const video = document.getElementById('myVideo');
        const message = document.getElementById('message');
        const resetBtn = document.getElementById('resetBtn');

        video.addEventListener('ended', function () {
            message.innerText = "재생이 종료되었습니다.";
            resetBtn.style.display = "inline-block";
        });

        resetBtn.addEventListener('click', function () {
            video.currentTime = 0;
```

```
            video.play();
            message.innerText = "";
            resetBtn.style.display = "none";
        });
    </script>
</body>

</html>
```

문제 8: 클릭 가능한 진행바를 통한 탐색(seeking) 기능 구현

요구사항:

- HTML 파일 내에 <video> 태그 외에 **div** 요소를 진행바로 구성합니다.
- 진행바는 배경 div와, 동영상 재생 비율에 맞춰 폭이 변화하는 색상 div로 구성합니다.
- 사용자가 진행바의 원하는 위치를 클릭하면, 동영상의 currentTime이 해당 비율에 맞춰 변경되어야 합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 8: Clickable Progress Bar</title>
    <style>
        body {
            font-family: sans-serif;
            text-align: center;
            margin-top: 2em;
        }

        #progressContainer {
            width: 640px;
            height: 20px;
            background: #ddd;
            margin: 1em auto;
            cursor: pointer;
            position: relative;
        }

        #progress {
            height: 100%;
            width: 0;
            background: #76a5af;
        }
    </style>
</head>
```

```
<body>
    <h1>문제 8: Clickable Progress Bar for Seeking</h1>
    <video id="myVideo" width="640" height="360" controls>
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <div id="progressContainer">
        <div id="progress"></div>
    </div>

    <script>
        const video = document.getElementById('myVideo');
        const progressContainer = document.getElementById('progressContainer');
        const progress = document.getElementById('progress');

        video.addEventListener('timeupdate', function () {
            const percent = (video.currentTime / video.duration) * 100;
            progress.style.width = percent + '%';
        });

        progressContainer.addEventListener('click', function (e) {
            const rect = progressContainer.getBoundingClientRect();
            const clickX = e.clientX - rect.left;
            const newTime = (clickX / progressContainer.clientWidth) * video.duration;
            video.currentTime = newTime;
        });
    </script>
</body>

</html>
```

문제 9: 동영상 정보(현재 시간, 전체 길이, 완료율) 표시하기

요구사항:

- HTML 파일에 <video> 태그와 현재 재생 시간, 전체 동영상 길이 및 완료율(%)을 표시할 수 있는 영역을 만듭니다.
- loadedmetadata와 timeupdate 이벤트를 활용해 각각 동영상의 전체 길이와 현재 재생 시간이 갱신되며, 완료율도 실시간으로 계산하여 표시합니다.

정답 예제 :

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 9: Display Video Information</title>
    <style>
```

```
        body {
            font-family: sans-serif;
            text-align: center;
            margin-top: 2em;
        }

        #info {
            margin-top: 1em;
            font-size: 18px;
        }
    </style>
</head>

<body>
    <h1>문제 9: Display Video Information</h1>
    <video id="myVideo" width="640" height="360" controls>
        <source src="videos/sample1.mp4" type="video/mp4">
        브라우저가 video 태그를 지원하지 않습니다.
    </video>
    <div id="info">
        <p>현재 시간: <span id="currentTime">0.00</span> 초</p>
        <p>전체 길이: <span id="duration">0.00</span> 초</p>
        <p>완료율: <span id="percentage">0</span>%</p>
    </div>

    <script>
        const video = document.getElementById('myVideo');
        const currentTimeEl = document.getElementById('currentTime');
        const durationEl = document.getElementById('duration');
        const percentageEl = document.getElementById('percentage');

        video.addEventListener('loadedmetadata', function () {
            durationEl.innerText = video.duration.toFixed(2);
        });

        video.addEventListener('timeupdate', function () {
            currentTimeEl.innerText = video.currentTime.toFixed(2);
            const percent = (video.currentTime / video.duration) * 100;
            percentageEl.innerText = percent.toFixed(1);
        });
    </script>
</body>

</html>
```