Video Player

Create a web video player in which you can alter playback speed and skip forward/backward.

Task:

Implement a video player that can play an mp4 video. Add five buttons below the video player:

- `1x`, `1.5x`, `2x`: when clicked, it sets the playback speed to the displayed value on the button
- `-30s`, `+30s` when clicked, it offsets the current time of the video by the displayed value

You can use HTML5 tags in the exercise, and you don't have to worry about cross-browser compatibility.

As an example, you can use this video.

Make your solution extensible so that it will be easy to add more fully functional playback speed and offset buttons without changing anything in your JavaScript code.

Solution:

If you have never used the HTML5 video API, it's time to google it. The video markup looks as follows:

```
<video class="js-video"
       width="540"
       height="360"
       controls
       src="http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4"></video>
```

We will also need some buttons:

```
<button class="js-speed-button"  data-speed="1">1x</button>
<button class="js-speed-button"  data-speed="1.5">1.5x</button>
<button class="js-speed-button"  data-speed="2">2x</button>

<button class="js-offset-button" data-offset="30">+30s</button>
<button class="js-offset-button" data-offset="-30">-30s</button>
```

I kept the markup lean, and added some classes that will make it easy to identify each button. As we have to take care of extensibility, it makes sense to reference each button with the same class.

We also used some data attributes to customize the offset. We will read these attributes in the event handlers.

We will write the event handlers such that the code handles all buttons of the same type generically:

```
document.querySelectorAll( '.js-speed-button' ).forEach( item =>
    item.addEventListener( 'click', function( e ) {
        // Handle the playback speed update
    } )
);

document.querySelectorAll( '.js-offset-button' ).forEach( item =>
    item.addEventListener( 'click', function( e ) {
        // Handle the offset update
    } )
);
```

`document.querySelectorAll` is similar to jQuery's `$` function. We pass it a selector, and it returns a DOM node collection. As this collection is an *iterable*, we can iterate on the elements. We can attach an event listener to each event.

Study the Video API reference to conclude how to change the playback speed and the offset.

Let's start with the playback speed:

```
document.querySelectorAll( '.js-speed-button' ).forEach( item =>
    item.addEventListener( 'click', function( e ) {
        const speed = e.target.dataset.speed;
        const video = document.querySelector( '.js-video' );
        video.playbackRate = speed;
    } )
);
```

Notice how we retrieve the `data-speed` attribute: `e.target.dataset` contains all the data attributes belonging to a DOM node.

The `.js-video` `video` HTML5 element has a `playbackRate` property. If we set it to a floating point, we can change the playback rate.

We can conclude this exercise with the implementation of the offset buttons:

```
document.querySelectorAll( '.js-offset-button' ).forEach( item =>
    item.addEventListener( 'click', function( e ) {
        const video = document.querySelector( '.js-video' );
        const duration = video.duration;
        const offset = Number.parseInt( e.target.dataset.offset );
        let newTime = video.currentTime + offset;
        if ( newTime > duration ) newTime = duration;
        if ( newTime < 0 ) newTime = 0;

        video.currentTime = newTime;
    } )
);
```

The video duration is in seconds. We can retrieve the offset from the `data-offset` attribute via the property `e.target.dataset.offset`. The `currentTime` property of the video contains the place where the video is at currently. After adding the offset to the current time, we have to check if we are still within the boundaries of the video to avoid indexing out from the video.

As the `currentTime` property of the video element is writable, we have to assign the new value to it to make the offset work.

Experiment with the solution in this codepen.