JavaScript Audio and Video Review

Audio Constructor and Common Methods

- **Definition**: The Audio constructor, like other constructors, is a special function called with the new keyword. It returns an HTMLAudioElement, which you can then use to play audio for the user or append to the DOM for the user to control themselves. When you call the constructor, you can optionally pass a URL as the (only) argument. This URL should point to the source of the audio file you want to play. Or, if you need to change the source dynamically, you can assign the URL to the src property of the returned audio element.
- **play() Method**: This method is used with the audio or video elements to begin playback for the media.

Example Code

```
const audio = document.getElementById('audio');


// Starts playing the audio

audio.play();
```

- **pause() Method**: This method is used with the audio or video elements to pause playback for the media. It maintains the current position, so if play() is called, it starts from that position.

Example Code

```
function pauseAudio() {

  const audio = document.getElementById('myAudio');

  audio.pause();  // Pauses the audio playback

}
```

- **addTextTrack() Method**: This method allows you to specify a text track to associate with the media element - which is especially helpful for adding subtitles to a video.
- **fastSeek() Method**: This method allows you to move the playback position to a specific time within the media.

The Audio() constructor has properties just as it has methods. Those properties include:

- currenttime: for getting the current playback time of an audio
- loop: for making the audio play continuously by automatically restarting when it reaches the end
- muted: for silencing the audio output of a media element regardless of volume setting

Different Audio and Video Formats

- **MIME type**: A MIME type, standing for Multipurpose Internet Mail Extensions, is a standardized way to programmatically indicate a file type. The MIME type can tell an application, such as your browser, how to handle a specific file. In the case of audio and video, the MIME type indicates it is a multimedia format that can be embedded in the web page.
- **source Element**: This is used to specify a file type and source - and can include multiple different types by using multiple source elements. When you do this, the browser will determine the best format to use for the user's current environment.
- **MP3**: This is a type of digital file format used to store music, audio, or sound. It's a compressed version of a sound recording that makes the file size smaller, so it's easier to store and share. MP3 has the widest browser support and the MIME type of audio/mp3.
- **MP4**: An MP4 is a type of digital file format used to store video and audio. It serves as a container that holds both the video (images) and the sound (music or speech) in one file. An MP4, can have the MIME type audio/mp4 OR video/mp4, depending on whether it's a video file or audio-only.
- Other formats are WMV which is associated with the Windows Media Player app, OGG, MKV, AVI, and more.

codecs

- **Definition**: A codec, short for "encoder/decoder", is an algorithm or software that can convert audio and video between analogue and digital formats. Codecs can be specified as part of the MIME type. The basic syntax to define a codec is to add a semi-colon after the media type, then codecs= and the codec.

HTMLMediaElement API

- **Definition**: The HTMLMediaElement API is used to control the behavior of audio and video elements on your page. It extends the base HTMLElement interface, so you have access to the base properties, methods as well as these helpful events like waiting, ended, canplay, canplaythrough, and more. Examples of the methods include play(), fastSeek(), pause(), and canPlayType() for checking if a browser is likely to be able to play an audio file.

Media Capture and Streams API

- **Definition**: The Media Capture and Streams API, or the MediaStream API, is used to capture audio and video from your device. In order to use the API, you need to create the MediaStream object. You could do this with the constructor, but it would not be tied to the user's hardware. Instead, the mediaDevices property of the global navigator object has a getUserMedia() method for you to use.

Example Code

```
window.navigator.mediaDevices.getUserMedia({
 audio: true,
 video: {
  width: {
   min: 1280,
   ideal: 1920,
   max: 3840
  },
  height: {
   min: 720,
   ideal: 1080,
   max: 2160
  }
 }
});
```

Screen Capture API

- **Definition**: The Screen Capture API is used to record a user's screen. This API is exposed by calling the getDisplayMedia() method of the mediaDevices object and consuming the returned media stream.

MediaStream Recording API

- **Definition**: The MediaStream Recording API works in tandem with the MediaStreams APIs, allowing you to record a MediaStream (or even an HTMLMediaElement directly).

Media Source Extensions API

- **topic**: The Media Source Extensions API is what allows you to directly pass a user's webcam feed to a video element with the srcObject property.

Web Audio API

- **Definition**: The Web Audio API which powers everything audible on the web. This API includes important objects like an AudioBuffer (representing a Buffer specifically containing audio data) or the AudioContext, used to represent an audio-processing graph.