Picture-in-Picture Web API

Description

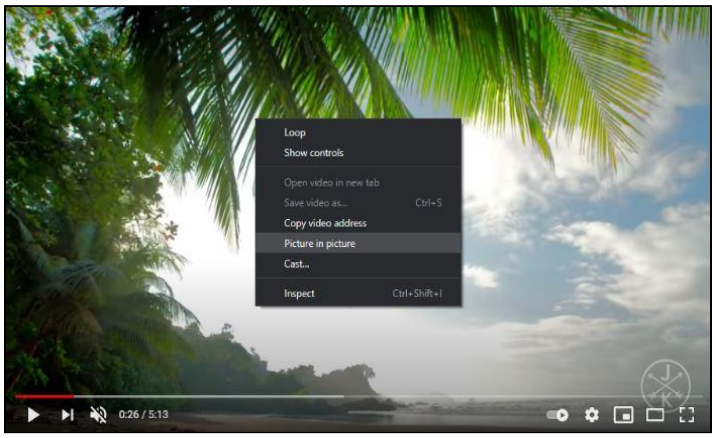
What is Picture-in-Picture?

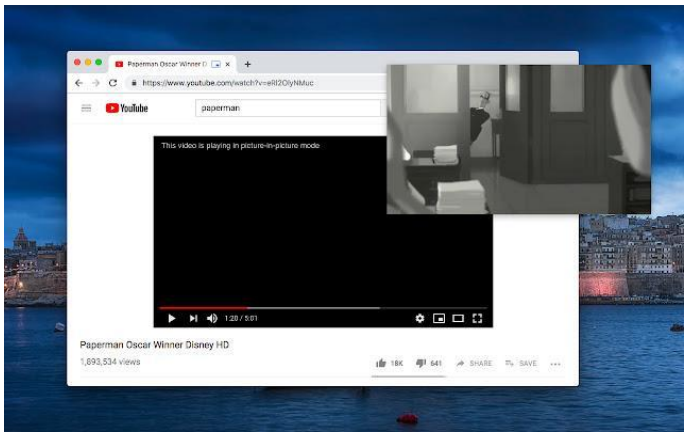
Picture-in-Picture is an API that allows a video to be displayed over other windows, a sort of “floating video” that can sit on top of other tabs, windows, or applications. The purpose of Picture-in-Picture is to allow the user to interact with other applications while watching a specific video. The problem the API solves is multi-tasking. It allows the user to simultaneously do other actions while watching the video.

Real-World Applications

YouTube

The largest video-sharing service, YouTube, uses Picture-in-Picture. In order to activate it, a video needs to be right-clicked twice in a row to open up a menu, where the “Picture in picture” button should be pressed. This gives the opportunity to more than 122 million daily YouTube users the option of watching videos using Picture-in-Picture.





Mobile Devices

The Picture-in-Picture Web API was modeled after the Android Picture-in-Picture created in April 2017. Given this, the application of Picture-in-Picture is very prevalent now in Android and iOS devices and is utilized by video-chat applications, streaming services, various browsers, and map applications.

Interface and Objects

HTMLVideoElement

All of Picture-in-Picture starts with an **`HTMLVideoElement`**, or `<video />` tag in HTML. When the `HTMLVideoElement` is retrieved, to enter Picture-in-Picture mode, the `requestPictureInPicture()` method must be called on it. An example below:

```
const video = document.getElementById("video");
const pipWindow = await video.requestPictureInPicture();
```

The `requestPictureInPicture()` returns a `Promise` object. If the request is accepted, it returns a `PictureInPictureWindow` object. Otherwise, it will return an exception,

discussed later. Since the `requestPictureInPicture()` method returns a `Promise`, *it is best practice to **await** it.*

PictureInPictureWindow

The **`PictureInPictureWindow`** object represents the video when it is in Picture-in-Picture mode, meaning that it is no longer displayed on the page, but rather floating above all applications.

The object has no methods, only two properties and an event. Its two properties are **`width`** and **`height`**, which represent the window's width and height, respectively.

The event that the `PictureInPictureWindow` object has is the **`resize`** event. This will be triggered when the window is resized. An example of the event is shown below:

```
const video = document.getElementById("video");
const pipWindow = await video.requestPictureInPicture();

pipWindow.addEventListener("resize", (ev) => {
     console.log(`This window is being resized! It is now
${pipWindow.width}px by ${pipWindow.height}px.`);
});
```

On resize, the Picture-in-Picture window's width and height are logged to console.

Document and Transitions

The `Document` object represents the entire web page and everything inside of it. It serves as the entry point to the content of the web page in the browser, and thus Picture-in-Picture is deeply connected to it. If a video is currently being displayed in Picture-in-Picture, it is stored in the `document.pictureInPictureElement`. Otherwise, it is set to `null`. To leave Picture-in-Picture, the `document.exitPictureInPicture()` method must be called.

Associated with this are two events, `"enterpictureinpicture"` and `"leavepictureinpicture"`.

The `"enterpictureinpicture"` event is triggered when a video enters Picture-in-Picture. This is triggered when the `Promise` returned by `video.requestPictureInPicture()` is accepted.

```
const video = document.getElementById("video");
video.addEventListener("enterpictureinpicture", (ev) => {
      console.log("Picture-in-Picture entered!");
}
```

The `"leavepictureinpicture"` event is triggered when a video exits Picture-in-Picture. This is triggered when the `Promise` returned by `document.exitPictureInPicture()` is accepted.

```
const video = document.getElementById("video");
video.addEventListener("leavepictureinpicture", (ev) => {
      console.log("Picture-in-Picture exited!");
}
```

Disabling Picture-in-Picture

There are two ways Picture-in-Picture can be disabled, through the `document` or on the video. There is a readonly attribute of the Document object `pictureInPictureEnabled`. If this is set to `false`, an error will be thrown when attempting to request Picture-in-Picture. This can be `false` because of a lack of browser support or a permissions policy. To disable Picture-in-Picture, the `video.disablePictureInPicture` attribute can be toggled by the developer (and the user using user inputs). Below is an example of toggling Picture-in-Picture.

```
const video = document.getElementById("video");
const togglePIP = document.getElementById("toggleButton");

togglePIP.addEventListener("click", (ev) => {
      video.disablePictureInPicture = !video.disablePictureInPicture;
});
```

Simple Example

The application is here: https://cs326h-picture-in-picture.herokuapp.com/
Source code is here: https://github.com/eyalJackman/cs326h-jackman-pip

This example consists of four parts: the **video**, the **toggle button**, the **disable button,** and the **resize** event listener.

Video and Buttons

The `<video />` element is the basis for the Picture-in-Picture element. In the source code, it is represented like this:

```
<video
        id="blender"
        src="http://commondatastorage.googleapis.com/
gtv-videos-bucket/sample/ElephantsDream.mp4"
```

```
        type="video/webm"
        width="512px"
        height="288px"
        controls
/>
```

The two buttons are underneath. These will be used to toggle Picture-in-Picture and enable/disable Picture-in-Picture, respectively.

```
<button id="toggle-button">Toggle Picture-in-Picture!</button>
<button id="disable-button">Picture-in-Picture Enabled</button>
```

Toggle Picture-in-Picture Button

The following code is responsible for toggling Picture-in-Picture on and off. An event listener waits for the button to be clicked. When it is clicked, it checks whether the `document` currently has a `pictureInPictureElement`. If it does, then it will exit Picture-in-Picture mode. Otherwise, if it's `null`, it will request a Picture-in-Picture on the `video` variable. Either way, it will log whether the request was accepted or rejected/an error was thrown.

```
const video = document.getElementById("blender");
const toggle_button = document.getElementById("toggle-button");

toggle_button.addEventListener("click", (ev) => {
  if (document.pictureInPictureElement) {
    document
      .exitPictureInPicture()
      .then(() => console.log("Video now not in Picture-in-Picture"))
      .catch(console.log);
  } else {
    video
      .requestPictureInPicture()
      .then(() => console.log("Video now in Picture-in-Picture"))
      .catch(console.log);
  }
});
```

As mentioned earlier, when the document enters Picture-in-Picture, the value `document.pictureInPictureElement` is set to the video.

Toggle Button in action:
https://youtube.com/clip/Ugkx96nQIakREfXt51zkfQiTK3zy9Xms8RHB

Disable Picture-in-Picture

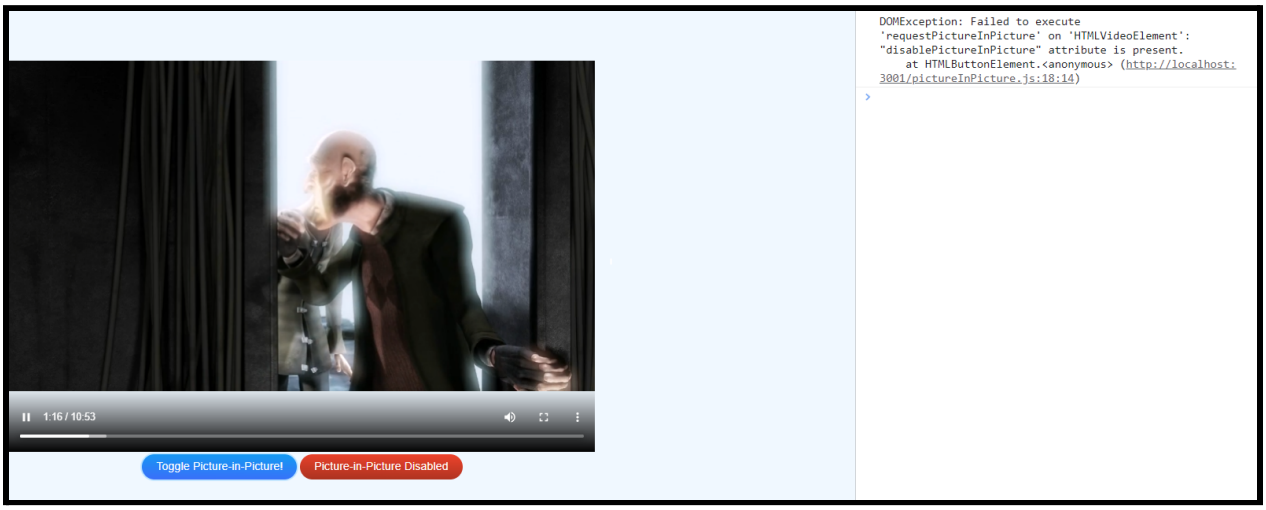
The following code is responsible for enabling/disabling Picture-in-Picture. This adds an event listener on the disable button. When clicked, if the `document` is currently in Picture-in-Picture mode, it will exit it. Then, it will toggle the `disablePictureInPicture` attribute, either allowing Picture-in-Picture or denying it. The following code example utilizes asynchronous programming.

```
const disable_button = document.getElementById("disable-button");

disable_button.addEventListener("click", async (ev) => {
  if (document.pictureInPictureElement) {
    await document.exitPictureInPicture();
  }
  video.disablePictureInPicture = !video.disablePictureInPicture;
});
```

Picture-in-Picture **Enabled**:

Picture-in-Picture **Disabled**:

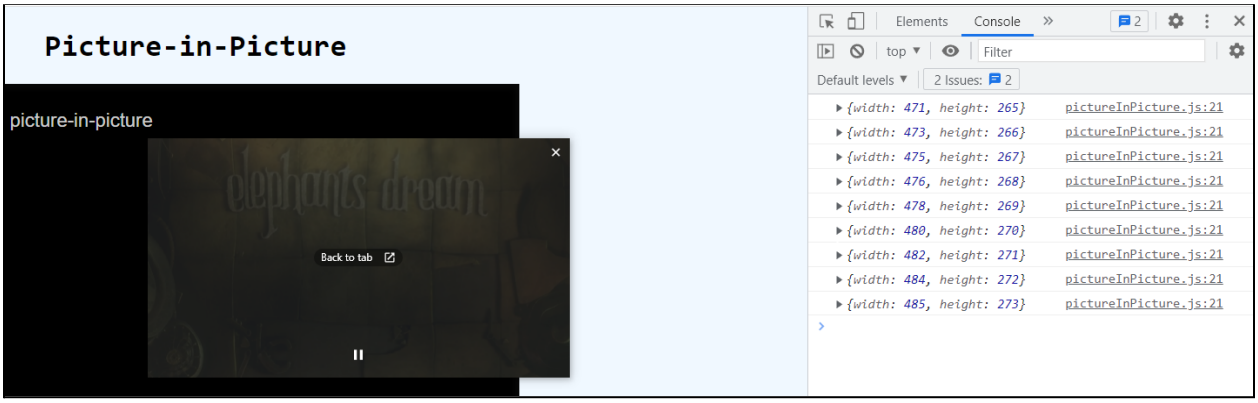


Note: When Picture-in-Picture is requested when it is disabled, a `DOMException` is thrown, preventing the video from entering Picture-in-Picture.

On Resize

The code example waits for a resize of the window and logs to the console any changes to the Picture-in-Picture window's changed dimensions. The code is below.

```
pipWindow.addEventListener("resize", (ev) => {
    console.log({ width: pipWindow.width, height: pipWindow.height});
});
```



Common Errors

`NotSupportedError`

The following error is thrown if Picture-in-Picture is not supported.

`SecurityError`

The following error is thrown if Picture-in-Picture is not allowed because of a permissions policy.

`InvalidStateError`

The following error has three potential causes:

- The video's `readyState` is set to `HAVE_NOTHING`
- The video has no video track
- The video's `disablePictureInPicture` attribute is set to `true`

Not Yet Supported Features

The following features have limited support and might gain more support in the future.

`autoPictureInPicture`

autoPictureInPicture is an attribute of the HTMLVideoElement that starts running Picture-in-Picture on the video automatically. It is implemented like this:

```
<video src="my_video.mp4" autoPictureInPicture />
```

It is currently only supported on Safari and Safari on IOS.

`picture-in-picture` CSS pseudo-class

This CSS pseudo-class triggers the CSS code when Picture-in-Picture

```
#video {
      // some stuff
}
#video:picture-in-picture {
      // some stuff while in pip
}
```

This is currently supported by no browsers.

Resources

Overview

Mozilla Picture-in-Picture API
CSS-Tricks
W3 Specification

- Provides the web standard explanation of the API

Object Documentation

HTMLVideoElement

- Provides documentation for the HTMLVideoElement Object

PictureInPictureWindow Object

- Provides documentation for the PictureInPictureWindow Object

Document Object

- Provides documentation for the Document Object

Examples

Heroku Example with Source Code

- This is the example program I wrote with the GitHub source code attached

Examples of The API

- GreenRoots's application of the different aspects of the API

Video: Toggle Picture-in-Picture

- This video clip shows Picture-in-Picture being toggled

Video: Live Video as Picture-in-Picture

- This video clip shows an interesting use of the Picture-in-Picture API

Other

CSS Pseudo-Class