PiP on any element notes:

* There’s already a version of PiP Web API, tho its too limited and cannot be too much customizable.
* The implementation will be on a full Document on aways-on-top window.
* This new window is a blank page opened via [open()](https://html.spec.whatwg.org/multipage/nav-history-apis.html#dom-open) method on [Window](https://html.spec.whatwg.org/multipage/nav-history-apis.html#window), with some differences:
  + The PiP window will float on top of any other window.
  + The PiPwindow will never outlive the opening window.
  + The website cannot set the position of the PiP window.
  + The PiP window cannot be navigated (any window.history or window.location calls that change to a new document will close the PiP window).
* The API is limited to [Secure-Contexts](https://w3c.github.io/webappsec-secure-contexts/).
* Spoofing:
  + [moveTo()](https://drafts.csswg.org/cssom-view-1/#dom-window-moveto) and [moveBy()](https://drafts.csswg.org/cssom-view-1/#dom-window-moveby) APIs must be disabled for document PiP windows.
  + Origin visibility is essential, for example, the user agent can display it on the window titlebar.
  + The maximum size needs to be restrict to prevent the website from covering the screen with an aways-on-top window and locking the user on that.

More technical definitions:

* Can pass pip properties on [documentPictureInPicture.requestWindow()](https://wicg.github.io/document-picture-in-picture/#documentpictureinpicture);
* Add event listeners to make pipWindow to send video back to main window when it closes:

pipWindow.**addEventListener**('pagehide', onLeavePiP.**bind**(pipWindow), { once: true });

* Its possible to access any elements on the pip video, just using the pip object:

**const** video = pipWindow.document.**querySelector**('#video');

video.loop = true;

* Its viable to customize buttons and controls that needs to respond to user input events, using the PiP document, as an example:

**const** pipDocument = pipWindow.document;

**const** video = pipDocument.**querySelector**('#video');

**const** muteButton = pipDocument.document.**createElement**('button');

muteButton.textContent = 'Toggle mute';

muteButton.**addEventListener**('click', () => {

video.muted = !video.muted;

});

pipDocument.body.**append**(muteButton);

* It’s possible to focus the opener tab from a PiP window:

**const** returnToTabButton = pipWindow.document.**createElement**('button');

returnToTabButton.textContent = 'Return to opener tab';

returnToTabButton.**addEventListener**('click', () => {

window.**focus**();

});

pipWindow.document.body.**append**(returnToTabButton);

* For a more friendly interface human-computer, the PiP can open on a position and size already used by the user in a past section, by setting a [documentPictureInPicture.requestWindow()](https://wicg.github.io/document-picture-in-picture/#documentpictureinpicture) attribute:

**await** documentPictureInPicture.**requestWindow**({

preferInitialWindowPlacement: true

});

* User agent usually display a button on the pip tab to go back to the opener, tho this button can be hidden and a new created and customized by setting [documentPictureInPicture.requestWindow()](https://wicg.github.io/document-picture-in-picture/#documentpictureinpicture) property:

**await** documentPictureInPicture.**requestWindow**({

disallowReturnToOpener: true

});