The ultimate guide to iframes

January 23, 2020 · 10 min read

The iframe element (short for inline frame) is probably among the oldest HTML tags and was introduced in 1997 with HTML 4.01 by Microsoft Internet Explorer.

Even though all modern browsers support them, many developers write endless articles advising against using them.

I believe that their bad reputation should not prevent you from relying on them. They have many legitimate uses cases.

Besides, it's not that hard to secure them, so you won't have to worry about your user's computer becoming infected.

To help you form your own opinion and sharpen your developer skills, we will cover all the essentials you should know about this controversial tag.

We'll go through most of the features the iframe element provides and talk about how you use them, as well as how iframe can be useful for overcoming some tricky situations. Finally, we'll talk about how you can secure your iframe to avoid potential vulnerabilities.

What is an iframe, and when do you use it?

Developers mainly use the iframe tag to embed another HTML document within the current one.

You may have crossed paths with it when you had to include a third-party widget (like the famous Facebook like button), a YouTube video, or an advertising section on your website.

For instance, the code below will display a 500px square with the google homepage within:

```
<iframe src="https://www.google.com/" height="500px" width="500px"></iframe>
```

Here is another example in which we display a button to tweet your web page on Twitter:

```
<iframe src="https://platform.twitter.com/widgets/tweet_button.html"
style="border: 0; width:130px; height:20px;"></iframe>
```

What you must keep in mind when thinking about an iframe is that it lets you embed an independent HTML document with its browsing context.

Thus, it will be isolated from the JavaScript and CSS of the parent. That is one of the valid purposes to use an iframe: to provide a measure of separation between your application and the iframe content.

Nonetheless, as you will see in this guide, the separation is not so perfect.

The iframe can still behave in annoying or malicious ways: triggering a popup or auto-playing videos for instance.

To illustrate how this isolation from the JavaScript and CSS is handy, let's take a look at these two situations:

In an application, the user could create emails and save them as templates. On a particular page, I needed to list them to let him preview and choose one.

But, to prevent the CSS of the current website from affecting the style of these templates, I figured out that using an iframe with the `srcdoc` attribute was the cleanest solution.

```
<iframe srcdoc="<html><body>The content you see here will never be affected by the CSS of its parent container. It supposed to be rendered in black on a white background.</body></html>"></iframe>
```

The other situation when iframes saved my life was when I had to build a WYSIWYG editor for a customer. But the thing with these editors is that you have to find a way to keep the focus and the selection when the user is clicking on all the buttons across the interface.

Because an iframe offers an isolated environment, this means that the focus or the selection is never lost when you are clicking outside of it.

By using communication events between the iframe and the parent (more on how to do so later in this article), I managed to design a powerful editor in a snap.

The attributes you need to know

To this day, there are eight attributes we can use to customize the behavior or styling of an iframe.

```
<iframe

  src="https://google.com/"    <!-- Sets the address of the document to embed
-->

  srcdoc="<p>Some html</p>"    <!-- Sets the HTML content of the page to show
-->

  height="100px"               <!-- Sets the iframe height in pixels -->

  width="100px"                <!-- Sets the iframe width in pixels -->

  name="my-iframe"             <!-- Sets the name of the iframe (mainly used
to reference the element in JavaScript -->

  allow="fullscreen"           <!-- Sets the feature policy for the iframe. -
->

  referrerpolicy="no-referrer" <!-- Set the referrer to send when fetching
the iframe content -->
```

You may find more than the ones listed above, but keep in mind that they are not supported in HTML5 anymore: `align` , `frameborder` , `longdesc` , `marginheight` , `marginwidth` and `scrolling` .

Note: By default, the iframe element has a border around it. To remove it, you can use the style attribute to set the border CSS property to none.

```
<iframe src="https://logrocket.com/" style="border: none;"></iframe>
```

iframe events and communication

Loading and errors

Because an iframe is a document, you can use most global event handlers.

When you are initiating the iframe, two of them come in handy to improve the experience, like displaying a spinner or a specific message to assist the user:

- The `load` event. It is triggered when the iframe is fully loaded. In other words, all static assets have been downloaded, and all the elements in the DOM tree have fired their load event.

- The `error` event that is triggered when the loading failed.

You can listen to them with the `onload` and `onerror` attribute respectively:

```
<iframe src="https://logrocket.com/" onload="onLoad()" onerror="onError()">
</iframe>
```

Or if you can add the listeners to your iframe programmatically.

```
// For a new iframe
const iframe = document.createElement("iframe");

iframe.onload = function() {
  console.log("The iframe is loaded");
};
iframe.onerror = function() {
  console.log("Something wrong happened");
};

iframe.src = "https://logrocket.com/";
document.body.appendChild(iframe);

// For an existing iframe
const iframe = document.querySelector('.my-iframe');

iframe.onload = function() {
  console.log("The iframe is loaded");
}
iframe.onerror = function() {
```

Communication with iframes

It is quite easy to send messages between the parent and the iframe. You have to use the `postMessage` function, which is documented here.

From the parent to the iframe

Send the message from the parent element:

```
const myiframe = document.getElementById('myIframe')

myIframe.contentWindow.postMessage('message', '*');
```

And listen to it in the iframe:

```
window.onmessage = function(event){
    if (event.data == 'message') {
        console('Message received!');
    }
};
```

From the iframe to the parent

Send the message from the iframe:

```
window.top.postMessage('reply', '*')
```

And listen to it in the parent:

```
window.onmessage = function(event){
    if (event.data == 'reply') {
        console('Reply received!');
    }
};
```

Note: Keep in mind that you can end up in some tricky situations when you need to debug something as messages are fire-and-forget (i.e., there is no real error handling).

Security

When you are using an iframe, you are mostly dealing with content coming from a third party over which you have no control.

Thus, you are increasing the risk of having a potential vulnerability in your application or simply having to deal with a bad user experience (like annoying video auto-play).

Thankfully, you can blacklist or whitelist specific features.

You have to use the `sandbox` and `allow` the attributes we discussed earlier.

> *Keep in mind that a good rule of thumb is to always grant the minimum level of capability necessary to a resource to do its job. Security experts refer to this concept as "the principle of least privilege."*

The sandbox attribute

Here is the complete list of sandboxing flags and their purposes:

Flag	Details
allow-forms	**Allows form submission.**
allow-modals	**Allows the resource to open new modal windows.**
allow-orientation-lock	**Allows the resource to lock the screen orientation.**
allow-pointer-lock	**Allows the resource to use the Pointer Lock API.**
allow-popups	**Allows the resource to open new popups or tabs.**
allow-popups-to-escape-sandbox	**Allows the resource to open new windows that will not inherit the sandboxing.**
allow-presentation	**Allows the resource to start a presentation session.**
allow-same-origin	**Allows the resource to maintain its origin.**
allow-scripts	**Allows the resource to run scripts.**
allow-top-navigation	**Allows the resource to navigate the top-level browsing context.**
allow-top-navigation-by-user-activation	**Allows the resource to navigate the top-level browsing context, but only if initiated by a user gesture.**

It is up to you to define which privileges you can grant to each iframe.

For instance, if your iframe only needs to submit forms and to open new modal windows, here is how you will configure the sandbox attribute:

```
<iframe sandbox="allow-forms allow-modals" src="https://www.something.com/">
</iframe>
```

For a situation when the sandbox attribute is configured, and one feature is not working correctly within the resource, it might be because it lacks a specific flag.

Make sure you know more about them to debug things quickly.

Also, keep in mind that using an empty sandbox attribute will fully sandbox the iframe.

This means that the JavaScript inside the iframe will not be executed, and all the privileges listed above will be restricted (like creating new windows or loading a plugin).

The empty sandbox attribute is mostly using for static content but will drastically reduce the capability required for other resources to work properly.

```
<iframe sandbox="allow-forms allow-modals" src="https://www.something.com/">
</iframe>
```

Note: The sandbox attribute is unsupported in Internet Explorer 9 and earlier.

The `allow` attribute

This `allow` attribute is currently experimental and only supported by Chromium-based browsers. It lets you allow whitelist specific features like letting the iframe access to the accelerometer, the battery information, or the camera.

There are more than 25 available flags, so I will not list them all here. You can browse them on the Mozilla Feature Policy Documentation. I summarized the most popular in the table below:

Flag	Details
accelerometer	**Allows access the Accelerometer interface**
ambient-light-sensor	**Allows access the AmbientLightSensor interface**
autoplay	**Allows to autoplay video and audio files**
battery	**Allows access to the Battery Status API**
camera	**Allows access to the camera**
fullscreen	**Allows access to fullscreen mode**
geolocation	**Allows access to the Geolocation API**
gyroscope	**Allows access to the Sensors API Gyroscope interface**
magnetometer	**Allows access to the Sensors API Magnetometer interface**
microphone	**Allows access to the device microphone**
midi	**Allows access to the Web MIDI API**
payment	**Allows access to the Payment Request API**
usb	**Allows access to the WebUSB API**
vibrate	**Allows access to the Vibration API**

Things to know about iframes

How to deal with browsers that do not support iframes

If a browser does not support an iframe, it will display the content included between the opening `<iframe>` tag and the closing `</iframe>` tag.

Thus, you should always think about placing a warning message as a fallback for those poor users.

```
<iframe>
  <p>Your browser does not support iframes.</p>
</iframe>
```

How can you render the iframe like it is actually part of the parent document (i.e., no borders and scrollbars) ?

🤓 The `seamless` attribute has been introduced for this exact purpose. It is still experimental and poorly supported among browsers (only Chromium-based understand it).

It's also not part of the W3C HTML5 specification at the time of this writing.

```
<iframe seamless src="https://logrocket.com/"></iframe>
```

Can iframes effect the SEO of my website?

I did not know much about this, so I had to dig a little. There are many speculations around this subject.

For a long time, crawlers could not understand them, but this is no longer the case. The most relevant answer I found was from this article and today's conclusion seems to be:

> *Since search engines consider the content in iframes to belong to another website, the best you can hope for is no effect. Iframes tend to neither help nor hurt your search engine ranking.*

Thus, it is best to assume that the content displayed via iframes may not be indexed or available to appear in Google's search results. A workaround would be to make sure to provide additional text-based links to the content they display so that Googlebot can crawl and index this content.

Note: You should also not worry about duplicate content issues since today's web crawlers usually recognize them.

Can iframes affect the loading speed of my website?

Every iframe on a page will increase the memory used as well as other computing resources like your bandwidth.

So, you should not use iframe excessively without monitoring what's going on, or you might end up harming your page performance.

To avoid having your iframes slow down your pages, a good technique is to lazy load them (i.e., loading them only when they are required like when the user scrolls near them).

This can be achieved easily just by adding the `loading="lazy"` attribute to the tag.

Keep in mind that all modern Chromium-based browsers support this at the time of this writing. You will be interested in the lazyload library for something that works everywhere.

```
<iframe src="https://logrocket.com/" loading="lazy"></iframe>
```

Note: The `loading="lazy"` attribute also works with the `img` tag, in case you didn't know that already. 😜

How can I make an iframe responsive?

As more people browse the web using their phones, it is essential to make sure every one of your interfaces is responsive.

But how can you do so when you have an iframe in your page?

We could dedicate an entire guide regarding the myriad of ways to make your iframe responsive. Instead, I will just link to two excellent articles:

- This first article (with probably the simplest solution) will show you how you can achieve this by wrapping your iframe in another HTML element and by adding a few CSS properties to it.
- This second article will show you how you can make an iframe responsive by dealing with aspect ratios.
- There is also the Iframe Resizer Library, but keep in mind that it comes with a lot of additional features you may not actually need.

Note: If you are using the bootstrap library in your project, there are the `embed-responsive` and `embed-responsive-16by9` that you can use straight out of the box to make your iframes responsive.

```
<div class="embed-responsive embed-responsive-16by9">
  <iframe src="https://logrocket.com/" loading="lazy"></iframe>
</div>
```

How to prevent the white flash that occurs while the iframe is loading

Yes my friends, there is a solution for this. In this article, Chris Coyier shares a little snippet that hides all the iframes on the page with some CSS and removes it until the window is loaded and then makes them visible.

How to reload the content of an iframe

Easy peasy lemon squeezy! As you can access the window element of the iframe with `contentWindow`, you have to do this:

```
// Get the iframe
const iframe = document.getElementById('myIframe');

// Reload the iframe
iframe.contentWindow.location.reload();
```

Bonus: about iframe accessibility

Most of the time, a screen reader will indicate that an iframe is present on the page.

Many will even enable you to navigate inside. If your iframe does not contain another webpage but instead contains external content like a video player or an ad, it is essential to add a title to the tag.

Adding a title to the tag gives users more context for what is the iframe is about.

```
html
<iframe src="an_ad.html" title="I contain an advertisement"></iframe>
```

Some will say that this attribute is not essential since most screen readers will only read the document's title when available and skip the title attribute on the iframe tag.

However, for the same title to be read correctly across various screen readers, it remains good practice to provide both and to make sure they match each other.

Another thing to keep in mind is that when your iframe includes non-readable content (like, for instance, a video game built with JavaScript), you will have to hide its contents from screen reader users using the aria-hidden attribute.

```
html
<iframe src="a_video_game.com" title="I contain a video game" aria-
hidden="true"></iframe>
```

We talked about this at the beginning of this guide, but make sure to include some content inside the iframe for all the older browsers that do not support them.

This will give more context to everyone about what should be displayed in the space.

```
html
<iframe>
  <p>Your browser does not support iframes.</p>
</iframe>
```

LogRocket: Monitor iframe performance and how users interact with them

LogRocket is a frontend application monitoring solution that lets you replay problems as if they happened in your own browser. When implementing iframes, it's critical that you understand how they impact the performance of the overall

app or page as well as the user experience. Instead of guessing why errors happen, or asking users for screenshots and log dumps, LogRocket lets you replay the session to quickly understand what went wrong. It works perfectly with any app, regardless of framework, and has plugins to log additional context from Redux, Vuex, and @ngrx/store.

In addition to logging Redux actions and state, LogRocket records console logs, JavaScript errors, stacktraces, network requests/responses with headers + bodies, browser metadata, and custom logs. It also instruments the DOM to record the HTML and CSS on the page, recreating pixel-perfect videos of even the most complex single-page apps.

Try it for free.

Conclusion

I hope this guide has helped you to improve your knowledge of iframes.

While they can be insecure if you're loading untrusted content, they also offer some significant advantages. So you should not ban them entirely from your developing arsenal but only use them in relevant situations.

If you have anything to add to this article, you can reach me in the comments below or just ping me on Twitter @RifkiNada

Share this:

Twitter Reddit LinkedIn Facebook

Nada Rifki Follow

Nada is a JavaScript developer who likes to play with UI components to create interfaces with great UX. She specialises in Vue.js and loves sharing anything and everything that could help her fellow front-end web developers. Nada also dabbles in digital marketing, dance and Chinese.

31 Replies to “The ultimate guide to iframes”

Serg Says:
January 25, 2020 at 10:21 am

Reply

Seamless attribute was removed from both W3C and WHATWG hrml specs, and implementation was removed from the browsers due to the security and other reasons https://caniuse.com/#feat=iframe-seamless

Ariona Rian Says:
January 28, 2020 at 5:55 pm

Reply

Missing guide: how to make iframe height automatically use it’s content height 🤔

Nada Rifki Says:
January 29, 2020 at 6:18 am

Reply

Thank you for the information, I thought it was only experimental for the time being.

Nada Rifki Says:
January 29, 2020 at 6:19 am

Reply

Hello Ariona, what do you mean exactly? 🙂

Piotr Says:
February 14, 2020 at 5:44 am

Reply

Nada, great art. Thanks. Just remove unnecessary “)” from postMessage link.

Matt Angelosanto Says:
February 14, 2020 at 8:23 am

Reply

Nice catch. All set.

Balexandre Says:
February 18, 2020 at 7:12 am

Reply

You talked nothing about “friendly iframe” when the iframe source is and that we do not need to send message events as those iframes can get all functionality from the parent window object…

BriarRose Says:
February 18, 2020 at 8:07 am

Reply

Thank you, Nada, for this informative article! A helpful addition would be addressing accessibility issues with iframes.

Adrian Roselli Says:
February 18, 2020 at 4:57 pm

Reply

There is a lot to be said about WCAG conformance an iframes. Technique H64: Using the title attribute of the frame and iframe elements (https://www.w3.org/TR/WCAG20-TECHS/H64.html) addresses how this applies to WCAG Success Criteria 2.4.1 Bypass Blocks (level A) and 4.1.2 Name, Role, Value (level A). While conformance is more nuanced than a pass/fail, at least acknowledging the challenge and risk of being flagged by automated accessibility testing tools should be in this article.

Tanya Says:
April 3, 2020 at 2:45 pm

Reply

I have a question.

I'm using an iframe to load a media file. Click the link to the file and it loads into the named iframe.

My link has a target="media" attribute on it and the first video loads fine. However when I try to load the second video it will not load in the iframe, it loads in a new tab.

Raj Says:
May 7, 2020 at 9:00 pm

Reply

Just wondering how you got the google.com example to work. I get a "Blocked by X-Frame -Options Policy" for google, and "Blocked by Content Security Policy" for some other domains that I tried, But the Twitter button and the Logrocket examples work fine. I am using localhost/ 127.0.0.1 for testing, which might be the issue.

Don Says:
June 9, 2020 at 6:00 am

Reply

I have a lot of content on my site, and I recently implemented the use of (loading="lazy") to defer the loading of an iframe until the viewer opens the window. I noticed that it did not interfere with the laoding of the iframe content, however, the code was flagged as "not allowed" when validating my html. I also noticed that when using speed testing sites such as Google insights and GTMetrics, the browser is seeing and loading the content within the iframe even though I am using the lazy loading tag in the iframe?

Nada Rifki (@RifkiNada) Says:
June 10, 2020 at 8:22 am

Reply

Hi Don, can you tell me which browser (and version) you're using ?

Arnaud Says:
June 10, 2020 at 2:12 pm

Reply

Hi, thanks for this guide, really interesting. You wrote "Because an iframe offers an isolated environment, this means that the focus or the selection is never lost when you are clicking outside of it."
I am actually facing a situation where my iFrame is losing its focus when clicking elsewhere… Any way to prevent that ?

Roy Hajj Says:
July 24, 2020 at 9:41 am

Reply

Hi, thanks for this comprehensive iframe documentation , But frankly , I was looking for an answer to a difficult question , a question that no one online , even the Top Programming websites , could answer it till now ..
This question is , how can we play a video game inside iframe , without the keyup or keydown forcing the whole page to scroll up or down , instead of focusing inside the iframe ?

Nada Rifki (@RifkiNada) Says:
July 28, 2020 at 3:34 pm

Reply

Why don't you simply disable the scroll when people are using the keyboard arrows? Something like https://stackoverflow.com/questions/8916620/disable-arrow-key-scrolling-in-users-browser

Nada Rifki (@RifkiNada) Says:
July 28, 2020 at 3:34 pm

Reply

If you want to check when an iframe is not focused anymore, you can check this thread https://stackoverflow.com/questions/5456239/detecting-when-an-iframe-gets-or-loses-focus Then, you only have to refocus the iframe with javascript when it happens

Horison Says:
August 1, 2020 at 9:44 am

Reply

Hi,
Thanks for this excellent information with unique content and it is very useful to know about the information based on comprehensive iframe documentation.

Julian Kandler Says:
August 11, 2020 at 2:57 am

Reply

Hi,

Thanks for your guide! I tried to use it for error handling if the iFrame could not be loaded, but this does not seem to be possible, since the onerror event never gets triggered. Ans it also seems like (at least for Chrome), they are not going to fix that: https://bugs.chromium.org/p/chromium/issues/detail?id=365457

Is there something im missing?

D Devid Says:
August 23, 2020 at 3:04 pm

Reply

Thank you, Nada, for this informative article! A helpful addition would be addressing accessibility issues with iframes.

Sankar Says:
August 25, 2020 at 10:43 am

Reply

Any idea how to use IFRAME in windows application?

Joe Hajj Says:
August 29, 2020 at 6:23 pm

Reply

Yes , it worked that way . Thanks again .

Michael Keuntje Says:

Reply

September 15, 2020 at 4:03 am

thx Nada – I really like the overview quality of your article.
if I integrate content via iFrame into a WP page is there a way I can avoid thrd parties to open the iFrame content without opening the complete page?

Madan Says:
September 24, 2020 at 9:54 am

Reply

Can you please tell me how we can display a PDF file inside the iframe html element without download & print option in the frame window?

Chad Says:
September 24, 2020 at 5:50 pm

Reply

For some you can't use external resources from iframe using a development server, especially when trying to fetch resources from Google. As a security measure google doesn't allow that… hence a development server with a "localhost" in it's domain is automatically blocked. There is a way to trick the system by changing your "localhost" domain or using https, I read somewhere tho I never actually tried it before just a friendly coder who stumbled on this article to learn more about iFrames.

Rick H Says:
October 4, 2020 at 3:40 pm

Reply

Hello. Great post. I had a question about using iframe and jump links together. I am building a recipe site with a list of links to recipes from different sites. When the links are clicked the recipes are displayed within the iframe window at the top of the same page. I wanted to add jump links so the user to taken to the top of the page where the recipe is being displayed. Any suggestions?

PatrickB Says:
December 14, 2020 at 1:20 am

Reply

Great Article Nada – well done. I came to the sight to get some information on how cookies work within an iframe. I'm using .Net Core 2.2. and application (session) cookies are not being recognized in the same way as if the application runs outside the iframe. Any insight on cookie limitations and using sameSite=None/Lax/Strict (etc), and the meaning of these would be a great add to your article.

Cheers!

Montrell Hale Says:
January 10, 2021 at 8:50 am

Reply

im trying to add a iframe into elementor on wordpress and i cannot figure out to adjust the height, there is also a rule for scrolling in a an iframe i didnt see that here.

John Perry Says:
January 22, 2021 at 5:11 am

Reply

I wrote a set of pages a while ago. In fact last changed 2011! Now I am trying to resurrect them. But they were built around frameset(s) and frame(s) – I did write a kind of "file explorer". I can work out what to change. But the main problems is how to move the iframe to be along the right, e.g. width=85% but I can't see (in several different pages) how to have the iframe set right. I've seen talk of attribute object-position, but I can't how to work it.

Help!

Raziya Says:
February 1, 2021 at 12:35 am

Reply

nice guide. got the info, what im looking for.

John Perry Says:
February 3, 2021 at 10:40 am

Reply

I've done all I want, almost...(I worked out the rightside by using tables).

Something like file explorer but with my photos across the world.

I insert one (one of a number of pages) into the right side. On its own it's fine, but inside an iframe, it does allow js. OK with variable but not OK with fumctions. I guess this tweaking some kind of attribute I need to use, but I can't hack it.
Help
——

All I have to do after that is visibility:collapse

And add or the places I travelled – mostly China / SE Asiia; and Europe. About 50000 photos, only about 1000 for this album.

Leave a Reply

Enter your comment here...