



hreflang: Risks and maintenance

In this lesson we'll tell you all about the risks you should be aware of when implementing and maintaining hreflang. First, we'll discuss risks related to initial hreflang implementation. Then, we'll talk to you about risks having to do with maintaining and updating your hreflang implementation. Finally, we'll list a few convenient tools for creating and testing hreflang tags.

hreflang implementation risks

What risks should you be aware of when implementing and maintaining hreflang? We can distinguish two types of risks:

1. Risks having to do with the initial hreflang implementation.
2. Risks having to do with maintaining and updating your hreflang implementation. In other words: a changing situation, which you have to account for.

After this course, you should know how hreflang works and how to implement it using the optimal method for your situation. In other words, you should have the first risks covered. However, the real problems start with updating and maintaining your site.

Risk 1: Initial hreflang implementation

Relative URLs

Relative URLs are all URLs that do not contain a fully qualified domain name and path, but instead just the path or a portion of the path. These URLs don't explicitly specify the protocol ("http://" or "https://") and/or domain ("www.example.com"). So when your website is *example.com*, you could be linking to your contact page from your homepage like this:

```
<a href="contact.html">Contact</a>
```



For example, an hreflang implementation with relative URLs will look like this:

```
<link rel="canonical" href="/us/lego" />
<link rel="alternate" href="/gb/lego" hreflang="en-gb" />
<link rel="alternate" href="/us/lego" hreflang="en-us" />
```

However, we do not recommend using relative URLs. Every link in an hreflang implementation should be the full, absolute URL. Using relative URLs is not strictly against the link element specification, but it makes it much harder for the search engines to interpret the hreflang implementation, so using relative URLs is a sure recipe for breakage. There is no margin for error with hreflang tags, so make sure to play it safe.

Language and country/regional codes

We've stressed the importance of getting the language and country/regional codes right throughout this course. We'll say it once more: if the codes don't comply with the ISO standards, your implementation will break!

Common mistakes include:

Faulty code	Explanation
en-en	The second en is not a regional code: there is no 'English' variation of English. This example is applicable to many other languages as well.
en-uk	uk is not a regional code: gb is. Make sure to check the official list.
gb-en	You cannot switch around the language and country / regional codes

Return links

It's simple: each and every page in the hreflang implementation needs to link to each and every page in the hreflang implementation.

Self links

Again, it's simple: each and every page in the hreflang implementation needs to link to itself.

Canonical interference

Each page should have a canonical pointing to itself and nowhere else.

Double implementations

This is the really obvious one, but we've seen it go wrong: don't implement hreflang in both the XML sitemap and the `<head>`. Google won't know which one to follow and may randomly pick one.

Risk 2: Maintaining and updating your implementation

Maintenance

You really need to be aware of the structure of your site and the underlying hreflang implementations. Once something in that set-up changes, the hreflang implementation should naturally be updated. In bigger implementations, it can be quite a hassle to stay on top of this. This is compounded by the fact that hreflang easily breaks when a small portion of the implementation is not current, although Google deals with this better than it used to. Therefore, it's imperative that you map out your implementations and consider any change that may break the chain of hreflang implementations. **This is probably the biggest challenge in your hreflang strategy!**

Redirected or broken URLs

One of the most common problems with hreflang is a lack of focus on redirected or broken URLs. You should account for each and every change you make if it pertains to your hreflang implementation. If the links don't work, the underlying hreflang implementation will break.

Make sure that people in your company who deal with content on your site know about hreflang. This makes sure they won't do things that break your implementation. Two things are very important:

1. When a page is deleted, check whether its counterparts are updated.
2. When a page is redirected, change the hreflang URLs on its counterparts.

If you do that and audit regularly, you shouldn't run into too many issues.

hreflang tools

In the video, Joost showed a couple of handy tools that are available for creating and testing hreflang tags. We'll list these tools below, followed by a brief description of their functionalities:

- [**The hreflang Tags Generator Tool**](#): this tool allows you to create hreflang tags for any URL. You enter the URL and specify which country and language it is for, and the tool will generate hreflang tags you can copy straight into your pages.
- [**hreflang Testing Tool**](#): you can enter a URL into this tool, and it will check the hreflang tags on that page and check if the return links are correct.
- [**hreflang Tags Testing Tool**](#): this tool is similar to the previous tool, but it'll give you more specific information. For example, it'll give you information about a missing default language tag.
- [**hreflang Tag Checker \(Chrome extension\)**](#): if you install this tool and go to a web page, it'll check whether there's an hreflang tag and check the back references.
- [**hreflang Sitemap Tool**](#): this tool helps you create an XML sitemap with hreflang implementation. Like Joost showed in the video, you can upload a CSV file, and it will email you an XML sitemap with a proper hreflang implementation.

Module 1.1	Module 1.2	Module 2.1	Module 2.2	Module 3.1	Module 3.2
Module 3.3	Module 4.1	Module 4.2	Module 4.3	Module 4.4	