hreflang implementation elements

In this lesson we'll teach you all about the different implementation elements of hreflang. We'll discuss self-referencing canonicals, return and self links and X-default. After that, we'll show you what a complete hreflang implementation looks like.

hreflang tag vs. hreflang implementation

We've seen what an hreflang tag looks like:

```
1. link rel  =  "alternate"
2. href      =  "http://example.com/"
3. hreflang  =  "en"
```

However, one hreflang tag does not make an hreflang implementation. For now, we'll explain how it works in the easiest implementation method: implementing hreflang in the `<head>` of your site. Later in this module, we'll contrast the other methods to the basic explanation.

A hreflang `<head>` implementation consists of the following elements:

1. an hreflang canonical
2. a self link
3. return links
4. optionally, an X-default link

On the British page of an hreflang implementation, the complete implementation may look something like this:

```
<link rel="canonical" href="http://example.com/gb/lego" />
<link rel="alternate" href="http://example.com/gb/lego" hreflang="en-gb" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="en-us" />
<link rel="alternate" href="http://example.com/fr/lego" hreflang="fr" />
<link rel="alternate" href="http://example.com/nl/lego" hreflang="nl" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="x-default" />
```

Now, let's explore all of these elements one by one.

hreflang: Canonical

As we mentioned at the beginning of this lesson, the list of hreflang attributes is not the only part of the hreflang implementation. Each page in a `<head>` implementation should also feature a self-referencing canonical.

Let's quickly review what the canonical link element does. It lets the search engines know which variation of a collection of similar or identical pages you want Google to index. Every page you want Google to index should feature a self-referencing canonical.

If a page is the less important variation of another page, you should implement a canonical link to the more important page. Google then passes on the link value of that page to the canonical page. The canonical page is the page you want Google to index.

A canonical consists of two elements: the `link rel=`, which specifies that we're dealing with a canonical link, and the `href`, which points to the canonical page.

```
<link rel="canonical" href="http://example.com/es" />
```

In an hreflang implementation, there is one big difference: you want Google to index every page variation. This means that every page in your hreflang implementation should feature a canonical link pointing to itself. By doing this, you tell Google that you want the English page to be indexed for your English audience, the Spanish for your Spanish audience, etc.

The href of the self-referencing canonical should be identical to that of the self link:

Canonical: `<link rel="canonical" href="http://example.com/es" />`
Self link: `<link rel="alternate" href="http://example.com/es" hreflang="es" />`

This is the only part of the hreflang implementation that is different for each page, because the canonical on each page should point to itself and not to the other pages.

hreflang: Return and self links

We've already established that an hreflang implementation consists of a list of hreflang attributes. These hreflang attributes specify all the available variations of a page. In addition, we mentioned that one of the hreflang attributes in an implementation always has to point to itself. The first links are called return links, the latter are aptly called self links. **Don't worry: the end conclusion of all this will be that you can simply use the same list of hreflang attributes for each page.** To understand why, however, we need to go through the details first. A good understanding of the workings of hreflang helps you update and maintain your implementations properly.

Return links

Every hreflang attribute that links to another page is a return link. This is because the hreflang implementation is only valid if each page in the implementation points to all other pages in the implementation. Consequently, if page A links to page B, page B needs to link back to page A, hence the name 'return link'.

Image 1: hreflang attribute that links to another page

In a simple implementation, this will lead to the situation detailed in image 2: each page links to the other pages.

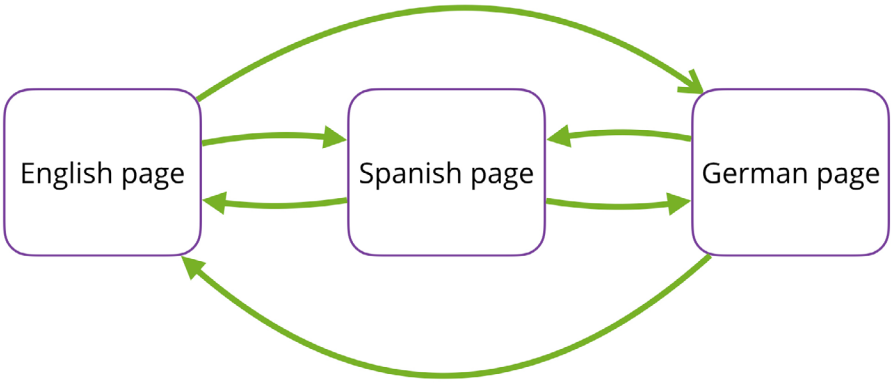

Image 2: Return links

In the code, it looks something like this:

Site A: English	<title>example.com</title>
	<link rel="canonical" href="http://example.com/" />
	<link rel="alternate" href="http://example.com/" hreflang="en" />
A → B	**<link rel="alternate" href="http://example.com/es/" hreflang="es" />**
	<link rel="alternate" href="http://example.com/de/" hreflang="de" />

Image 3: English page linking to Spanish page:

Site B: Spanish	<title>example.com in Spanish</title>
	<link rel="canonical" href="http://example.com/es/" />
B → A	**<link rel="alternate" href="http://example.com/" hreflang="en" />**
	<link rel="alternate" href="http://example.com/es/" hreflang="es" />
	<link rel="alternate" href="http://example.com/de/" hreflang="de" />

Image 4: Spanish page linking back to English page

Self links

In addition to return links, each page needs to link to itself. Confusingly, you still need to use the `rel="alternate"` attribute for this. So a self link on the Spanish page would look like this:

```
<link rel="alternate" href="http://example.com/es" hreflang="es" />
```

> *One hreflang tag does not make an hreflang implementation.*

Or, embedded in an implementation:

Site B: Spanish	<title>example.com in Spanish</title>
	<link rel="canonical" href="http://example.com/es/" /> <link rel="alternate" href="http://example.com/" hreflang="en" />
B ⟶ B	**<link rel="alternate" href="http://example.com/es/" hreflang="es" />**
	<link rel="alternate" href="http://example.com/de/" hreflang="de" />

Image 5: Self link on the Spanish page

If we add the self links to the return links, we arrive at the following complete implementation:

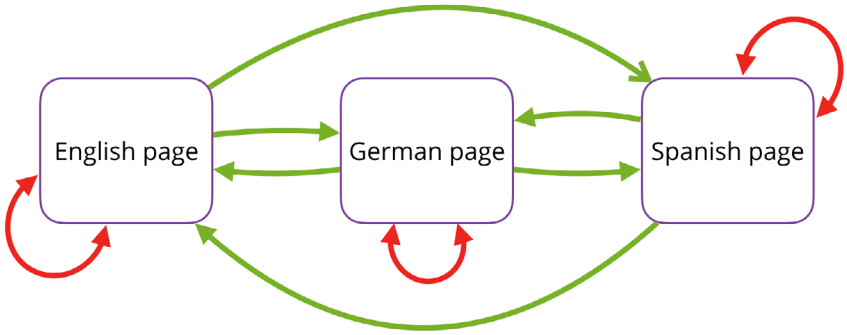

Image 6: Representation of valid hreflang implementation using return links and self links

Each page links to all other pages and to itself. Therefore, simply re-using the same list of hreflang attributes for each pages suffices. Just look back at the example images we provided. The list of hreflang attributes is always the exact same. As we'll explore below, only the canonical links are different. In theory, that's quite simple. However, maintaining this can be quite a hassle. That is why it's important for you to understand the underlying principles.

hreflang: X-default

There's only one hreflang implementation option we haven't discussed yet: the X-default. What happens when there is no viable alternative in the hreflang implementation? If you don't account for this, Google won't know where to send the user.

Let's consider the following situation. A Brazilian user wants to visit the Lego article we discussed earlier. His browser states that he accepts Portuguese and Spanish, but the hreflang implementation features neither language. Where should Google send him?

You can solve this problem for Google by adding an X-default to your hreflang implementation. In an X-default, you simply specify `"x-default"` as the value of the hreflang name-value pair:

```
<link rel="alternate" href="http://example.com/us/lego" hreflang="x-default" />
```

The X-default tells Google what page it should send the user to if none of the examples fit the user. In the example above, Google will show the Brazilian user http://example.com/us/lego, which the implementer apparently considers to most likely be the best result.

An alternative approach could be to create a special page in which you allow the user to choose the language. Doing this requires nothing more than linking to the dedicated page as the X-default:

```
<link rel="alternate" href="http://example.com/lang/lego" hreflang="x-default" />
```

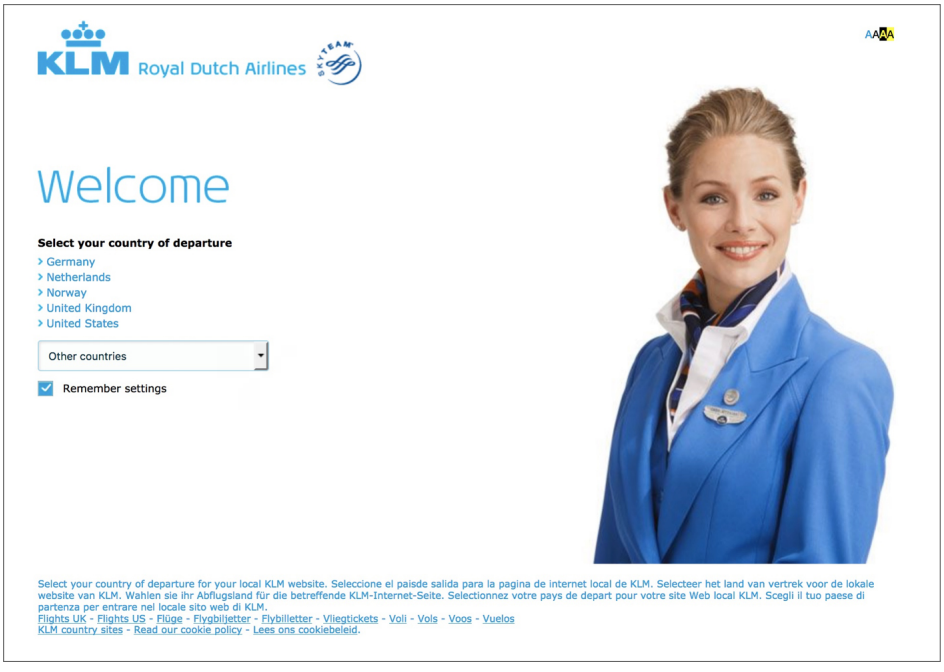

Image 6: KLM language landing page

Architectural implementation choices

One thing is very important when implementing hreflang: don't be too specific! Let's say you have three types of pages:

- German
- German, specifically aimed at Austria
- German, specifically aimed at Switzerland

You could choose to implement them using three hreflang attributes like this:

- `de-de` targeting German speakers in Germany
- `de-at` targeting German speakers in Austria
- `de-ch` targeting German speakers in Switzerland

However, which of these three results should Google show to someone searching in German in Belgium? The first page would probably be the best. To make sure that every German searching user who does not match either `de-at` or `de-ch` gets that one, change that hreflang attribute to just `de`. Specifying just the language is a smart thing to do in many cases.

It's good to know that when you create sets of links like this, the most specific one wins. The order in which the search engines sees the links doesn't matter, it'll always try to match from most specific to least specific.

What does a complete hreflang implementation look like?

So let's see what a complete hreflang implementation may look like for the Lego article. For now, we will stick to what it would look like in the header of your page. As we will see in the next lesson, your method of implementation influences what the hreflang implementation will look like.

First, we specify the canonical on each page. Then, we add the list of return and self links and the x-default. As we have seen, this list is always the same. Only the highlighted canonical is different for each page.

Page 1: English GB

```
<link rel="canonical" href="http://example.com/gb/lego" />
<link rel="alternate" href="http://example.com/gb/lego" hreflang="en-gb" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="en-us" />
<link rel="alternate" href="http://example.com/fr/lego" hreflang="fr" />
<link rel="alternate" href="http://example.com/nl/lego" hreflang="nl" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="x-default" />
```

Page 2: English US

```
<link rel="canonical" href="http://example.com/us/lego" />
<link rel="alternate" href="http://example.com/gb/lego" hreflang="en-gb" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="en-us" />
<link rel="alternate" href="http://example.com/fr/lego" hreflang="fr" />
<link rel="alternate" href="http://example.com/nl/lego" hreflang="nl" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="x-default" />
```

Page 3: French

```
<link rel="canonical" href="http://example.com/fr/lego" />
<link rel="alternate" href="http://example.com/gb/lego" hreflang="en-gb" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="en-us" />
<link rel="alternate" href="http://example.com/fr/lego" hreflang="fr" />
<link rel="alternate" href="http://example.com/nl/lego" hreflang="nl" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="x-default" />
```

Page 4: German

```
<link rel="canonical" href="http://example.com/de/lego" />
<link rel="alternate" href="http://example.com/gb/lego" hreflang="en-gb" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="en-us" />
<link rel="alternate" href="http://example.com/fr/lego" hreflang="fr" />
<link rel="alternate" href="http://example.com/nl/lego" hreflang="nl" />
<link rel="alternate" href="http://example.com/us/lego" hreflang="x-default" />
```