Paginator | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/elements/paginator.html> |
| Date de scraping | 2025-04-08T15:54:14.872441+00:00 |

## Paginator ​

In this article, we explain how to:  
Style the Paginator element Filter a collection in the frontend using the Paginator element Filter a collection in the backend using the Paginator element

## Styling the Paginator ​

You can style any element inside the Paginator, change the style of the arrows or change the color of the active page number for example.  
To change the color of the active page number:  
select the "Text" element inside the Paginator, create an active state in the style section of this element, select the active state and style it like you would any other state.  
In the example below, we styled the Paginator's "Text" element so that the active page number appears blue and bold:

## Frontend pagination ​

Before we go any further, note that when you add pagination in the frontend, all your collection items loaded in the user's browser, even if only 10 items are displayed on the page.  
If you do not want this to happen, consider adding pagination in your backend before fetching the collection in WeWeb.  
Learn more about server-side (backend) vs client-side (frontend) pagination here .  
If you do want to add pagination in the frontend, here's how you do it in WeWeb:  
drag-and-drop the Paginator element on the page that displays the collection you want to paginate ensure the collection on the page is bound to the Collection object and not a Collection['data'] array tell the Paginator which collection you want to paginate using the Collection dropdown:  
  
To learn more about frontend pagination in the context of a real project, consider watching this video from level 1 of the WeWeb Academy:

### Images

<https://docs.weweb.io/assets/paginator1.DwqEVTSx.jpg>

## The limits of frontend pagination ​

With frontend pagination, data that is not displayed on the page is still accessible in the browser:  
This has two consequences:  
the data being loaded may still be heavy enough to slow the browser down or strain your visitors' data plans even though it is not displayed on the page, the data is accessible to anyone who knows where to look for it, i.e. in the inspector / network.  
If you want to avoid this, consider adding pagination in the backend instead.

## Backend pagination ​

When working with large amounts of data, it is best practice to paginate your Collection of data in the backend.  
If you paginate a Collection in the frontend, the entire set of data will be loaded in your users' browsers, even if you only display 10 items on the page.  
This is a big no-no if you're working with a heavy set of data as your users' browsers will crash.  
People often talk about millions of records but another way to look at it is in terms of volume. The less you tap into your users mobile data plans, the better.  
This next steps explain how to setup custom pagination on the Paginator Element.

## Step 1: add paging to the backend ​

In Xano, you'll need to enable Paging in the Output of your query:  
If you run and debug your API call, you should now see additional information in Xano like the number of itemsReceived and total number of pages, i.e. pageTotal :  
  
In WeWeb, the collection should look something like this with context about the collection in addition to the items :  
  
Note that, in the screenshot above, we can see Xano sends back 5 items. This is because we asked for 5 items per page when we enabled Paging in Xano.

### Images

<https://docs.weweb.io/assets/paginator-back1.ww-alYYt.jpg>

<https://docs.weweb.io/assets/paginator-back3.CGgB7abr.png>

## Step 2: have the backend listen for info about the current page and offset ​

With the WeWeb Paginator element, we can send info about the current page and offset to tell Xano what items the user needs to see next. In order to to feed that information to Xano, we will create two integer inputs in Xano – page and offset – which we'll map in the External tab:

### Images

<https://docs.weweb.io/assets/paginator-back2.CtOLl5UY.png>

## Step 3: test the API call ​

Through a "Run & Debug" in Xano and/or in a Collection in WeWeb, test your API call to make sure it sends back the data you expect depending on the page or offset you send.  
Here we chose to tell Xano which page we want:

### Images

<https://docs.weweb.io/assets/paginator-back7.BOFmDtbD.png>

## Step 4: create a paginator variable ​

Create a paginator variable of type Object with the default values your backend needs to send the first items you want to display from your table.  
Usually, the default page value should be 1 – paginator is on the first page – and the default offset value should be 0 – no items should be skipped:

### Images

<https://docs.weweb.io/assets/paginator-back6.CkfMeKnX.png>

## Step 5: Paginator workflow ​

On the page with the collection you want to paginate, add the WeWeb paginator element with a workflow that:  
triggers On change updates the paginator object variable with the information from the event refetches the Collection you are working with  
  
That way your paginator variable will have the up to date offset and current page so however you chose to make the call to Xano, you’ll have the correct info based on user interactions with the paginator.

### Images

<https://docs.weweb.io/assets/paginator-back8.B1K2_w1B.png>

## Step 6: Paginator settings ​

On the paginator element, enable "Custom pagination", and bind:  
the total items returned by your backend (e.g. Xano sends back itemsTotal ), the fixed number of items per page (note: this is NOT the number of items received because if you’re on the last page of the collection, the backend might return only 17 items leftover instead of the full 25 per page for example) the offset – the offset should be current page (from the paginator event) - 1 \* number of items per page

### Images

<https://docs.weweb.io/assets/paginator-back9.DThTAEoh.png>

## Step 7: bind the collection ​

In WeWeb, instead of binding your collection to the collection with its metadata, you'll need to bind directly to an array or list of items.  
If we take Xano as an example, that means instead of binding to Collection['data'] or Collection , you'll need to bind Collection['data']['items'] :  
  
This is because the container to which you bind the collection expects an Array, i.e. a list of items.  
The binding of the Collection List items is the same as with any other Collection List. If you're unsure how to display the job title or company logo on the page, please refer to the user documentation on binding .

### Images

<https://docs.weweb.io/assets/paginator-back4.B1omv2j7.png>

## Step 8: make your API call dynamic ​

Back in your collection, you can now bind the input you are sending to your backend to the data in your paginator variable (for example, the page or offset).  
The input you send to your backend will depend on the info your backend needs to return the items you want.  
And voilà! You should be good to go.  
Enable the Preview mode to check that everything is working correctly.  
Setting up backend pagination can be challenging. Please don't hesitate to reach out to the WeWeb Community if you get stuck.