**Angular 9 - Simple Pagination Example**

Example built with **Angular 9.1.11**

Other versions available:

* **Angular:**[Angular 10](https://jasonwatmore.com/post/2020/10/03/angular-10-simple-pagination-example), [8](https://jasonwatmore.com/post/2019/06/18/angular-8-simple-pagination-example), [2/5](https://jasonwatmore.com/post/2016/08/23/angular-2-pagination-example-with-logic-like-google), [Angular + Node](https://jasonwatmore.com/post/2019/06/28/angular-8-node-server-side-pagination-tutorial-example)
* **React:**[React](https://jasonwatmore.com/post/2017/03/14/react-pagination-example-with-logic-like-google), [React + Node](https://jasonwatmore.com/post/2019/07/18/react-node-server-side-pagination-tutorial-example)
* **Vue:**[Vue](https://jasonwatmore.com/post/2019/08/21/vue-js-simple-pagination-example), [Vue + Node](https://jasonwatmore.com/post/2019/08/12/vuejs-node-server-side-pagination-tutorial-example)
* **AngularJS:**[AngularJS](https://jasonwatmore.com/post/2016/01/31/angularjs-pagination-example-with-logic-like-google)
* **ASP.NET:**[Razor Pages](https://jasonwatmore.com/post/2018/10/15/aspnet-core-razor-pages-pagination-example), [ASP.NET MVC](https://jasonwatmore.com/post/2015/10/30/aspnet-mvc-pagination-example-with-logic-like-google)

This is a simple example of how to implement client-side pagination in Angular 9.

The example contains a hard coded array of 150 objects split into 15 pages to demonstrate how the pagination component works. Styling in the example is done with bootstrap 4.4 css but you can style any way you like using the provided css classes, details are provided later in the tutorial.

Here it is in action: (See on StackBlitz at <https://stackblitz.com/edit/angular-9-pagination-example>)

**Angular 9 Pagination Component**

Pagination is implemented with the <jw-pagination> component that comes with the jw-angular-pagination package available on npm.

**Installation**

Install the angular pagination component with the command npm install jw-angular-pagination.

**Integration with your Angular 9 app**

Import the JwPaginationModule into your app module (app.module.ts) and add it to the imports array to make the <jw-pagination> component available within your Angular module.

This is the app module (app.module.ts) from the example, the pagination module is imported on line 3 and added to the imports on line 10.

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { JwPaginationModule } from 'jw-angular-pagination';

import { AppComponent } from './app.component';

@NgModule({

imports: [

BrowserModule,

JwPaginationModule

],

declarations: [

AppComponent

],

bootstrap: [AppComponent]

})

export class AppModule { }

**Usage**

There are 2 required properties for using the Angular 9 pagination component:

* **items** - the array of items to be paged
* **changePage** - a callback function for handling the changePage event

There are also a few optional properties:

* **pageSize** - the number of items displayed on each page (defaults to 10)
* **maxPages** - the max number of page links to display in the pagination nav (defaults to 10)
* **initialPage** - the initial page to display (defaults to 1)

**Example App Component**

This is the app component (app.component.ts) from the example, it creates a hardcoded array of items to be paged in the ngOnInit() method, and updates the current page of items in the onChangePage() callback method.

import { Component, OnInit } from '@angular/core';

@Component({ selector: 'app', templateUrl: 'app.component.html' })

export class AppComponent implements OnInit {

items = [];

pageOfItems: Array<any>;

constructor() { }

ngOnInit() {

// an example array of 150 items to be paged

this.items = Array(150).fill(0).map((x, i) => ({ id: (i + 1), name: `Item ${i + 1}`}));

}

onChangePage(pageOfItems: Array<any>) {

// update current page of items

this.pageOfItems = pageOfItems;

}

}

**Example App Component Template**

This is the app component template (app.component.html) from the example, it renders the current page of items using the \*ngFor Angular directive on line 5, and includes the pagination component (<jw-pagination ...>) on line 8.

The pagination component is bound to items property of the app component using the Angular model binding attribute [items]="items", and is bound to the onChangePage() method of the app component using the Angular event binding attribute (changePage)="onChangePage($event)".

<!-- main app container -->

<div class="card text-center m-3">

<h3 class="card-header">Angular 9 Pagination Example</h3>

<div class="card-body">

<div \*ngFor="let item of pageOfItems">{{item.name}}</div>

</div>

<div class="card-footer pb-0 pt-3">

<jw-pagination [items]="items" (changePage)="onChangePage($event)"></jw-pagination>

</div>

</div>

**Styling the Pagination Component**

The JW Angular pagination component is unstyled by default, you can use the below CSS selectors to add your own custom styles.

You can also plug in Bootstrap (3.x or 4.x) which the component works well with.

* **.pagination** - Pagination component container (ul element)
* **.pagination .page-item** - All list items in the pagination component
* **.pagination .page-item .page-link** - All pagination links including first, last, previous and next
* **.pagination .number-item** - All page numbers (1, 2, 3 etc) pagination elements
* **.pagination .first-item** - The 'First' pagination element
* **.pagination .last-item** - The 'Last' pagination element
* **.pagination .previous-item** - The 'Previous' pagination element
* **.pagination .next-item** - The 'Next' pagination element

**Hiding Pagination Buttons**

To hide any of the buttons you can simply set them to display: none; using the css selectors described above.

**Further Customisation of the Pagination Component**

If you want to make other customisations such as changing the HTML template of the component, I'd recommend just copying the pagination component code into your own custom Angular component, it's only 60 lines and will give complete flexibility to make any changes you like.

To use this approach you need to install the jw-paginate package from npm with the command npm install jw-paginate. The jw-paginate package contains the pagination logic used to paginate any array or list of items. For more info about the pagination logic see [this post](https://jasonwatmore.com/post/2018/08/07/javascript-pure-pagination-logic-in-vanilla-js-typescript).

This is the complete pagination component code, it's also available [here](https://github.com/cornflourblue/jw-angular-pagination/blob/master/projects/jw-pagination/src/lib/jw-pagination.component.ts) on GitHub.

import { Component, Input, Output, EventEmitter, OnInit, OnChanges, SimpleChanges } from '@angular/core';

import paginate = require('jw-paginate');

@Component({

moduleId: module.id,

selector: 'jw-pagination',

template: `<ul \*ngIf="pager.pages && pager.pages.length" class="pagination">

<li [ngClass]="{disabled:pager.currentPage === 1}" class="page-item first-item">

<a (click)="setPage(1)" class="page-link">First</a>

</li>

<li [ngClass]="{disabled:pager.currentPage === 1}" class="page-item previous-item">

<a (click)="setPage(pager.currentPage - 1)" class="page-link">Previous</a>

</li>

<li \*ngFor="let page of pager.pages" [ngClass]="{active:pager.currentPage === page}" class="page-item number-item">

<a (click)="setPage(page)" class="page-link">{{page}}</a>

</li>

<li [ngClass]="{disabled:pager.currentPage === pager.totalPages}" class="page-item next-item">

<a (click)="setPage(pager.currentPage + 1)" class="page-link">Next</a>

</li>

<li [ngClass]="{disabled:pager.currentPage === pager.totalPages}" class="page-item last-item">

<a (click)="setPage(pager.totalPages)" class="page-link">Last</a>

</li>

</ul>`

})

export class JwPaginationComponent implements OnInit, OnChanges {

@Input() items: Array<any>;

@Output() changePage = new EventEmitter<any>(true);

@Input() initialPage = 1;

@Input() pageSize = 10;

@Input() maxPages = 10;

pager: any = {};

ngOnInit() {

// set page if items array isn't empty

if (this.items && this.items.length) {

this.setPage(this.initialPage);

}

}

ngOnChanges(changes: SimpleChanges) {

// reset page if items array has changed

if (changes.items.currentValue !== changes.items.previousValue) {

this.setPage(this.initialPage);

}

}

private setPage(page: number) {

// get new pager object for specified page

this.pager = paginate(this.items.length, page, this.pageSize, this.maxPages);

// get new page of items from items array

var pageOfItems = this.items.slice(this.pager.startIndex, this.pager.endIndex + 1);

// call change page function in parent component

this.changePage.emit(pageOfItems);

}

}