Pagination Component

By: Mosh Hamedani

Requirements

We should be able to use this component as follows:

```
<pagination [items]="posts"></pagination>
```

So we give it the array of items (as the data source), and it will figure out how many pages we need. It assumes 10 records in one page, but we can override it:

```
<pagination [items]="posts" [page-size]="pageSize"></pagination>
```

We should be notified when the current page changes (eg user clicking on one of the page numbers, or previous / next buttons). At this point, we can re-calculate the items in the current page (on the server or the client):

```
<pagination [items]="posts" (page-changed)="doSomething($event)">
```

$event should contain the new page number.

Other requirements:

- Pagination should be displayed only if the records do not fit on one page.
- Previous button should be disabled if we're on the first page.
- Next button should be disabled if we're on the last page.
- Currently selected page should be highlighted.
- When changing the user filter, and going back to "Select a user…", we should always start from page 1 (not the previously selected page).

Pagination Component

By: Mosh Hamedani

Tips

Markup

Use Bootstrap Pagination component's markup for the layout.

http://getbootstrap.com/components/#pagination

Lifecycle Hooks

We bind the **items** property of the pagination component to a property in the host component:

```
<pagination [items]="posts"></pagination>
```

In this case, we're binding the **items** property to the **posts** property. Initially, before we get the list of posts from the server, **posts** is an empty array:

```
posts = [];
```

So, at this moment, the pager will not render. When we get the list of posts from the server, our **posts** property will be set to a new array. At this point, pagination should be notified and re-rendered.

You should implement **OnChanges** interface on your pagination component. This is another lifecycle hook that is called every time the bindings are changed. So, when we get the list of posts from the server, the following binding is changed:

```
<pagination [items]="posts"></pagination>
```

Angular will call **ngOnChanges** on the host component (**PostsComponent**) and all its children. This is the opportunity for the pagination component to be notified and re-render.

Paging Data on the Client

For now, implement a simple method to get the posts in a given page:

```
getPostsInPage(page) {}
```

Here, you can calculate the index of the first and last record for the given page, and use a for loop to copy all the records from the source array into a new array.

Take into account an edge case where the last page has less records than the page size.

In the next video, I'll show you a better way to implement paging.