
Table of Contents

Introduction	1.1
Features	1.1.1
Architecture	1.1.2
Installing	1.1.3
Themeing	1.1.4
Getting Started	1.1.5
API	1.2
Table	1.2.1
Inputs	1.2.1.1
Outputs	1.2.1.2
Methods	1.2.1.3
Internals	1.2.1.4
Column	1.2.2
Column Modes	1.2.2.1
Internals	1.2.2.2
Demos	1.3
Online	1.3.1
Source Code	1.3.2
Contributing	1.4
Building	1.4.1
Guidelines	1.4.2
Community	1.4.3
Credits	1.4.4
Changelog	1.5

ngx-datatable

`ngx-datatable` is a Angular2 component for presenting large and complex data. It has all the features you would expect from any other table but in a light package with *no external dependencies*. The table was designed to be extremely flexible and light; it doesn't make any assumptions about your data or how you: filter, sort or page it.

It was built for modern browsers using *TypeScript, CSS3 and HTML5* and Angular `~2.0.0`. This is the sister project of the [angular-data-table](#) that is designed for Angular 1.x.

In The News

AngularAir Esp 76



[Video link](#)

The project was featured on [AngularAir](#) where [@amcdnl](#) spoke about the project, challenges and whats to come.

Features

The table was designed to be *extremely flexible and light*; it doesn't make any assumptions about your data or how you: filter, sort or page it. That said, we wanted to keep the features strictly related to dealing with the data rather than implementing complex filtering/etc that is often very use-case specific. The current features include:

- Handle large data sets (Virtual DOM)
- Expressive Header and Cell Templates
- Horizontal & Vertical Scrolling
- Column Reordering & Resizing
- Client/Serve side Pagination & Sorting
- Intelligent Column Width Algorithms (Force-fill & Flex-grow)
- Integrated Pager
- Cell & Row Selection (Single, Multi, Keyboard, Checkbox)
- Fixed AND Fluid height
- Left and Right Column Pinning
- Row Detail View
- Decoupled theme'ing with included Google Material theme
- Light codebase / No external dependencies

Roadmap

We are really excited about the table and wanted to get it out into the open as quickly as possible so we had to delay some of the features but we plan to add the following:

- Touch selection
- Tree Grids
- Row Grouping

Alternatives

This might not be the best fit for you given the above, there are some other great solutions and some this project borrows from; heres a short list:

- [ng2-super-table](#)
- [ng2-table](#)
- [vaadin-grid](#)
- [iron-data-table](#)
- [paper-datatable](#)

Architecture

Manifesto

There is some things that it doesn't do nor do we plan to do. Lets say you have a requirement to have the ability to edit cell values in the row. Thats a awesome feature but somewhat catered to your use case. How do you invoke the edit? What type of control do you show in the cell? If its a date, do you have controls for that? Its a slippery slope...if you do want to do that you can use the expressive column templates to put whatever components inside your table you want.

What we wanted to do is design a table component that isn't bloated with features that won't fit every use case but instead make a component that does what it does the best possible and is flexible enough to allow you to do what you need to do to solve your requirement.

Some examples of the things we don't plan to build in this project are:

- Export such as CSV
- Context Menus
- Column Filters
- Column Toggling
- Inline Editing

If you are interested in building something like that for this project, I'm happy to help point you in the right direction and would be happy to list a link as a extension.

Installing

You can grab the latest release from the [Releases Page](#) in Github or via [NPM](#).

- `npm install @swimlane/ngx-datatable`

Also, the release code is checked in and resides [here](#).

Module Version

The module is packaged using UMD. The file is `release/index.js`.

CSS

Additionally you will need to include `./release/datatable.css`.

If you want to use material theme, include `./release/material.css`. For more information, visit the 'Themes' section.

Developing

If you are wanting to run the demos locally, just do:

- `npm i`
- `npm start`
- Browse to `http://localhost:9999`

Themes

Out of the box, the data-table is not styled. This gives you maximum flexibility.

There is a separate material theme distributed with data-table. In order to use it, you need to include that in your application `release/material.css` and add the CSS class `material` to your data-table.

This is a simple way to apply the style of the demo.

```
@import '/node_modules/@swimlane/ngx-datatable/release/datatable.css';
@import '/node_modules/@swimlane/ngx-datatable/release/material.css';
@import '/node_modules/@swimlane/ngx-datatable/release/assets/icons.css';
```

You can just add above to your `scss` file and then specify the class of your ngx-datatable to `<ngx-datatable class="material">`

CSS Classes

- `datatable` : Master Table class
 - `fixed-header` : The header is fixed on the table
- `datatable-header` : Header row class
 - `datatable-header-cell` : Header cell class
 - `resizeable` : Cell resizeable class
 - `sortable` : Cell drag/drop sortable class
 - `longpress` : Cell long-press activated
 - `dragging` : Cell dragging activated
 - `sort-active` : Sort active on column
 - `sort-asc` : Sort active on column with ascending applied
 - `sort-desc` : Sort active on column with descending applied
 - `datatable-header-cell-label` : Header cell text label
 - `draggable` : Header cell draggable class
- `datatable-body-row` : Body row class
 - `datatable-row-even` : Odd row class
 - `datatable-row-odd` : Even row class
 - `datatable-body-cell` : Body cell class

- `sort-active` : Sort active on column
- `sort-asc` : Sort active on column with ascending applied
- `sort-desc` : Sort active on column with descending applied

Getting Started

After [Installing](#), include `Angular2DataTableModule` in your application module like:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { NgxDatatableModule } from '@swimlane/ngx-datatable';

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

@NgModule({
  declarations: [AppComponent],
  imports: [NgxDatatableModule, BrowserModule],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

then in your `app.component.ts` you define a table like:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app',
  template: `
    <div>
      <ngx-datatable
        [rows]="rows"
        [columns]="columns">
      </ngx-datatable>
    </div>
  `
})
export class AppComponent {
  rows = [
    { name: 'Austin', gender: 'Male', company: 'Swimlane' },
    { name: 'Dany', gender: 'Male', company: 'KFC' },
    { name: 'Molly', gender: 'Female', company: 'Burger King' },
  ];
  columns = [
    { prop: 'name' },
    { name: 'Gender' },
    { name: 'Company' }
  ];
}
```

and your off to the races!

For more examples, visit the [demos](#) directory in the source code!

Table Inputs

columnMode

The mode which the columns are distributed across the table. For example: `flex` will use flex-grow api, `force` will distribute proportionally and `standard` will just distribute based on widths. Default value: `standard`

columns

Array of columns to display.

count

The total count of all rows. Default value: `0`

externalPaging

Should the table use external paging vs client-side. Default value: `false`

externalSorting

Should the table use external sorting vs client-side. Default value: `false`

footerHeight

The height of the footer in pixels. Pass a `falsey` for no footer. Default value: `0`

headerHeight

The height of the header in pixels. Pass a `falsey` for no header. Default value: `30`

limit

The page size to be shown. Default value: `undefined` .

loadingIndicator

Show the linear loading bar. Default value: `false`

offset

The current offset (`page - 1`) shown. Default value: `0`

reorderable

Column re-ordering enabled/disabled. Default value: `true`

rowHeight

The height of the row. This is necessary for virtual scrolling in order to calculate height for the scrollbar.

rows

Array of rows to display.

scrollbarH

Enabled horizontal scrollbars. Default value: `false`

scrollbarV

Enable vertical scrollbar for fixed height vs fluid. This is necessary for virtual scrolling.
Default value: `false`

selectionType

Type of row selection. Options are `single` , `multi` and `checkbox` . For no selection pass a `falsey` . Default value: `undefined`

selected

List of row objects that should be represented as selected in the grid. It does object equality, for prop checking use the `selectCheck` function.

Default value: `[]`

sorts

Array of sorted columns by property and type. Default value: `[]`

sortType

Single vs Multi sorting. When in single mode, any click after the initial click will replace the current selection with the next selection. In multi selection mode, any incremental click will add to the current selection array.

Default value: `single`

detailRowHeight

Row height of the detail row

rowDetailTemplate

TemplateRef for the row detail.

cssClasses

Custom CSS classes that can be defined to override the icons classes for up/down in sorts and previous/next in the pager. Defaults:

```
sortAscending: 'icon-down',
sortDescending: 'icon-up',
pagerLeftArrow: 'icon-left',
pagerRightArrow: 'icon-right',
pagerPrevious: 'icon-prev',
pagerNext: 'icon-skip'
```

rowIdentity

This will be used when displaying or selecting rows. When tracking/comparing them, we'll use the value of this fn `(fn(x) === fn(y))` instead of `(x === y)` .

messages

Static messages in the table you can override for localization.

```
{
  // Message to show when array is presented
  // but contains no values
  emptyMessage: 'No data to display',

  // Footer total message
  totalMessage: 'total'
}
```

selectCheck

A boolean/function you can use to check whether you want to select a particular row based on a criteria. Example:

```
(row, column, value) => { return value !== 'Ethel Price'; }
```

trackByProp

A property on the row object that uniquely identifies the row. Example: `name`

Table Outputs

All outputs are Angular2 `EventEmitter` ers.

page

The table was paged either triggered by the pager or the body scroll.

```
{
  count
  pageSize
  limit
  offset
}
```

resize

Column was resized.

```
{
  column
  newValue
}
```

reorder

Columns were re-ordered.

```
{
  column
  newValue
  prevValue
}
```

sort

Column sort was invoked.

```
{
  sorts
  column
  prevValue
  newValue
}
```

detailToggle

Row detail row was toggled.

```
{
  rows
  currentIndex
}
```

select

A cell or row was selected.

```
{
  selected
}
```

activate

A cell or row was focused via keyboard or mouse click.

```
{
  type: 'keydown'|'click'|'dblclick'
  event
  row
  column
  value
  cellElement
  rowElement
}
```

scroll

Body was scrolled typically in a `scrollbarV:true` scenario.


```
{  
  offsetX  
  offsetY  
}
```

rowContextMenu

The context menu was invoked on a row.

```
`` { event, row }
```

Table Methods

toggleExpandRow(row)

Toggle expand/collapse a row detail when using row detail templates.

expandAllRows()

Expand all row details when using row detail templates.

collapseAllRows()

Collapse all row details when using row detail templates.

recalculate()

Recalculate the grid sizes.

Table Internals

Row Indexes

Each row is decorated with a `$$index` attribute. This allows us to track the actual index of the row. This is needed because if you are lazy loading data into the grid the index upon which the data is loaded might not always be the 'actual' index its inserted.

This is also leveraged by the virtual scroll so it knows how to offset the row in the page's view port.

Table Column Options

name : **string**

Column label. If none specified, it will use the prop value and decamelize it.

prop : **string**

The property to bind the row values to. If `undefined`, it will camelcase the name value.

flexGrow : **number**

The grow factor relative to other columns. Same as the [flex-grow API](#). It will any available extra width and distribute it proportionally according to all columns' flexGrow values. Default value: `0`

minWidth : **number**

Minimum width of the column in pixels. Default value: `100`

maxWidth : **number**

Maximum width of the column in pixels. Default value: `undefined`

width : **number**

The width of the column by default in pixels. Default value: `150`

resizeable : **boolean**

The column can be resized manually by the user. Default value: `true`

comparator

Custom sort comparator, used to apply custom sorting via client-side. See [MDN](#) for more info.

sortable : **boolean**

Sorting of the row values by this column. Default value: `true`

draggable : boolean

The column can be dragged to re-order. Default value: `true`

canAutoResize : boolean

Whether the column can automatically resize to fill extra space. Default value: `true`

cellTemplate : TemplateRef

Angular TemplateRef allowing you to author custom body cell templates

headerTemplate : TemplateRef

Angular TemplateRef allowing you to author custom header cell templates

checkboxable : boolean

Indicates whether the column should show a checkbox component for selection. Only applicable when the selection mode is `checkbox` .

headerCheckboxable : boolean

Indicates whether the column should show a checkbox component in the header cell. Only applicable when the selection mode is `checkbox` .

Column Modes

Column modes allow you to have a variety of different ways to apply column width distribution to the columns. The table comes with 3 modes; `standard` , `flex` , `force` .

Standard

Columns are distributed given the width's defined in the column options.

Flex

Flex mode distributes the width's grow factor relative to other columns. It works the same as the [flex-grow API](#) in CSS. Basically it takes any available extra width and distribute it proportionally according to each column's `flexGrow` value.

Flex is **not** suggested for when using `scrollH` .

Force

Force mode forces the widths of the columns to distribute equally but overflowing when the min-width of each column is reached. The rules are:

- If combined widths are less than the total width of the grid, proportion the widths given the min / max / normal widths to fill the width.
- If the combined widths, exceed the total width of the grid, use the standard widths.
- If a column is resized, it should always use that width.
- The proportional widths should never fall below min size if specified.
- If the grid starts off small but then becomes greater than the size (+ / -) the width should use the original width; not the newly proportioned widths.

Force is usually the ideal column distribution method when columns do not need to be a fixed sized.

Column Internals

Ids

Each row is decorated with a `$$id` attribute. This allows us to track the column for faster change detection/etc.

Building

This project uses [npm tasks](#) for builds.

Pre-reqs

- Node >= 6.0.0
- Angular CLI
- TypeScript >= 2.0.0

Commands

- `npm start` : Starts Webpack dev server
- `npm run release` : Builds code to `dist` folder
- `npm run tests` : Runs E2E Tests
- `npm run package` : Runs builds, packages and copies results to `./release`

Depedencies

- webpack (`npm install --global webpack`)
- webpack-dev-server (`npm install --global webpack-dev-server`)
- karma (`npm install --global karma-cli`)
- protractor (`npm install --global protractor`)
- typescript (`npm install --global typescript`)

Guidelines

We would love for you to contribute to our project and help make it ever better! As a contributor, here are the guidelines we would like you to follow.

Found an Issue?

If you find a bug in the source code or a mistake in the documentation, you can help us by submitting an issue to our GitHub Repository. Including an issue reproduction (via CodePen, JsBin, Plunkr, etc.) is the absolute best way to help the team quickly diagnose the problem. Screenshots are also helpful.

You can help the team even more and submit a Pull Request with a fix.

Want a Feature?

You can request a new feature by submitting an issue to our GitHub Repository. If you would like to implement a new feature, please submit an issue with a proposal for your work first, to be sure that we can use it. Please consider what kind of change it is:

- For a Major Feature, first open an issue and outline your proposal so that it can be discussed. This will also allow us to better coordinate our efforts, prevent duplication of work, and help you to craft the change so that it is successfully accepted into the project.
- Small Features can be crafted and directly submitted as a Pull Request.

Issue Etiquette

Before you submit an issue, search the archive, maybe your question was already answered.

If your issue appears to be a bug, and hasn't been reported, open a new issue. Help us to maximize the effort we can spend fixing issues and adding new features by not reporting duplicate issues. Providing the following information will increase the chances of your issue being dealt with quickly:

- Overview of the Issue - if an error is being thrown a non-minified stack trace helps
- Angular and angular2-data-table Versions - which versions of Angular and angular2-data-table are affected
- Motivation for or Use Case - explain what are you trying to do and why the current

behavior is a bug for you

- Browsers and Operating System - is this a problem with all browsers?
- Reproduce the Error - provide a live example (using CodePen, JsBin, Plunker, etc.) or a unambiguous set of steps
- Screenshots - Due to the visual nature of angular2-data-table, screenshots can help the team triage issues far more quickly than a text description.
- Related Issues - has a similar issue been reported before?
- Suggest a Fix - if you can't fix the bug yourself, perhaps you can point to what might be causing the problem (line of code or commit)

Community

Below is a list of community contributions and projects that use the table.

- [ng2-inline-editor](#) - inline edit cell values with ng2-inline-editor and angular2-data-table projects.

If you have a project using the table and would love to share with the community, please reach out on gh issues and we would love to add it to the growing list!

Credits

`angular2-data-table` is a [Swimlane](#) open-source project; we believe in giving back to the open-source community by sharing some of the projects we build for our application. Swimlane is an automated cyber security operations and incident response platform that enables cyber security teams to leverage threat intelligence, speed up incident response and automate security operations.

Contributors

Changelog

4.1.0

- Feature: Adding multi-click selection
- Bug: Use tracking fn vs inline (#388)
- Bug: Fix AoT header issue (#389)
- Bug: Fix `reorderable` not being honored (#387)
- Bug: Fix multi-select CTRL not working (#381)
- Chore: Upgrade Angular to ^2.4.1
- Docs: Multi-select

4.0.0

- BREAKING!!!! - Renamed project from `angular2-datatable` to `ngx-datatable` and updated components from `<swui-datatable>` to `<ngx-datatable>`

3.0.0

- BREAKING: Removed `shiftMulti` option, its not useful and creates odd scenarios
- Bug: Permanently high CPU utilization after sorting (#359)
- Bug: Fix checkbox showing in header if defined but not using checkbox select
- Bug: Fix AoT errors (#370)
- Bug: Fix build not outputting AoT files in correct directory
- Bug: Fix checkbox selection not working w/ shift key
- Chore: Upgrade Angular/Webpack

2.2.3

- Bug: Fix header checkbox mutating selection array

2.2.2

- Bug: Fix default selections and don't mutate selection arrow now

2.2.1

- Bug: Fix AoT by reverting TypeScript upgrade until CLI is updated

2.2.0

- Bug: Fix AoT private props (#352)
- Bug: Added null chk to selection getter (#340)
- Bug: multiShift not working after first page (#294)
- Chore: Update Angular to 2.3.0
- Chore: Update TypeScript to 2.1.x
- Chore: Update RxJS to 5.0.0-rc.4
- Chore: Update ZoneJS to 0.7.2
- Demo: Add templates for Webpack plugin for Google Analytics

2.1.3

- Bug: Fix footer not updating when all removes removed
- Bug: Fix Add/Remove items in array not updating (#255)

2.1.2

- Bug: Fix sizing method being debounced on view inits rather than just window resize.

2.1.1

- Bug: Fix memory leak with visibility observer

2.1.0

- Feature: Checkbox selection mode
- Enhancement: Selection mode auto added to host class for easier styling
- Style: Remove row active/select state when in cell mode of material theme
- Bug: Force fill column mode w/ scrollbarH false allowing bleed when resized
- Perf: Faster render time by adding css class via host
- Perf: Debounce window resize calcs

2.0.0

- BREAKING: Rename from `datatable` to `swui-datatable` (#320)
- BREAKING: Redo how row-detail/cell/header templates are described in markup (#304)
- BREAKING: Change Detection switched back to normal
- Enhancement: Row Context Menu Event `rowContextmenu` (#331)
- Bug: Set default value for selected (#314)
- Bug: Fix widths being int vs float (#309)
- Bug: Fix column resize not setting widths to the right (#308)
- Bug: Fix row object mutations (#255)
- Bug: Async Problem when using `@Input` (#253)
- Bug: Sort doesn't work on the first click (#254)
- Bug: Pagination "skip to last page" is not selected after click (#324)
- Bug: `ngFor` Not working in the `datatable-row-detail-template` (#274)
- Chore: Resolve all implicit `any` s (#330)
- Chore: Bump Angular to latest `2.2.4`

1.7.0

- Enhancement: Add RxJS Support
- Chore: Fix builds on windows

1.6.0

- Bug: Column headers not re-ordering (#238)
- Bug: Datatable doesn't sort data correctly when data changed (#284)
- BREAKING: `comparator` now is a normal sort function arguments of `propA, propB` .

1.5.1

- Bug: Custom comparator should return new array (#286)
- Bug: Init selection to empty array (#285)

1.5.0

- Enhancement: `externalSorting` input for simpler server-sorting usage (#281)
- Enhancement: Add `trackByProp` for change detection with mutation of row data

- Bug: Row height variable access protection (#280)
- Chore: Upgrade to Angular 2.2.0
- Demo: Inline editing updates
- Demo: Live data refresh demo
- Breaking: Remove `refresh` method per recommendation by @robwormald

1.4.1

- Bug: Ignore next page when already at last (#223)
- Chore: Redid Webpack Config to be clean
- Chore: TESTS! TRAVIS BUILDS! COVERAGE REPORTING!

1.4.0

- Enhancement: Added `refresh` API for updating table (#255)
- Bug: Fix intersection observer type errors (#268)

1.3.1

- Bug: Fix force column width distribution overriding new resize (#245)

1.3.0

- Enhancement: `selectCheck` fn to prevent selection
- Bug: Fix columns leaking event handlers
- Bug: Fix column toggling errors (#245)
- Bug: Fix AoT Metadata not creating

1.2.0

- Bug: Fix columns losing templates on resize (#252)
- Bug: Fix pager not having right pages when hidden by default
- Bug: Fix expressive column width as attribute with standard column distribution
- Bug: Fix body columns not readjusting after window resize (#251)
- Enhancement: Refactor `emptyMessage` and `totalMessage` to `messages` object
- Enhancement: Huge perf improvement for tables hidden by default

1.1.0

- Feature: NGC Complation
- Bug: Null value in deepValueGetter (#243)
- Chore: Update Depedencies

1.0.0

- Feature: Cell Selection and Keyboard Navigation
- Feature: `activation` events
- Enhancement: `onPush` all the things!
- Enhancement: Add `totalMessage` option for localization
- Enhancement: Demo Page
- Enhancement: Page Count Formatted
- Enhancement: Automatically format column `prop` when no `name` passed
- Enhancement: Add ability to pass false to `comparator` for sort handling via event
- Bug: Window resize not updating rows in virtual scrolling
- Chore: Switch to SemVer
- BREAKING: `TableOptions` has been removed and options are `Input` on component now
- BREAKING: `TableColumn` class has been removed, just pass normal objects
- BREAKING: Event names has been renamed using Angular2 standards
- BREAKING: Components have been renamed to Angular2 standards
- BREAKING: Removed `StateService`

0.12.0

- Bug: Return empty string on undefined deep values (#232)
- Bug: Fix force fill alog (#218)
- Enhancement: Support for other icon types (#235)
- Enhancement: Add ability to identify rows for proper selection (#154)

0.11.2

- Enhancement: Add ability to define css icon classes for pager / header
- Chore: Upgrade to Angular 2.1.1

0.11.1

- Chore: Polish on new build

0.11.0

- Chore: New build process
- Bug: Fix detail row bug (#212)

0.10.1

- Bug: Fix `$$expanded` undefined with server paging (#210)

0.10.0

- Chore: Upgrade to Angular 2.1.0 (#202)
- Chore: Removed engine restrictions (#195)
- Bug: windows builds with node-sass (#207)
- Bug: resizing not closing correctly (#196)
- Bug: Fix height paging (#208)
- Enhancement: Improve Active CSS (#204)
- Enhancement: Add Empty Message (#194)
- Enhancement: Add deep value getter to sortRows function (#181)
- Enhancement: Sort Classes are applied to body cells (#166)
- Enhancement: AoT Compatibility (#199)
- Feature: Row Detail (#201)

0.9.3

- Column resize sometimes gives weird behaviour on mouse resize/click (#155)
- Fix order of setters in DataTable ngOnChanges (#179)
- Remove document event listener subscription leak in draggable & resizable
- Fix `setScrollTop` undefined error (#182)

0.9.2

- Fix `name` being `undefined` introduced in 0.9.0 release

0.9.1

- Export component references for external consumption (#176)

0.9.0

- Fix accidental breaking change of renaming `HeaderCell` column property to `model` . See [commit](#).
- Ensure `minWidth` and `maxWidth` values are specified saved as numbers (#167)
- Add row double click option (#168)

0.8.0

- Added the ability to define header templates expressively *Breaking Change!* Renamed `template` to `cellTemplate` in column options

0.7.4

- Removed #142 in favor of `style height`
- Fixed issue with `height + scrollbarV` not sizing right
- Fix limit not applied (#133)
- Fix sort not resetting to top of page (#136)
- Added option validation

0.7.3

- Huge perf bumps (#149)

0.7.2

- Build fixes

0.7.1

- Removed template wrapper in favor of native template outlet

0.7.0

- Upgrade Angular 2.0.1 & ZoneJS
- Angular Code Style Compliance (#147)
- Fix initial load of rows jumbled (#156)
- Update row/options setting to ngOnChanges (#151)
- Fix column height not set correctly (#144)

0.6.1

- Virtual Scrolling Emits Paging (#130)

0.6.0

- Update to Angular 2.0.0!
- Fix horizontal header issue (#129)

0.5.1

- Fixed Multiple Tables on Same Page (#103)
- Fix TS Helpers not being included in release (#107)
- Update `onPage` API to reflect docs (#116)

0.5.0

- Upgrade to Angular2 RC7