Datagrid (Advanced) | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/elements/advanced_datagrid.html> |
| Date de scraping | 2025-04-08T15:53:28.990909+00:00 |

## Datagrid (Advanced) ​

The advanced datagrid component allows you to display and interact with structured data in a tabular format. Built on the AG Grid framework, it offers high-performance rendering with advanced features like sorting, filtering, pagination, and cell editing. You can fully customize its appearance and behavior to meet your specific requirements.

## Features ​

High-Performance Rendering : Efficiently handles large datasets with virtual scrolling Advanced Sorting & Filtering : Helps users quickly find and organize data Row Selection : Single or multiple row selection with full event handling Interactive Cell Editing : Edit data directly within the grid with validation Custom Cells : Add your own UI components to grid cells Pagination : Traditional page-based navigation for structured data review Column Management : Resize, reorder, and pin columns as needed Comprehensive Styling : Extensive appearance customization for all grid elements

## Getting Started ​

## Using AI ​

The quickest way to set up the datagrid is by using AI.  
First, ensure you have a variable or collection with your data ready. Ask AI: "Create a datagrid that uses {{insert your data}} ". Make sure you pass in your data as context in the chat.  
  
Continue refining and customizing your datagrid by asking AI for specific modifications: "Add an action column with an 'Edit' button" "Enable multiple row selection on the datagrid" "Make the email column wider and show the status column with color indicators" "Add pagination showing 5 rows per page"

### Images

<https://docs.weweb.io/assets/datagrid-setup-with-ai.DPl9_d9v.gif>

<https://docs.weweb.io/assets/datagrid-update-with-ai.vMRLFvjg.gif>

## Manual Setup ​

Add the datagrid to your page from the Add panel Configure the data source: Click the Data property and select a variable or collection Set a unique row ID formula (typically using the primary key of your data) Click Generate Columns to create columns based on your data structure  
  
Customize columns as needed through the properties panel Configure appearance settings in the Style panel Add workflows to handle interactions like row selection or action buttons

### Images

<https://docs.weweb.io/assets/datagrid-bind-data.Bfw-a6VU.gif>

## Column Configuration ​

The datagrid organizes your data into columns that can be extensively customized.

## Column Types ​

## Custom Display Formatting ​

For Text, Number, Boolean, and Date columns, you can customize how values are displayed without changing the underlying data:  
Select a column in the properties panel Toggle on Custom display value Create a formula that transforms the raw value into your desired format

### Images

<https://docs.weweb.io/assets/datagrid-custom-display.DSe_NdNT.gif>

## Interactive Features ​

## Cell Editing ​

Enable users to edit data directly in the grid:  
Select a column in the properties panel Toggle on Editable  
Users can now double-click a cell to edit its content  
  
Create a workflow on the datagrid using the On Cell Value Changed trigger  
In your workflow, you'll have access to: The old value The new value Which column was changed The complete row data

### Images

<https://docs.weweb.io/assets/datagrid-cell-edit.TOTg8zfA.gif>

## Row Selection ​

Allow users to select one or multiple rows:  
In the datagrid properties, set Row Selection to either Single or Multiple Users can now select rows by clicking on them  
  
Create workflows for selection events: On Row Selected : Triggered when a row is selected On Row Deselected : Triggered when a row is deselected  
Access all currently selected rows via the exposed selectedRows variable OR access only the data of the row that triggered the On Row Selected or On Row Deselected event

### Images

<https://docs.weweb.io/assets/datagrid-selecting-rows.BjmovuCY.gif>

<https://docs.weweb.io/assets/datagrid-row-selection-workflow.D5M6lUda.png>

<https://docs.weweb.io/assets/datagrid-row-deselection-workflow.XXYNJnaC.png>

<https://docs.weweb.io/assets/datagrid-row-selected-var-data.HFbcF87W.png>

<https://docs.weweb.io/assets/datagrid-row-selected-data.6UAKVrPc.png>

## Action Buttons ​

Add interactive buttons to each row of your datagrid:  
Add a new column with type set to Action Configure: Action Name : Identifier for the action (used in workflows) Action Label : Text displayed on the button  
  
Create a workflow on the datagrid using the On Action trigger In your workflow, you'll have access to: Which action was triggered The complete row data

### Images

<https://docs.weweb.io/assets/datagrid-action-buttons.D_Nb20cx.png>

<https://docs.weweb.io/assets/datagrid-on-action.DWivhecI.png>

## Custom Cell Content ​

Create completely custom cell rendering:  
Add a column with type set to Custom A Dropzone appears on the canvas where you can add any UI elements Drag and drop elements into this dropzone to create your custom cell appearance These elements will have access to the cell value and the entire row data

### Images

<https://docs.weweb.io/assets/datagrid-custom-cell.DcksvaIG.gif>

## Managing Large Datasets ​

The datagrid provides two approaches to handling large amounts of data:

## Virtual Scrolling (Default) ​

By default, the datagrid uses virtual scrolling to efficiently render large datasets:  
Only visible rows are rendered in the DOM Rows load dynamically as users scroll Provides smooth performance even with thousands of rows

### Images

<https://docs.weweb.io/assets/datagrid-virtual-scroll.Udae4-3i.gif>

## Pagination ​

For a more traditional approach to large datasets:  
Toggle on Pagination in the datagrid properties Set the desired number of rows per page (default: 10)  
The datagrid will now display page navigation controls, and users can browse through pages of data.

### Images

<https://docs.weweb.io/assets/datagrid-using-pagination.BwdBcXfb.gif>

## Advanced Usage ​

## Dynamic Column Definition ​

For programmatically defining columns, bind the columns property to a variable with this structure:

### Code

Langage: unknown

[  
 {  
 "headerName": "Name",  
 "cellDataType": "text",   
 "field": "name",  
 "width": "200px",  
 "sortable": true  
 },  
 {  
 "headerName": "Price",  
 "cellDataType": "number",  
 "field": "price",   
 "width": "100px",  
 "sortable": true  
 },  
 {  
 "headerName": "In Stock",  
 "cellDataType": "boolean",  
 "field": "inStock",  
 "width": "100px"  
 },  
 {  
 "headerName": "Product Image",  
 "cellDataType": "image",  
 "field": "thumbnail",  
 "imageWidth": "50px",  
 "imageHeight": "50px"  
 },  
 {  
 "headerName": "Actions",   
 "cellDataType": "action",  
 "field": "edit",  
 "actionName": "editRow",  
 "actionLabel": "Edit"  
 }  
]

## Column Width Strategies ​

The datagrid offers two approaches to handling column widths:  
Fixed Width : Set a specific pixel width for the column Good for consistent layouts Best for columns with predictable content length Flex Width : Distribute available space proportionally Adapts to different screen sizes Set flex values to control relative proportions

## Column Pinning ​

For tables with many columns, pin important columns to ensure they stay visible:  
Set the Pinned property to either Left or Right Pinned columns remain fixed while users scroll horizontally Useful for ID columns, status indicators, or action buttons

### Images

<https://docs.weweb.io/assets/datagrid-pin-column.BcBo7_oI.gif>

## Example Use Cases ​

Product Catalog : Display products with images, pricing, inventory status, and edit/view actions User Management : List users with filterable columns for roles, status, and registration dates Financial Dashboard : Present financial data with custom number formatting and row selection Task Tracker : Show tasks with status indicators, due dates, and action buttons Data Analysis : Enable sorting and filtering for data exploration with visual indicators

## Forking ​

While the DataGrid component offers extensive built-in customization options, there may be cases where you need functionality beyond what's natively available. In these situations, you can fork the component and modify it to meet your specific requirements.  
If you are unsure how to fork an element, you can learn more in the dedicated documentation .

## Forking Example: Custom Pagination Styling ​

One reason to fork the DataGrid would be to create styling properties not available in the standard component. For example, you might want to be able to change the styling of the pagination.  
To do so, you could fork the datagrid, then edit with AI and pass in the following prompt:  
The AI will then add new styling properties to control the appearance of different aspects of the pagination:

### Code

Langage: unknown

I want to edit this forked datagrid so that I can change the styling of the pagination

### Images

<https://docs.weweb.io/assets/datagrid-pagination-fork.DxhYdtwk.gif>

## Properties Reference ​

## Settings Properties ​

## Column Properties ​

Each column in the datagrid can be configured with these properties:

## Image Column Properties ​

## Action Column Properties ​

## Styling Properties ​

## General ​

## Header ​

## Row ​

## Column ​

## Cell ​

## Menu ​

## Action ​

## Event Triggers ​

The datagrid component provides events you can use to trigger workflows:

## Example Event Payloads ​

## On Action ​

### Code

Langage: unknown

{  
 actionName: "edit",  
 row: {  
 id: 42,  
 name: "Product A",  
 price: 29.99,  
 inStock: true  
 },  
 id: 42,  
 index: 5,  
 displayIndex: 2  
}

## On Cell Value Changed ​

### Code

Langage: unknown

{  
 oldValue: 29.99,  
 newValue: 24.99,  
 columnId: "price",  
 row: {  
 id: 42,  
 name: "Product A",  
 price: 24.99,  
 inStock: true  
 }  
}

## On Row Selected / On Row Deselected ​

### Code

Langage: unknown

{  
 row: {  
 id: 42,  
 name: "Product A",  
 price: 29.99,  
 inStock: true  
 }  
}

## Exposed Variables ​

The datagrid component exposes the following variables for use in your application: