1
00:00:08.740 --> 00:00:12.470
Hi everyone. Organizing complex data is crucial for a great

2
00:00:12.620 --> 00:00:16.500
user experience. That's why we're excited to introduce the Jutro Table

3
00:00:16.600 --> 00:00:17.120
component.

4
00:00:19.320 --> 00:00:23.310
The Jutro Table component helps your users efficiently scan, compare,

5
00:00:23.670 --> 00:00:27.500
and act on large data sets. The table component gives you the power to

6
00:00:27.580 --> 00:00:30.720
build tables right into your applications.

7
00:00:30.780 --> 00:00:34.760
You can use it for anything from simple data displays to complex tables that

8
00:00:34.800 --> 00:00:38.670
handle large data sets. Need advanced features like search and filtering?

9
00:00:39.140 --> 00:00:42.760
This component has you covered. Since the implementation is modular,

10
00:00:43.060 --> 00:00:47.000
you select only the features your application needs, keeping things clean.

11
00:00:47.060 --> 00:00:50.980
We provide a set of composable core components that you can easily extend or

12
00:00:51.000 --> 00:00:53.120
customize to fit your specific use case.

13
00:00:53.520 --> 00:00:57.340
This means you can create simple tables for basic needs or build flexible,

14
00:00:57.500 --> 00:01:00.860
powerful solutions for your most complex requirements.

15
00:01:00.900 --> 00:01:04.840
For developers, it's built with first-class accessibility and best practices baked

16
00:01:04.960 --> 00:01:08.100
in, letting you focus on the solution, not the boilerplate.

17
00:01:09.320 --> 00:01:12.590
We'll show you how to scaffold a basic table and then dive into the most

18
00:01:12.620 --> 00:01:16.560
sought-after features: sorting, filtering, row selection,

19
00:01:16.980 --> 00:01:19.550
row expansion, and handling large data sets.

20
00:01:21.760 --> 00:01:25.640
Let's start by understanding the basic scaffold and semantic structure of the

21
00:01:25.680 --> 00:01:26.340
Jutro table.

22
00:01:27.180 --> 00:01:31.120
You import primitives from @jutro/components and nest them in a

23
00:01:31.200 --> 00:01:34.960
strict hierarchy: Table, TableHeader, HeaderRow, HeaderCell, and so on.

24
00:01:35.280 --> 00:01:38.210
This ensures accurate accessibility semantics out of the

25
00:01:38.260 --> 00:01:41.650
box. You must explicitly set columnIndex on

26
00:01:41.720 --> 00:01:45.700
HeaderCell, and columnIndex and rowIndex on BodyCell to

27
00:01:45.720 --> 00:01:49.540
ensure accurate accessibility semantics when rendering multiple cells.

28
00:01:50.160 --> 00:01:53.380
The columnIndex and rowIndex must start from zero.

29
00:01:54.020 --> 00:01:57.900
The table component does not include any built-in titles or subtitles.

30
00:01:57.960 --> 00:02:01.570
You can easily add them in your implementation using the Jutro Typography

31
00:02:01.600 --> 00:02:04.920
component. To ensure accessibility, add aria-labelledby and

32
00:02:04.940 --> 00:02:08.521
aria-describedby to the table component to reference the title and subtitle.

33
00:02:11.540 --> 00:02:15.080
Next, let's look at how to make columns sortable, a feature highly

34
00:02:15.120 --> 00:02:19.060
requested by our users. Users can click a column header to sort the

35
00:02:19.100 --> 00:02:22.600
table data by that column. A directional icon in the header

36
00:02:22.720 --> 00:02:24.300
indicates the current sort order.

37
00:02:25.500 --> 00:02:27.900

Making a column sortable is straightforward.

38
00:02:27.980 --> 00:02:31.920
Simply add the Boolean isSortable prop to the HeaderCell component.

39
00:02:31.960 --> 00:02:35.160
You'll need to manage the sorting logic and state within your component.

40
00:02:35.560 --> 00:02:39.520
Use the isSorted prop to indicate the current direction and the onSort

41
00:02:39.580 --> 00:02:41.700
handler to trigger your state change logic.

42
00:02:42.030 --> 00:02:45.700
The nice part is that HeaderCell handles the accessibility details for you.

43
00:02:46.080 --> 00:02:50.000
When you wire up isSortable and isSorted, the component automatically sets

44
00:02:50.040 --> 00:02:54.000
the correct aria-sort value and descriptive labels, so screen readers

45
00:02:54.020 --> 00:02:57.829
can announce whether a column is sorted ascending or descending without any

46
00:02:57.880 --> 00:03:01.489
extra work on your side. Next, let's look at search and filtering,

47
00:03:01.790 --> 00:03:04.500
another common requirement when you're working with larger data sets.

48
00:03:05.850 --> 00:03:09.400
Search lets users quickly find matching rows across the whole table.

49

00:03:09.420 --> 00:03:13.230
To implement search, you can wire a search bar to your table using a

50
00:03:13.260 --> 00:03:17.240
small data management hook that debounces input and filters rows based

51
00:03:17.280 --> 00:03:20.800
on accessor functions. You place a search input above the table,

52
00:03:21.100 --> 00:03:24.920
track a search query in state, and use a small helper or hook to filter your

53
00:03:25.000 --> 00:03:27.200
data array before rendering rows.

54
00:03:28.660 --> 00:03:32.500
Where search lets users type free-form text, filtering lets them

55
00:03:32.540 --> 00:03:36.090
narrow the data set using structured controls such as dropdowns and

56
00:03:36.120 --> 00:03:37.760
checkboxes for specific columns.

57
00:03:38.220 --> 00:03:42.160
Each control updates filter state, and your filtering logic returns only

58
00:03:42.220 --> 00:03:45.700
the rows that match all active filters, which you pass into BodyRow and

59
00:03:45.760 --> 00:03:46.360
BodyCell.

60
00:03:47.860 --> 00:03:51.680
For users who need to take actions on multiple items, row selection is

61
00:03:51.720 --> 00:03:55.580

essential. You implement this by introducing a checkbox in the first

62
00:03:55.660 --> 00:03:59.160
column for both the header for select all and the body.

63
00:03:59.900 --> 00:04:03.760
The table component allows you to reflect the selection state visually

64
00:04:03.820 --> 00:04:06.770
using the Boolean selected prop on the BodyRow component.

65
00:04:07.200 --> 00:04:11.180
You'll maintain your source of truth for the selected IDs in your component state

66
00:04:11.520 --> 00:04:14.410
and use the selected prop to apply the correct styling.

67
00:04:16.240 --> 00:04:20.180
On top of selection, the table component provides a first-class way to support row

68
00:04:20.221 --> 00:04:20.860
expansion.

69
00:04:22.720 --> 00:04:26.340
Expandable rows use an icon, typically a chevron, in a

70
00:04:26.360 --> 00:04:30.260
dedicated column so users can reveal additional details directly under the

71
00:04:30.300 --> 00:04:34.280
main row. The expanded content lives in subrows that share the same grid

72
00:04:34.380 --> 00:04:37.930
structure, which means keyboard navigation and focus management still work

73
00:04:38.000 --> 00:04:39.780

exactly as expected.

74
00:04:41.000 --> 00:04:44.419
Expanded content is just extra rows in the same grid.

75
00:04:44.520 --> 00:04:48.400
As long as each cell has the right row index and column index, the

76
00:04:48.440 --> 00:04:52.400
table's keyboard navigation treats expanded sections as part of one

77
00:04:52.420 --> 00:04:53.969
continuous coordinate system.

78
00:04:55.960 --> 00:04:59.880
When dealing with large amounts of data, performance and user experience

79
00:04:59.920 --> 00:05:03.730
are paramount. Adding pagination is key to keeping your application

80
00:05:03.760 --> 00:05:07.460
fast and scannable. For pagination, you typically maintain

81
00:05:07.500 --> 00:05:11.440
page index and page size in your component state and only render

82
00:05:11.450 --> 00:05:13.780
the slice of rows that belong to the current page.

83
00:05:14.280 --> 00:05:18.050
Jutro provides a dedicated pagination component for the controls at the bottom of

84
00:05:18.060 --> 00:05:22.020
the table, including the ability to move between pages and change how

85
00:05:22.040 --> 00:05:25.490
many rows are shown per page. Beyond pagination,

86

00:05:25.900 --> 00:05:29.300
always implement feedback states for loading, empty, or error

87
00:05:29.400 --> 00:05:33.200
scenarios. Jutro provides two purpose-built components for table

88
00:05:33.320 --> 00:05:36.170
states: TableLoader and TableEmptyState.

89
00:05:36.600 --> 00:05:40.420
These components automatically center content inside the table body

90
00:05:40.460 --> 00:05:43.380
and use context-aware height scaling to avoid flicker.

91
00:05:43.860 --> 00:05:47.410
You render TableLoader inside TableBody while data is being

92
00:05:47.540 --> 00:05:51.440
fetched. It shows a spinner and a loading data label by default.

93
00:05:51.880 --> 00:05:55.790
When there's no data to show, you can use TableEmptyState in place
of

94
00:05:55.820 --> 00:05:59.740
rows. It gives you a main message like "Nothing to display," and you

95
00:05:59.760 --> 00:06:03.360
can wire it with logic-based actions like "Add a new entry" for an

96
00:06:03.400 --> 00:06:06.860
initial empty state. For error scenarios, you can configure

97
00:06:06.900 --> 00:06:10.640
TableEmptyState to display a warning-style message and an action
like

98
00:06:10.660 --> 00:06:11.180
reload.

99

00:06:12.290 --> 00:06:15.910
The Jutro Table component provides a powerful, accessible, and

100
00:06:16.040 --> 00:06:18.660
structured foundation for your data needs.

101
00:06:18.720 --> 00:06:22.680
We've covered building the core structure, implementing sorting, filtering,

102
00:06:22.840 --> 00:06:26.710
row selection and expansion, and handling large data with pagination

103
00:06:26.720 --> 00:06:30.260
and feedback states. Ready to apply this new knowledge?

104
00:06:30.300 --> 00:06:34.240
Find examples, usage information, and component props in the

105
00:06:34.300 --> 00:06:36.160
Jutro Table component documentation.

106
00:06:38.440 --> 00:06:42.180
For detailed code samples for different use cases, including examples of

107
00:06:42.280 --> 00:06:45.980
using the TanStack Table external library, see the table component

108
00:06:46.040 --> 00:06:48.040
examples in Jutro Storybook.