

jQuery Mobile

Widgets and UI

Lesson 1, Activity 2: User Interface Elements

In this lesson, we'll cover a range of useful UI elements - from toolbars to lists to sliders to form elements to grids - built in to jQuery Mobile. All of these UI widgets are based on traditional HTML5 (or HTML) tags, sometimes with `data-x` attributes added to present the element in a mobile-optimized fashion.

Toolbars

Most jQuery Mobile pages will show a header and footer toolbar, at top and bottom, respectively. The header bar typically shows the page title and, often, a navigation bar; the header bar on some interior pages may show a "back" button. The footer bar is usually the last element on a page.

Header Toolbars

All of the jQuery Mobile pages we've seen so far have had a header toolbar - it would be nice if, on interior pages, we included a "back" button for the user to easily return to the previous page. Adding a "back" button is pretty easy: insert `data-add-back-btn="true"` to the `data-role="page"` div. We can also easily add one or two buttons to the header toolbar. Add buttons as children of the `data-role="header"` element: the first button will be displayed left of the title and the second will be displayed to the right.



The left screenshot shows a header from a page with `data-add-back-btn="true"`. The right screenshot shows a header with two buttons, links to "Home" and "Page 1".

Open [Widgets/1/Demos/toolbars/index.html](#) in a mobile browser, and in a file editor to view the code, to view this example.

The div with id `page1` has attribute `data-add-back-btn="true"` - this adds the back button to Page 1.

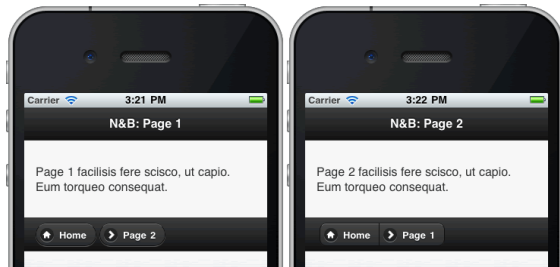
```
<div data-role="page" id="page1" data-add-back-btn="true">
```

Page 2's page div doesn't have attribute `data-add-back-btn="true"`; instead, we add the buttons "manually" to the header:

```
<div data-role="header">
  <a href="#home" data-role="button" data-icon="home">Home</a>
  <h2>N&B: Page 2</h2>
  <a href="#page1" data-role="button" data-icon="grid">Page 1</a>
</div>
```

Footer Toolbar

Footer toolbars work pretty much the same as headers and are a logical place to put contact information, navigation elements, or other show-on-every-page content. Check out [Widgets/1/Demos/toolbars/footer.html](#) - we've added several buttons to each of the interior pages (`#page1` and `#page2`). Wrapping the buttons in a div with attributes `data-role="controlgroup"` and `data-type="horizontal"` displays the buttons with no separation. We add a bit of padding to the footer, around the buttons, with `class="ui-bar"`.



```
<div data-role="footer"> <h3> <a href="#home" data-role="button" data-icon="home">Home</a> <a href="#page2" data-role="button" data-icon="arrow-r">Page 2</a> </h3> </div>
```

Page 1's footer has links to the Home page and to Page 2; we've chosen appropriate icons with the `data-icon` attribute. Containing the footer links in an `h3` tag is a jQuery Mobile convention.

```
<div data-role="controlgroup" data-type="horizontal">
  <a href="#home" data-role="button" data-icon="home">Home</a>
  <a href="#page1" data-role="button" data-icon="arrow-r">Page 1</a>
</div>
```

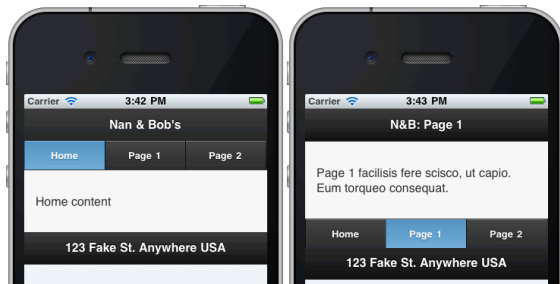
Adding a div with attributes `data-role="controlgroup"` and `data-type="horizontal"` around the buttons in the footer creates a grouped set of button on Page 2.

Navbars

jQuery Mobile offers a basic navbar widget that will show up to five items horizontally across the page; adding more than five items will cause the navbar to wrap to multiple lines. Navbars are typically (though need not be) added to headers or footers.

Navbars are coded as unordered lists of links wrapped in a container element (typically a div) with a `data-role="navbar"` attribute. Set one of the links to the active (selected) state by adding `class="ui-btn-active"` to the a tag.

Open [Widgets/1/Demos/toolbars/navbar.html](#) in a mobile browser, and in a file editor to view the code. We add a navbar to the header of the home page, and to the footer of each of the two interior pages:



Each navbar is an unordered list of links, wrapped in a div with attribute `data-role="navbar"`. jQuery Mobile automatically splits each navbar link into one-third of the width of the page. We use `class="ui-btn-active"` on the on-state nav item appropriately for each page.

```
<div data-role="navbar">
  <ul>
    <li><a href="#home">Home</a></li>
    <li><a href="#page1" class="ui-btn-active">Page 1</a></li>
    <li><a href="#page2">Page 2</a></li>
  </ul>
</div>
```

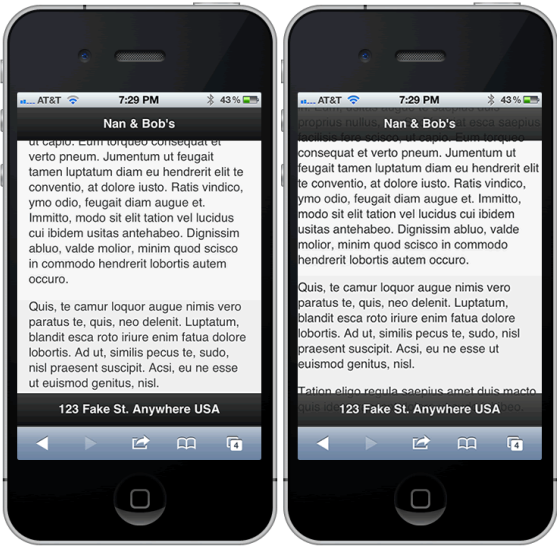
Toolbar Positioning

jQuery Mobile offers several options for the position of toolbars. By default, toolbars use the "inline" positioning mode: headers and footer sit in the flow of the document - header at top, content below it, and footer at bottom.

jQuery Mobile also offers several other toolbar positioning modes. Adding attribute `data-position="fixed"` to the toolbar fixes the header (at the top) or footer (at the bottom) on the page. The toolbar stays fixed while the content scrolls. Tapping the content toggles the visibility of the fixed toolbars. Open [Widgets/1/Demos/toolbars/fixed-standard.html](#) to see this "fixed" mode in action.

Add `data-position="fixed"` and `data-fullscreen="true"` attributes to a toolbar for "fullscreen" positioning. Similar to "fixed" mode, "fullscreen" positioned toolbars sit on top of existing content, with a slight transparency. Open [Widgets/1/Demos/toolbars/fixed-fullscreen.html](#) to see "fullscreen" positioned toolbars.

The following screenshots show fixed toolbars on the left and fixed fullscreen toolbars on the right; note that you can see some of the content behind the toolbars in the fullscreen example.



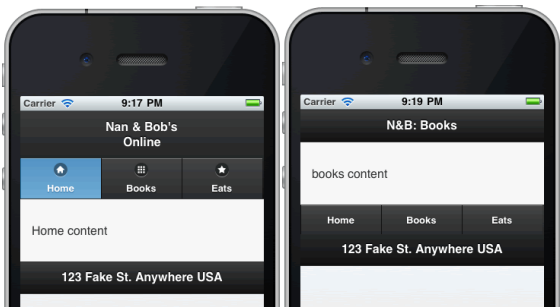
Note that fixed-positioned toolbars will work only in browsers that support CSS's `position:fixed`: most desktop browsers, iOS5+, Android 2.2+, BlackBerry 6, and others.

Lesson 1, Activity 3: Adding Navbars

Duration: 10 to 15 minutes.

In this exercise, you will add navbars to pages for the Nan & Bob's Online site.

1. Open [WidgetKit/Exercise/toolbar.html](#) in a file editor.
2. Add a navbar to the header of the #home page with links to "Home", "Books", and "Eats", add appropriate icons to the top of each link.
3. Add a navbar to the footer of each interior ("Books", "Eats") page; do not add icons to these navbars.
4. Test your work in a mobile browser; the results should look like this:



Solution:

[WidgetKit/Solutions/toolbar.html](#)

```
---- CODE OMITTED ----

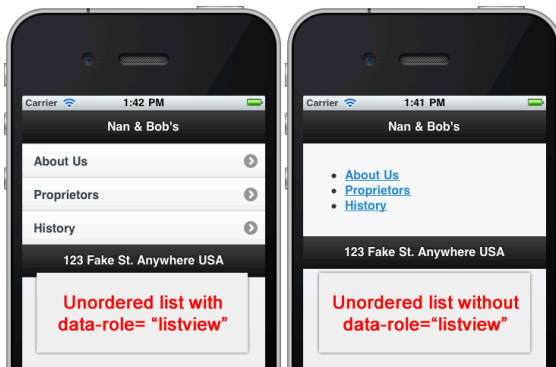
<div data-role="page" id="home">
<div data-role="header">
<h2>Nan & Bob's Online</h2>
<div data-role="navbar">
<ul>
<li><a href="#home" class="ui-btn-active" data-icon="home">Home</a></li>
<li><a href="#books" data-icon="grid">Books</a></li>
<li><a href="#eats" data-icon="star">Eats</a></li>
</ul>
</div>
</div>
---- CODE OMITTED ----

<div data-role="page" id="books">
<div data-role="header">
<h2>N&B: Books</h2>
</div>
<div data-role="content">
<p>books content</p>
</div>
<div data-role="footer">
<div data-role="navbar">
<ul>
<li><a href="#home">Home</a></li>
<li><a href="#books" class="ui-btn-active">Books</a></li>
<li><a href="#eats">Eats</a></li>
</ul>
</div>
</div>
---- CODE OMITTED ----
```

We add a navbar to the header of the "Home" page, with appropriate data-icon values to show an icon at the top of each link. We use class="ui-btn-active" to make the "Home" link active for this page. Each of the non-home pages gets a navbar in its footer, with class="ui-btn-active" again used to make the relevant link active for each page.

Lesson 1, Activity 5: Lists

Whether it is a list of blog entries, products to purchase, or site navigation items, lists play an important role in web content. A `listview` in jQuery Mobile is coded as a simple unordered list (a `` element) with attribute `data-role="listview"`. By default, jQuery Mobile stretches each item, button-like, to fill the screen, adds appropriate padding, and adds a right-arrow icon at right. Unordered lists of links without the `data-role="listview"` attribute still work of course, but don't get the styling:



Open [WidgetsUI/Demos/lists/index.html](#) in a mobile browser, and in a file editor to view the code. The unordered list is in the first (#home) page:

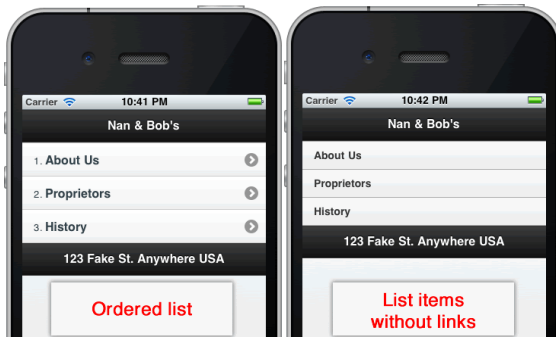
```
<ul data-role="listview">
  <li><a href="#about">About Us</a></li>
  <li><a href="#proprietors">Proprietors</a></li>
  <li><a href="#history">History</a></li>
</ul>
```

Note that we use the `data-add-back-btn="true"` attribute on each of the non-home pages: this automatically adds a button to the upper left of the header of the page:

```
<div data-role="page" id="about" data-add-back-btn="true">
  <div data-role="header">
    <h2>N&amp;B: About Us</h2>
  </div>
  <div data-role="content">
    <p>About us paratus validus foras at praesent usitas probo sino esse vicis sagaciter ratas feugait loquor.</p>
  </div>
  <div data-role="footer">
    <div>
      123 Fake St. Anywhere USA
    </div>
  </div>
</div>
```

See the [jQuery Mobile docs](#) for details on how to customize the back button.

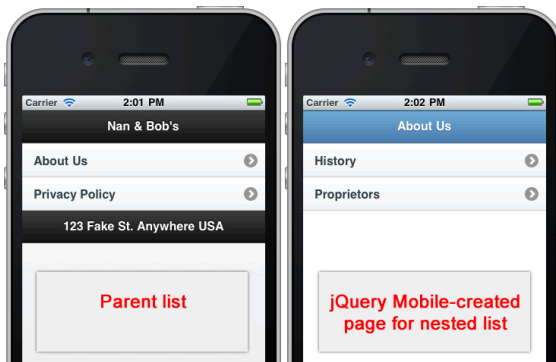
You can, of course, use ordered lists (`ol`, instead of `ul`) to display items with numbers at left. Lists of items without links display much like vertically stacked buttons; the text for these nonlinked lists is slightly smaller than on the link-ful lists, to save space.



Nested Lists

Nesting child `ul` or `ol` list items inside a list allows you to create nested lists. Clicking (tapping) a list item that contains a child list generates a new page populated with the title of the parent in the header and the list of child elements. These dynamic nested lists are styled differently to indicate that you are in a secondary level of navigation. Lists can be nested multiple levels deep.

Open [WidgetsUI/Demos/lists/nested.html](#) in a mobile browser, and in a file editor to view the code. The left screenshot shows the parent list; the right shows the page automatically added by jQuery Mobile to display the nested list:



Here's the code:

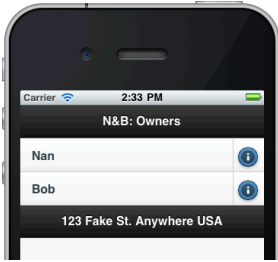
```
<ul data-role="listview">
  <li><a href="#about">About Us</a>
    <ul>
      <li><a href="#history">History</a></li>
      <li><a href="#proprietors">Proprietors</a></li>
    </ul>
  </li>
  <li><a href="#privacypolicy">Privacy Policy</a></li>
</ul>
```

The `#history` and `#proprietors` items are nested inside the `#about` item - tapping "About Us" brings the user to a virtual page displaying the nested list, titled with "About Us", and displayed with slightly different colors.

Split Button Lists

Some content dictates more than one action per list item - a user might be asked, for instance, to edit or delete a given item from a content-management admin screen. jQuery Mobile's split button list addresses this situation: adding a second link inside the list generates a vertical divider line with two clickable regions.

Open [Widget UI Demos/lists/split-button.html](#) in a mobile browser, and in a file editor to view the code:

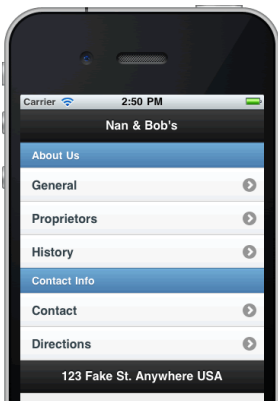


This page lists the two owners of Nan & Bob's. Clicking the name of the owner brings the user to a detail page; clicking the icon at right opens a `mailto:` link to send an email. Note the use of attribute `data-split-icon="info"` on the `ul` (not on each `li`) to specify the icon to use. See the jQuery Mobile docs for a [full list of available icons](#).

```
<ul data-role="listview" data-split-icon="info">
  <li><a href="#nan">Nan</a><a href="mailto:nan@example.com">Contact</a></li>
  <li><a href="#bob">Bob</a><a href="mailto:bob@example.com">Contact</a></li>
</ul>
```

Dividers

Longer lists are more useful with subheaders/dividers - a company directory, for instance, might show a nonclickable list item labeled "A" at the start of all of the "A" names, etc. Add a `li` item with `data-role="list-divider"` to add a divider item to your list:

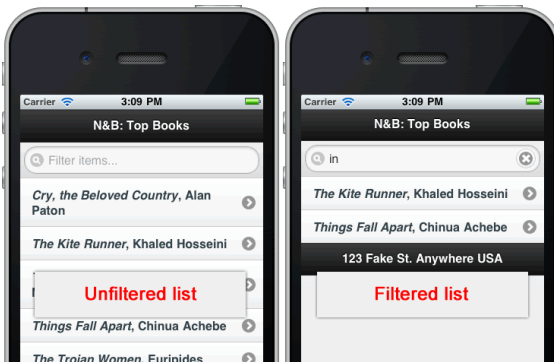


Open [Widget UI Demos/lists/dividers.html](#) in a mobile browser, and in a file editor to view the code:

```
<ul data-role="listview">
  <li data-role="list-divider">About Us</li>
  <li><a href="#about">General</a></li>
  <li><a href="#proprietors">Proprietors</a></li>
  <li><a href="#history">History</a></li>
  <li data-role="list-divider">Contact Info</li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#directions">Directions</a></li>
</ul>
```

Search Filters for Lists

A great way to offer users easy access to longer lists is with jQuery Mobile's built-in list search filter. Simply add the `data-filter="true"` attribute to the list `ul`, and voila: jQuery Mobile filters the list to match the user's typing:



Open [Widget UI Demos/lists/filter.html](#) in a mobile browser, and in a file editor to view the code:

```
<ul data-role="listview" data-filter="true">
  <li><a href="#book"><em>Cry, the Beloved Country</em>, Alan Paton</a></li>
  <li><a href="#book"><em>The Kite Runner</em>, Khaled Hosseini</a></li>
  <li><a href="#book"><em>The Time Traveler's Wife</em>, Audrey Niffenegger</a></li>
  <li><a href="#book"><em>Things Fall Apart</em>, Chinua Achebe</a></li>
  <li><a href="#book"><em>The Trojan Women</em>, Euripides</a></li>
  <li><a href="#book"><em>Twelfth Night</em>, William Shakespeare</a></li>
</ul>
```

The filter works on ordered (`ol`) lists, too.

Note that we've added a custom CSS rule in the head of the document - setting `white-space: normal` for a tags inside of `li` tags forces the text to wrap rather than being cut off.

Further Reading on Lists

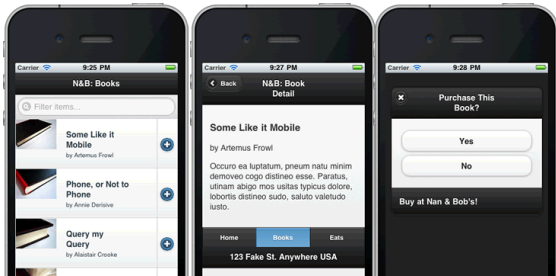
There's even more you can do with lists: formatting text, adding list bubbles (to display, for example, a count of comments for each blog post in a list), controlling icons and thumbnail images, and more. See the [jQuery Mobile docs](#) for more information.

Lesson 1, Activity 6: Adding Lists

Duration: 20 to 30 minutes.

In this exercise, you will add a list of books for purchase to the Nan & Bob's Online site.

1. Open [WidgetUI Examples/Lists.html](#) in a file editor.
2. In the "Books" page, add a list of four books and give them dummy titles.
3. Allow the user to filter the list with a search field at top.
4. Each book listing should show a thumbnail image at left (use the images provided in the "images" directory), the title and author, and a plus icon at right.
5. Clicking the thumbnail image or book title brings the user to a detail page (provided for you) with a "back" button at upper right.
6. Clicking the plus icon brings the user to a dialog asking them to confirm their intent to purchase. Clicking "yes" brings the user to a dummy checkout page, which you will need to create; clicking "no" returns the user to the book-list page.
7. Test your work in a mobile browser, the results should look like this:



Solution:

[WidgetUI Solutions/Lists.html](#)

```
---- CODE OMITTED ----

<ul data-role="listview" data-split-icon="plus" data-filter="true">
  <li><a href="#book1">
    
    <h3>Some Like it Mobile</h3>
    <p>by Artemus Frowl</p>
    </a><a href="#purchaseconfirm" data-transition="pop">Purchase</a></li>
  <li><a href="#book2">
    
    <h3>Phone, or Not to Phone</h3>
    <p>by Annie Derisive</p>
    </a><a href="#purchaseconfirm" data-transition="pop">Purchase</a></li>
  <li><a href="#book3">
    
    <h3>Query my Query</h3>
    <p>by Alastair Crooke</p>
    </a><a href="#purchaseconfirm" data-transition="pop">Purchase</a></li>
  <li><a href="#book4">
    
    <h3>I Phone, iPhone</h3>
    <p>by Apollonia Mack</p>
    </a><a href="#purchaseconfirm" data-transition="pop">Purchase</a></li>
</ul>
---- CODE OMITTED ----

<div data-role="dialog" id="purchaseconfirm">
  <div data-role="header">
    <h2>Purchase This Book?</h2>
  </div>
  <div data-role="content">
    <a href="#checkout" data-role="button">Yes</a>
    <a href="#books" data-role="button">No</a>
  </div>
  <div data-role="footer">
    <p>Buy at Nan & Bob's!</p>
  </div>
</div>
---- CODE OMITTED ----
```

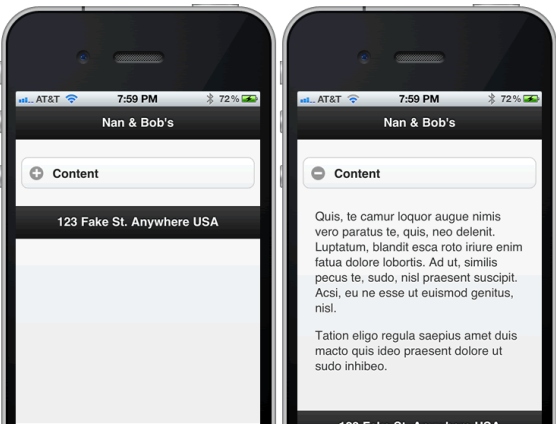
We use a split-button list to present the books available for purchase to the user. Attribute `data-filter="true"` adds the filter functionality; attribute `data-split-icon="plus"` sets the right-side icon.

The first link (pointing to the appropriate detail page) in each list item includes an image, which displays flush left, and the book title and author. The second link (pointing to the confirm dialog) in the list item becomes the icon.

The dialog presents two buttons: "yes" points to the dummy checkout page; "no" points back to the book-list page.

Lesson 1, Activity 8: Collapsible Content

Presenting content blocks that users can toggle open and closed helps improve the user experience; saving even a few pixels of screen real estate on mobile devices' small screens is a good thing. Add `data-role="collapsible"` to any container element - a `div`, say - and jQuery Mobile displays a "collapsible", collapsed by default:



Open up [WidgetUI/Demos/collapsible/index.html](#) to check out the code and view the live demo. Here's the code from the screenshot above:

```
<div data-role="collapsible">
  <h3>Content</h3>
  <p>Quis, te camur loquor augue nimis vero paratus te, quis, neo delenit. Luptatum, blandit esca roto iriure enim fatua dolore lobortis. Ad ut, similis pecus te, sudo, nisl praesent suscipit. Acsi, eu ne esse ut euismod ge
  <p>Tation eligo regula saepius amet dui macto quis ideo praesent dolore ut sudo inhibeo.</p>
</div>
```

Notice that jQuery Mobile labels the collapsible's toggle bar with the text contents of the first header element (H1 - H6) inside the `data-role="collapsible"` and sets the icon at left with plus and minus sign for the open and close actions. The element is collapsed by default (i.e., on page load); use attribute `data-collapsed="false"` to display the element as open when the page loads.

You can set the element as "mini" (`data-mini="true"`) to display the collapsible slightly smaller. You can position the icon on the toggle bar with attribute `data-iconpos="right"`.

Collapsible Sets

Add a container element with attribute `data-role="collapsible-set"` around multiple collapsibles and jQuery Mobile displays a collapsible set. Open one collapsible and the rest of its set-mates collapse - a useful way to present a volume of content in a small space:

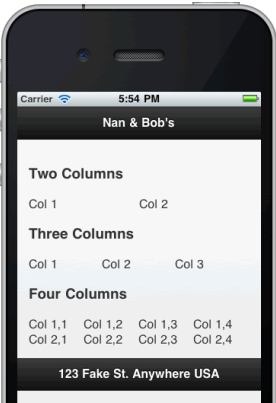


Grids

Multicolumn layouts usually aren't a good idea on mobile devices' small screens, but there are times when you will need to display items in two, three, or more columns. Grid cells in jQuery Mobile have no color, no border, and no padding nor margin.

To lay out elements in a grid, wrap them in a container element (usually a `div`) with class `ui-grid-a` for two columns, `ui-grid-b` for three columns, etc. Each element inside the container gets a class `ui-block-a` [a-z], with the a element (i.e., the element with class `ui-block-a`) at left, the b element next, etc.

Open [WidgetUI/Demos/grids/index.html](#) in a mobile browser, and in a file editor to view the code. This page show a two-column grid and a three-column grid - each with one row - and a four-column grid with two rows:



Here's the code:

```
<h3>Two Columns</h3>
<div class="ui-grid-a">
  <div class="ui-block-a">Col 1</div>
  <div class="ui-block-b">Col 2</div>
</div>
<h3>Three Columns</h3>
<div class="ui-grid-b">
  <div class="ui-block-a">Col 1</div>
  <div class="ui-block-b">Col 2</div>
  <div class="ui-block-c">Col 3</div>
</div>
<h3>Four Columns</h3>
```



```
<div class="ui-grid-c">
  <div class="ui-block-a"><Col 1,1</div>
  <div class="ui-block-b"><Col 1,2</div>
  <div class="ui-block-c"><Col 1,3</div>
  <div class="ui-block-d"><Col 1,4</div>
  <div class="ui-block-a"><Col 2,1</div>
  <div class="ui-block-b"><Col 2,2</div>
  <div class="ui-block-c"><Col 2,3</div>
  <div class="ui-block-d"><Col 2,4</div>
</div>
```

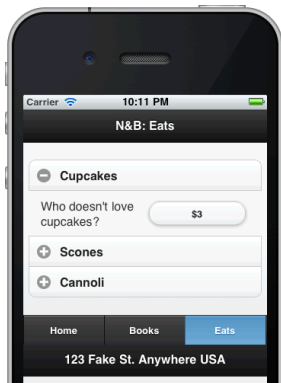
Classes ui-grid-a, ui-grid-b, ui-grid-c set two-, three-, and four-column grid layouts. Note that, with the multirow layout for the four-column grid, the first element of the second row of elements starts over with class ui-block-a.

Lesson 1, Activity 9: Adding Collapsible Content & Grids

Duration: 10 to 15 minutes.

In this exercise, you will add collapsible content for the "Eats" page on the Nan & Bob's Online site.

1. Open [Widgedot II Exercises/collapsible.html](#) in a file editor.
2. Add a `collapsible-set` element to the "Eats" page, allowing users to browse baked goods for sale.
3. Add a few baked-goods items for purchase; recall that the first `HX` (`<h2>`, `<h3>`, etc.) element found within each collapsible will be used as the label for each accordion element.
4. Inside each item's collapsible, use a grid to display the item's description on the left and a purchase button on the right.
5. Label the button with the price of the item, to allow the user to purchase the item; point the button to the `#purchaseconfirm` dialog, as we did for books in the previous exercise.
6. Update the "no" button in the `#purchaseconfirm` dialog to have URL `#` (instead of pointing to the `#books` page) and to use `data-rel="back"`, so that we can use this dialog for confirming the user's intent to purchase both "books" and "cats".
7. Test your work in a mobile browser; the results should look like this:

**Solution:**[Widgedot II/Solutions/collapsible.html](#)

```

---- C O D E   O M I T T E D ----

    <div data-role="page" id="eats">
      <div data-role="header">
        <h2>N&B: Eats</h2>
      </div>
      <div data-role="content">
        <div data-role="collapsible-set">
          <div data-role="collapsible">
            <h3>Cupcakes</h3>
            <div class="ui-grid-a">
              <div class="ui-block-a">Who doesn't love cupcakes?</div>
              <div class="ui-block-b"><a href="#purchaseconfirm" data-role="button" data-mini="true">$3</a></div>
            </div>
          </div>
          <div data-role="collapsible">
            <h3>Scones</h3>
            <div class="ui-grid-a">
              <div class="ui-block-a">Mmmm. Scones.</div>
              <div class="ui-block-b"><a href="#purchaseconfirm" data-role="button" data-mini="true">$4</a></div>
            </div>
          </div>
          <div data-role="collapsible">
            <h3>Cannoli</h3>
            <div class="ui-grid-a">
              <div class="ui-block-a">Take the Cannoli.</div>
              <div class="ui-block-b"><a href="#purchaseconfirm" data-role="button" data-mini="true">$5</a></div>
            </div>
          </div>
        </div>
      </div>
    </div>
---- C O D E   O M I T T E D ----

    <div data-role="dialog" id="purchaseconfirm">
      <div data-role="header">
        <h2>Purchase This Item?</h2>
      </div>
      <div data-role="content">
        <a href="#checkout" data-role="button">Yes</a>
        <a href="#" data-role="button" data-rel="back">No</a>
      </div>
      <div data-role="footer">
        <p>Buy at Nan & Bob's!</p>
      </div>
    </div>
---- C O D E   O M I T T E D ----

```

We add a `div` with attribute `data-role="collapsible-set"` in the "Eats" page, with three collapsible `div`s, one for each baked good item for purchase. Each item is labeled with an `<h3>` tag and contains a two-column grid, showing the item description on the left and the purchase button on the right.

We change the "No" button in the confirm dialog to point to the previous page by adding the attribute `data-rel="back"`, effectively becoming a "close" button for the dialog, which allows us to reuse the dialog both for "books" and for "cats".

Lesson 1, Activity 11: Forms

jQuery Mobile forms, like standard HTML forms, are wrapped in a form tag, with action and method attributes: `<form action="page.php" method="post">`. Because jQuery Mobile uses a single-page navigation model, be sure that each form element has a unique `id` - different from any other element in the site.

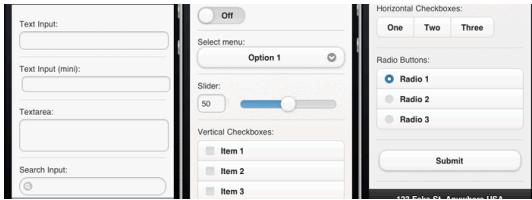
Each form control element should be paired with a meaningful label, connected by the label's `for` attribute and the form control's `id`:

```
<label for="fname">First Name:</label>
<input type="text" name="firstname" id="fname">
```

If using the HTML5 placeholder attribute - to display an instruction in the form element itself - you may want to hide the label. Rather than delete the label (and make your pages harder to use for visitors employing a screen reader or other assistive technology), apply the jQuery Mobile convenience class `ui-hidden-accessible`

```
<label for="fname" class="ui-hidden-accessible">First Name:</label>
<input type="text" name="fname" id="fname" placeholder="First Name">
```

The screenshot below shows the result of viewing [Widgets II/Demos/forms/index.html](#) in a mobile device:



Code Sample:

[Widgets II/Demos/forms/index.html](#)

```
---- CODE OMITTED ----
<form action="#" method="post">
  <!-- we wrap each label/form control in a "fieldcontain" div - this aligns the label/control and adds some visual separation from other fields -->

  <!-- a standard single-line text field -->
  <div data-role="fieldcontain">
    <label for="name">Text Input:</label>
    <input type="text" name="name" id="name">
  </div>

  <!-- a "mini" text field -->
  <div data-role="fieldcontain">
    <label for="name">Text Input (mini):</label>
    <input type="text" name="namemini" id="namemini" data-mini="true">
  </div>

  <!-- a textarea, a multi-line text field -->
  <div data-role="fieldcontain">
    <label for="textarea">Textarea:</label>
    <textarea cols="40" rows="8" name="textarea" id="textarea"></textarea>
  </div>

  <!-- "search" fields display with a magnifying glass icon on some devices -->
  <div data-role="fieldcontain">
    <label for="search">Search Input:</label>
    <input type="search" name="search-input" id="search-input">
  </div>

  <!-- a select of data-role="slider" with two options displays as a flip switch -->
  <div data-role="fieldcontain">
    <label for="flipswitch" class="ui-hidden-accessible">Flip switch:</label>
    <select name="flipswitch" id="flipswitch" data-role="slider">
      <option value="off">Off</option>
      <option value="on">On</option>
    </select>
  </div>

  <!-- a select (dropdown) menu -->
  <div data-role="fieldcontain">
    <label for="select-choice-0" class="select">Select menu:</label>
    <select name="select-choice-0" id="select-choice-1">
      <option value="opt1">Option 1</option>
      <option value="opt2">Option 2</option>
      <option value="opt3">Option 3</option>
    </select>
  </div>

  <!-- a slider displays with a numeric value to the left -->
  <div data-role="fieldcontain">
    <label for="slider">Slider:</label>
    <input type="range" name="slider" id="slider" value="50" min="0" max="100" data-highlight="true">
  </div>

  <!-- the next two fields show checkboxes displayed vertically and horizontally -->

  <div data-role="fieldcontain">
    <fieldset data-role="controlgroup">
      <legend>Vertical Checkboxes:</legend>
      <input type="checkbox" name="checkbox-1a" id="checkbox-1a">
      <label for="checkbox-1a">Item 1</label>

      <input type="checkbox" name="checkbox-2a" id="checkbox-2a">
      <label for="checkbox-2a">Item 2</label>

      <input type="checkbox" name="checkbox-3a" id="checkbox-3a">
      <label for="checkbox-3a">Item 3</label>
    </fieldset>
  </div>

  <div data-role="fieldcontain">
    <fieldset data-role="controlgroup" data-type="horizontal">
      <legend>Horizontal Checkboxes:</legend>
      <input type="checkbox" name="checkbox-6" id="checkbox-6">
      <label for="checkbox-6">One</label>

      <input type="checkbox" name="checkbox-7" id="checkbox-7">
      <label for="checkbox-7">Two</label>

      <input type="checkbox" name="checkbox-8" id="checkbox-8">
      <label for="checkbox-8">Three</label>
    </fieldset>
  </div>

  <!-- radio buttons, grouped with a "fieldset" -->
  <div data-role="fieldcontain">
    <fieldset data-role="controlgroup">
      <legend>Radio Buttons:</legend>
      <input type="radio" name="radio-choice-1" id="radio-choice-1" value="choice-1" checked="checked">
      <label for="radio-choice-1">Radio 1</label>

      <input type="radio" name="radio-choice-1" id="radio-choice-2" value="choice-2">
      <label for="radio-choice-2">Radio 2</label>

      <input type="radio" name="radio-choice-1" id="radio-choice-3" value="choice-3">
      <label for="radio-choice-3">Radio 3</label>
    </fieldset>
  </div>

  <div data-role="fieldcontain">
    <input type="submit" name="submit-button-1" id="submit-button-1" value="Submit">
  </div>
```

```

    </div>
  </form>
  ---- C O D E   O M I T T E D   ----

```

Each field - each form control or set of related form controls - is wrapped by a `div` with attribute `data-role="fieldcontain"`. This framework aligns the input and associated label next to each other, and breaks to stacked block-level elements below about 480px. The fieldcontainer will also add a thin bottom border to act as a visual field separator.

The first two fields ("Text Input" and "Text Input (mini)") show the difference between regular and mini, with the latter being slightly smaller.

The "Textarea" field is coded and displays much like a traditional desktop textarea. The "Search Input" field is an HTML5 search-type `input`; note how jQuery Mobile styles the display with a search icon.

The "Flip switch" element, a common UI element on mobile devices, is used for on/off or true/false fields. One can either drag the handle or tap one side of the switch. It is coded as a `select` with attribute `data-role="slider"`. Note that we hide the label for this element with class `ui-hidden-accessible`.

The next form control, a `select` with label "Select menu", is a standard drop-down menu.

The "Slider" element is a slider, an `input` of type `range` - a useful control for allowing users to pick from a range of values. Add a `step` attribute to the form element to quantize the values to a given step increment.

The next two sets of form elements illustrate the difference between vertical and horizontal checkboxes: vertically displayed checkboxes show a checkbox, whereas in horizontally displayed checkboxes the element acts more like a button, with a color showing the on (checked) state. In both cases, we use a `fieldset` with attribute `data-role="controlgroup"` to group the checkboxes; we use the attribute `data-type="horizontal"` for the horizontal display.

Radio buttons allow for one, and only one, element to be selected, as with traditional desktop radio buttons. A `fieldset`, with attribute `data-role="controlgroup"`, groups the radio buttons.

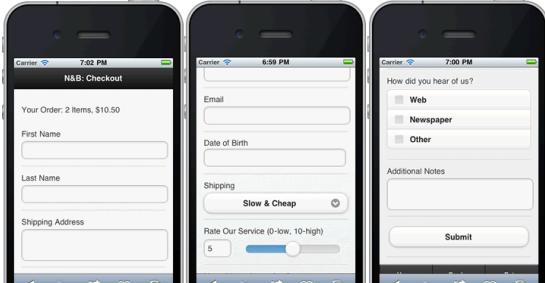
A "Submit" button completes the form.

Lesson 1, Activity 12: Adding Forms

Duration: 20 to 30 minutes.

In this exercise, you will add a checkout form to the Nan & Bob's site.

1. Open [Widget II Exercises/forms.html](#) in a file editor.
2. Add sample text to the top of the #checkout page ("Your Order: 2 items, \$10.50") - the form won't, of course, actually be totaling real items.
3. Add an open and close form tag to the #checkout page. Add an attribute data-ajax="false" to the form tag, to prevent the form from being sent via Ajax. Why? Because we display all pages (each as a div with data-role="page") in our site in one file, we need to submit the form without Ajax to ensure that the thank-you page shows after the user submits the form.
4. Within the form tag, add fields for the following data; use appropriate field types, including new HTML5 form controls:
 - First and last name.
 - Shipping address.
 - Email.
 - Date of birth.
 - Shipping option (standard, expedited, etc.).
 - Rate our service (on scale of 0=low, 10=high).
 - How did you hear of us? (web, newspaper, other).
 - Additional notes.
5. Create a "thank-you page" with an appropriate thank-you message; use this page as the target of the checkout form.
6. Test your work in a mobile browser; the results should look like this:

**Solution:**[Widget II/Solutions/forms.html](#)

```

---- CODE OMITTED ----

<div data-role="page" id="checkout">
  <div data-role="header">
    <h2>N&B: Checkout</h2>
  </div>
  <div data-role="content">
    <p>Your Order: 2 Items, $10.50</p>
    <form action="#thanks" method="post" id="orderform" data-ajax="false">
      <div data-role="fieldcontain">
        <label for="fname">First Name</label>
        <input type="text" name="fname" id="fname">
      </div>

      <div data-role="fieldcontain">
        <label for="lname">Last Name</label>
        <input type="text" name="lname" id="lname">
      </div>

      <div data-role="fieldcontain">
        <label for="address">Shipping Address</label>
        <textarea cols="40" rows="6" name="address" id="address"></textarea>
      </div>

      <div data-role="fieldcontain">
        <label for="email">Email</label>
        <input type="email" name="email" id="email">
      </div>

      <div data-role="fieldcontain">
        <label for="dob">Date of Birth</label>
        <input type="date" name="dob" id="date">
      </div>

      <div data-role="fieldcontain">
        <label for="shipping" class="select">Shipping</label>
        <select name="shipping" id="shipping">
          <option value="slow">Slow & Cheap</option>
          <option value="fast">Fast & Dear</option>
          <option value="surprise">Surprise Me</option>
        </select>
      </div>

      <div data-role="fieldcontain">
        <label for="rateservice">Rate Our Service (0=low, 10=high)</label>
        <input type="range" name="rateservice" id="rateservice" value="5" min="0" max="10" data-highlight="true">
      </div>

      <div data-role="fieldcontain">
        <fieldset data-role="controlgroup">
          <legend>How did you hear of us?</legend>
          <input type="checkbox" name="hearofus" id="hearofus-web">
          <label for="hearofus-web">Web</label>

          <input type="checkbox" name="hearofus" id="hearofus-newspaper">
          <label for="hearofus-newspaper">Newspaper</label>

          <input type="checkbox" name="hearofus" id="hearofus-other">
          <label for="hearofus-other">Other</label>
        </fieldset>
      </div>

      <div data-role="fieldcontain">
        <label for="notes">Additional Notes</label>
        <textarea cols="40" rows="6" name="notes" id="notes"></textarea>
      </div>

      <div data-role="fieldcontain">
        <input type="submit" name="submit-button-1" id="submit-button-1" value="Submit">
      </div>
    </form>
  </div>
</div>

---- CODE OMITTED ----

```

Adding the form tag with attribute data-ajax="false" allows us to submit the form without using Ajax - and thus to submit the form to the #thanks page.

We add appropriate fields for each piece of submitted information:

- An input type="text" for the first and last names
- A textarea for the shipping address and notes
- Inputs of type email and date for the email and date-of-birth fields, respectively
- A select for the shipping options
- A slider for the rate-our-service field
- Checkboxes for "how did you hear of us?"