ECWM 506 Week 7 Lecture

JQuery Mobile Extra Features

Geolocation

CW1 Examples - Padlet and Lecture 6

1. Page setup
2. Lists and Links
3. Pictures (carousel)
4. UI - Simple
5. Filter, Sort, Search, Match

Extras - Forms

Not critical for CW1

Useful for interactions and maps

Moving to security features...

Forms

- All forms should be wrapped in a form tag that has an action and method that will handle the form data processing on the server.

```
<form action=“handler.php" method="post">
  ...
</form>
```

Forms

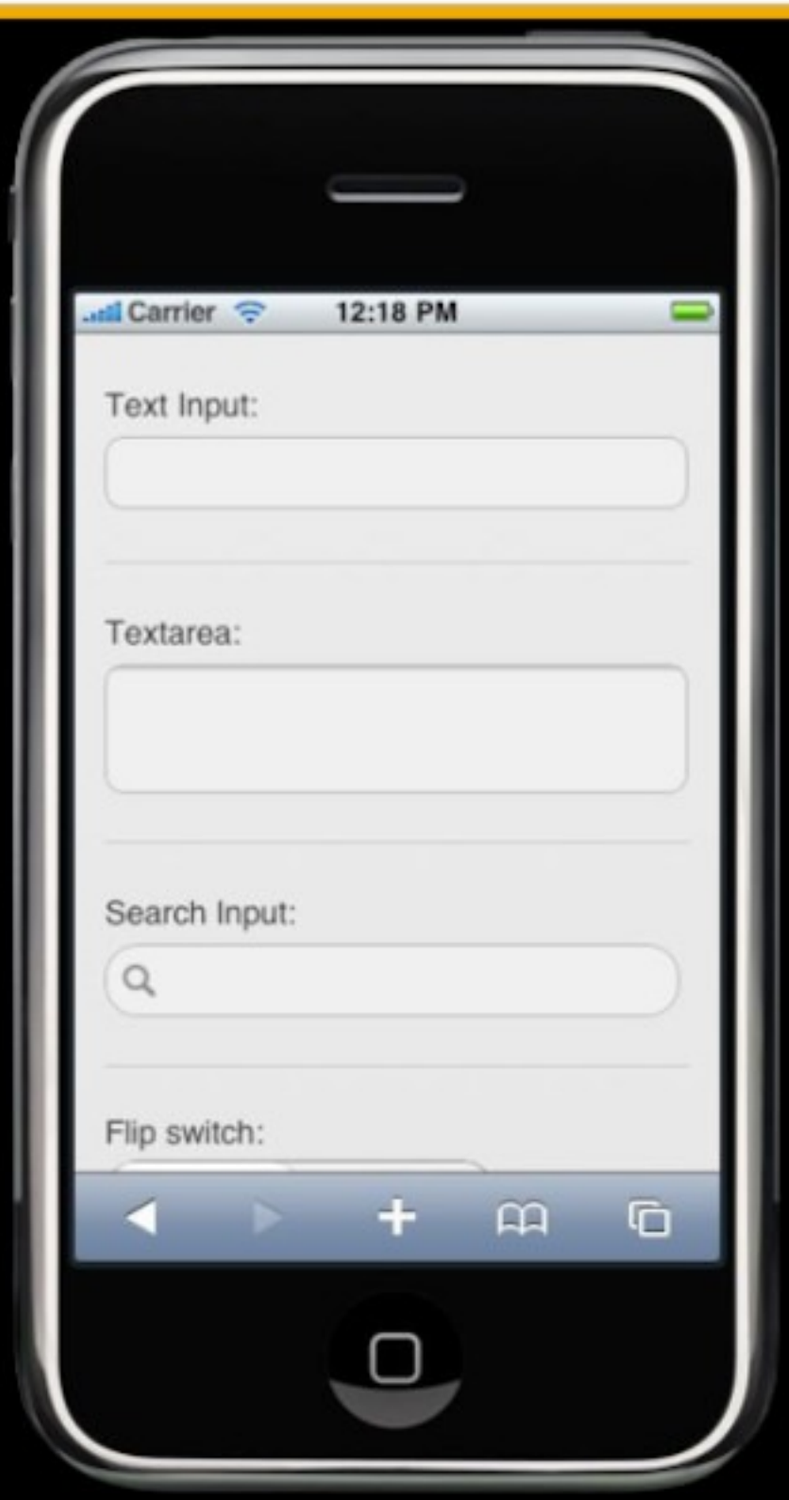
- The `id` attributes of form controls need to be not only unique on a given page, but also unique across the pages in a site, because jQuery Mobile's single-page navigation model allows many different "pages" to be present in the DOM at the same time
- Pair form controls properly with label elements via the `for` attribute

Associate labels with IDs

```
<form action="handler.php" method="post">
    <label for="name">Text Input:</label>
    <input type="text" name="name" id="name" value="" />
</form>
```

Auto-optimised

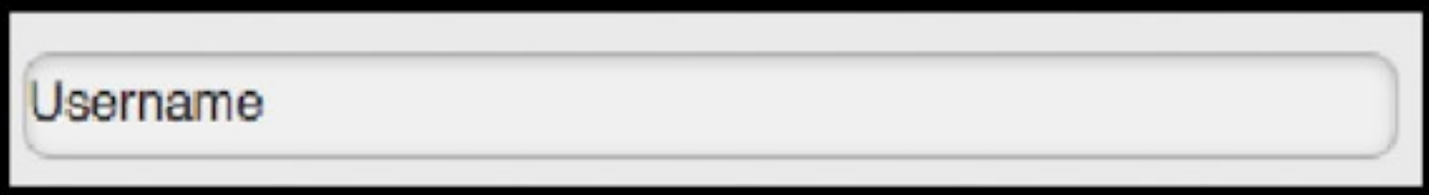
Auto-optimised label/input layout based on width



Forms

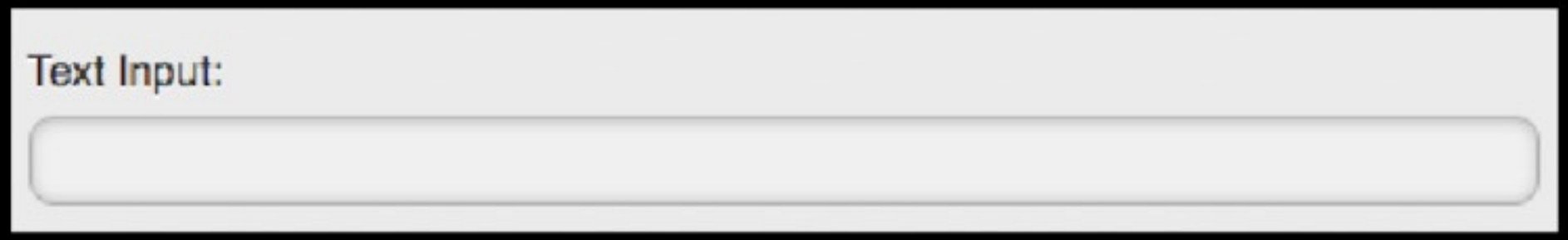
- all form elements must be paired with a meaningful `label` (for accessibility)
- To hide labels in a way that leaves them visible to assistive technologies, apply the helper class `ui-hidden-accessible` to the label itself:

```
<label for="username"
class="ui-hidden-accessible”>Username:
</label>
<input type="text" name="username"
  id="username" value=""
  placeholder="Username"/>
```

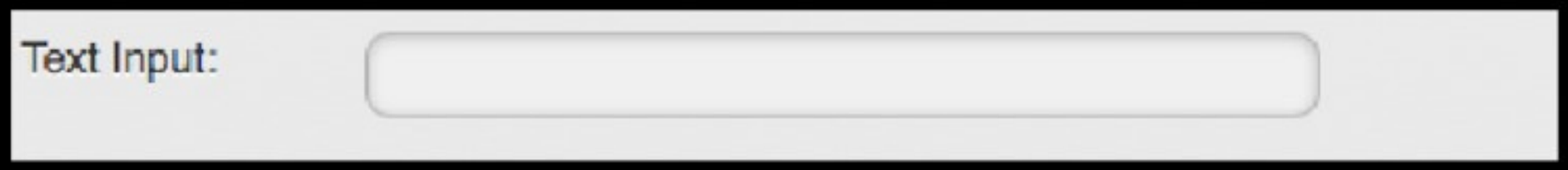


Text input

```
<label for="basic">Text Input:</label>
<input type="text" name="name" id="basic" value="" />
```



Optionally wrap the text input in a container with the `data-role="fieldcontain"` attribute to help visually group it in a longer form.




```
<div data-role="fieldcontain">
    <label for="name">Text Input:</label>
    <input type="text" name="name" id="name" value="" />
</div>
```

type=“password”

type=“number”


Number:
Previous
Next
AutoFill
Done
1 2 3 4 5 6 7 8 9 0
- / : ; ( ) $ & @ ”
#+=
. , ? ! ’
ABC
space
Go

type=“email”


Email:
Previous
Next
AutoFill
Done
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
_123
space
@
.
Go

type=“url”


Url:
Previous
Next
AutoFill
Done
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
@123
.
/
.com
Go

type=“tel”


Tel:
Previous
Next
AutoFill
Done
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
+ * #
0

Auto-grow Textarea

Search

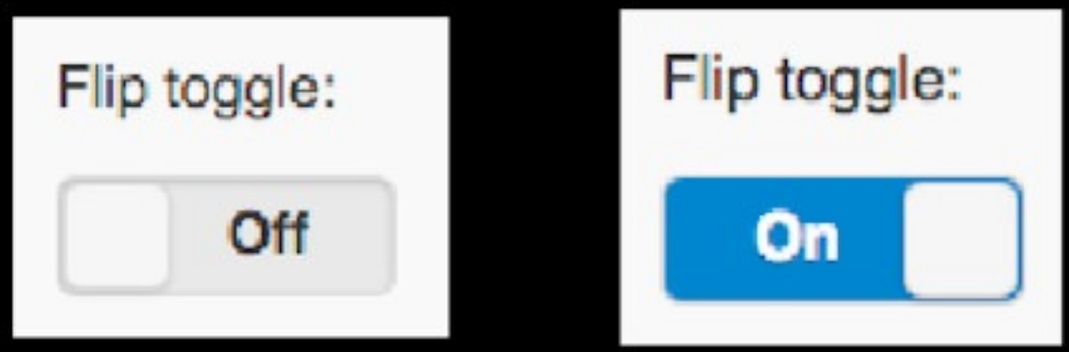

```
<label for="find">Search:</label>
<input type="search" id="find" value="" />
```

Search

Slider

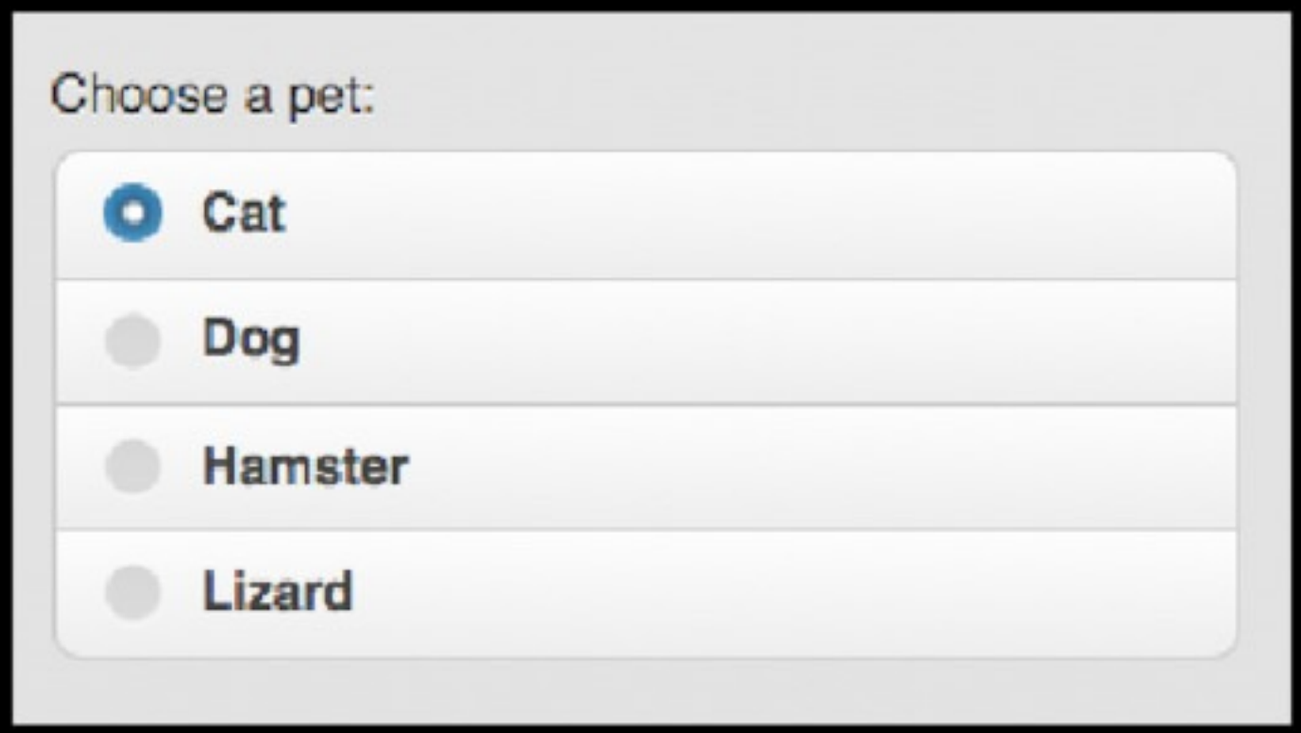

```
<div data-role="fieldcontain">
   <label for="slider">Input slider:</label>
   <input type="range" name="slider"
   id="slider" value="50" min="0" max="100" />
</div>
```

Flip Switch Slider




```
<label for="switch">Flip toggle:</label>
<select name="switch" id="switch" data-role="slider">
    <option value="off">Off</option>
    <option value="on">On</option>
</select>
```

Radio button set



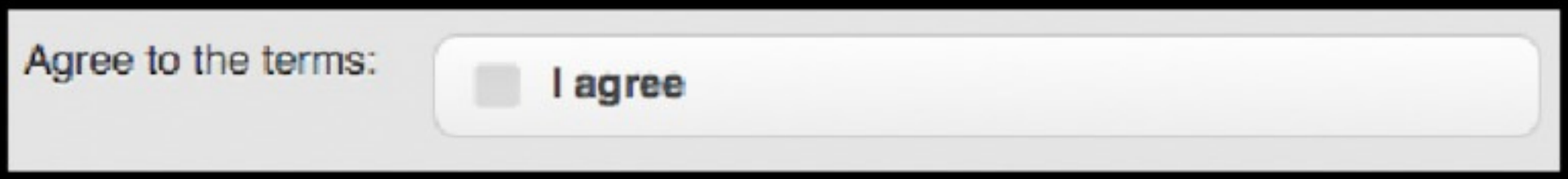
Radio button set

```
<fieldset data-role="controlgroup">
    <legend>Choose a pet:</legend>
        <input type="radio" name="r1" id="r1" value="choice-1"
        checked="checked" />
        <label for="r1">Cat</label>

        <input type="radio" name="r1" id="r2" value="choice-2" />
        <label for="r2">Dog</label>

        <input type="radio" name="r1" id="r3" value="choice-3" />
        <label for="r3">Hampster</label>

        <input type="radio" name="r1" id="r4" value="choice-4" />
        <label for="r4">Lizard</label>
</fieldset>
```

Horizontal Set


```
<fieldset data-role="controlgroup" data-type="horizontal">
```

Checkboxes




```
<div data-role="fieldcontain">
    <fieldset data-role="controlgroup">
      <legend>Agree to the terms:</legend>
      <input type="checkbox" name="checkbox-1"
       id="checkbox-1" class="custom" />
      <label for="checkbox-1">I agree</label>
    </fieldset>
</div>
```

Checkboxes

Choose as many snacks as you'd like:

- [ ] Cheetos
- [x] Doritos
- [ ] Fritos
- [x] Sun Chips

Font styling:

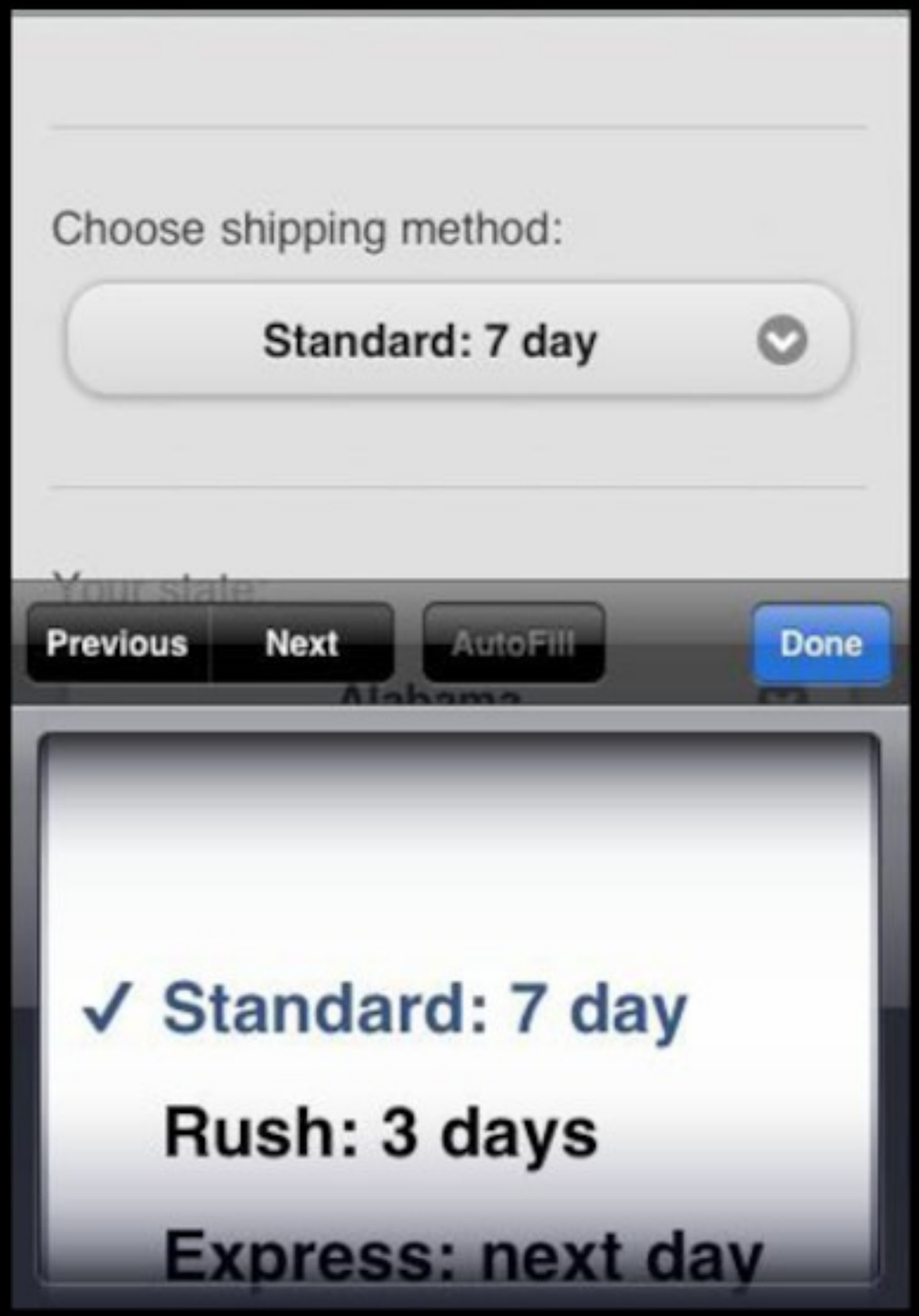
b | *i* | u

Extras - Select Menu

Select menu

```
<label for="select-choice-1">Choose shipping method:</label>
<select name="select-choice-1" id="select-choice-1">
    <option value="standard">Standard: 7 day</option>
    <option value="rush">Rush: 3 days</option>
    <option value="express">Express: next day</option>
    <option value="overnight">Overnight</option>
</select>
```

Default: native



data-native-menu="false"

If there is a small number of options that will fit on the device's screen, it will appear as a small overlay with a pop transition.

Choose shipping method:

Standard: 7 day

If there is a select menu with too many options to show on the device's screen, the framework will automatically create a new "page" populated with a standard list view that contains all the options.

If there is a small number of options that will fit on the device's screen, it will appear as a small overlay with a pop transition.

Standard: 7 day

Rush: 3 days

Express: next day

Overnight

automatically create a new "page" populated with a standard list view that contains all the options.

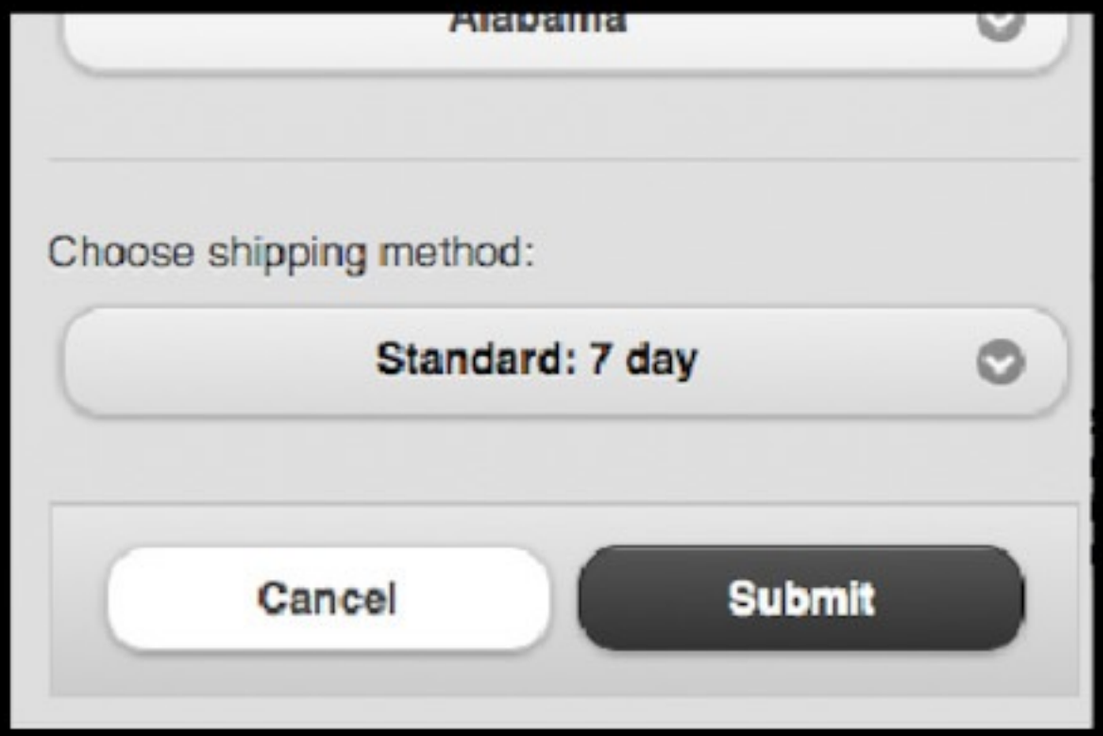
Long Select Menu

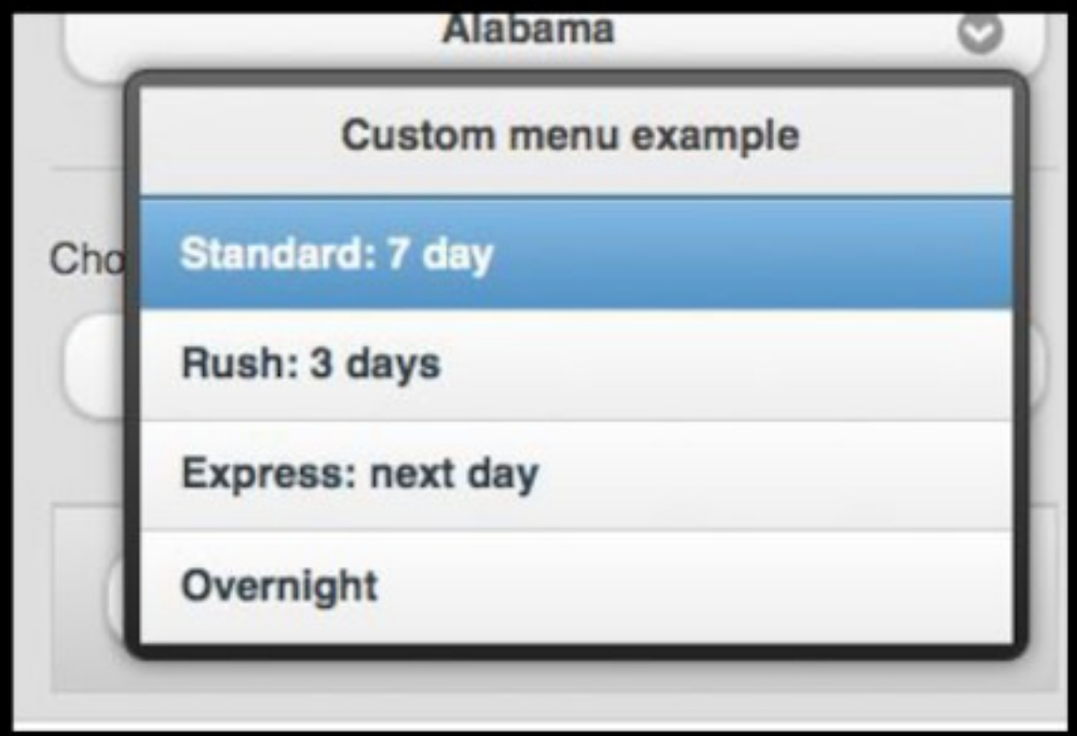
Displayed as a pop up window

Placeholder option

- An option with:
  - No (or empty) value attribute
  - No text node
  - data-placeholder="true"

Select Menu


Alabama
Choose shipping method:
Standard: 7 day
Cancel
Submit


Alabama
Custom menu example
Standard: 7 day
Rush: 3 days
Express: next day
Overnight

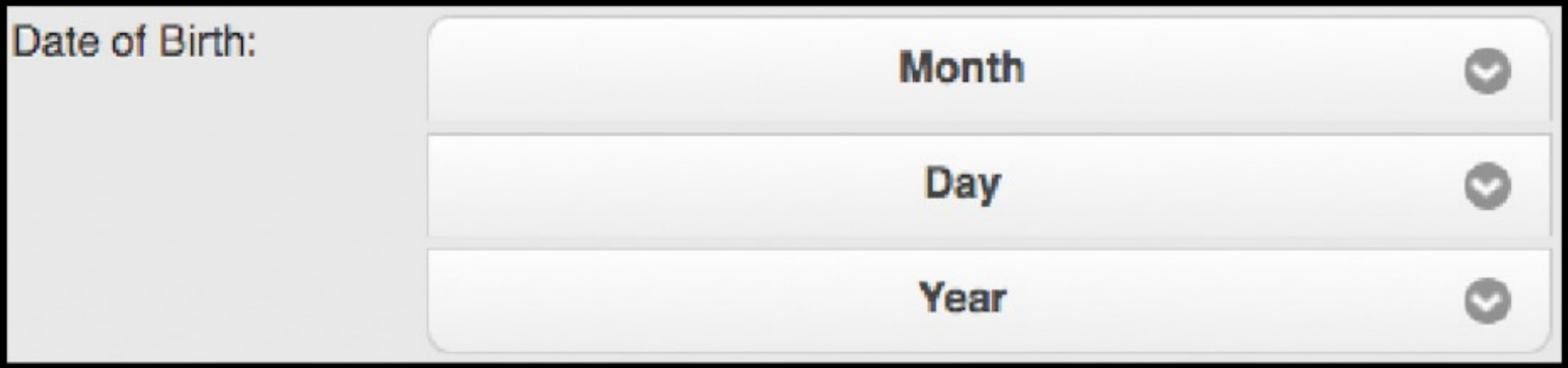
Long Select Menu

Long native menu

Vertically grouped select inputs

- To create a grouped set of select inputs, first add select and a corresponding label.
- Set the for attribute of the label to match the ID of the select so they are semantically associated.
- Because the label element will be associated with each individual select input, we recommend wrapping the selects in a fieldset element that has a legend which acts as the combined label for the grouped inputs.
- Lastly, one needs to wrap the fieldset in a div with data-role="controlgroup" attribute, so it can be styled as a group.

Vertically grouped select inputs



Vertically grouped select inputs

```
<div  class="ui-field-contain">
      <fieldset data-role="controlgroup">
      <legend>Date of Birth:</legend>
      <label for="select-choice-month">Month</label>
      <select name="select-choice-month" id="select-choice-month">
            <option>Month</option>
            <option value="jan">January</option>
            <!-- etc. -->
      </select>
      <!--...Day... -->
      <label for="select-choice-year">Year</label>
      <select name="select-choice-year" id="select-choice-year">
            <option>Year</option>
            <option value="2011">2011</option>
            <!-- etc. -->
      </select>
</fieldset>
</div>
```

Horizontally grouped selects

- Select inputs can also be used for grouped sets with more than one related selections. To make a horizontal button set, add the data-type="horizontal" to the fieldset.
- Note that the buttons which trigger the select will resize depending on the currently selected option's value. Note that browsers without support for display: inline-block; will group the selects vertically, as above.

Horizontally grouped selects

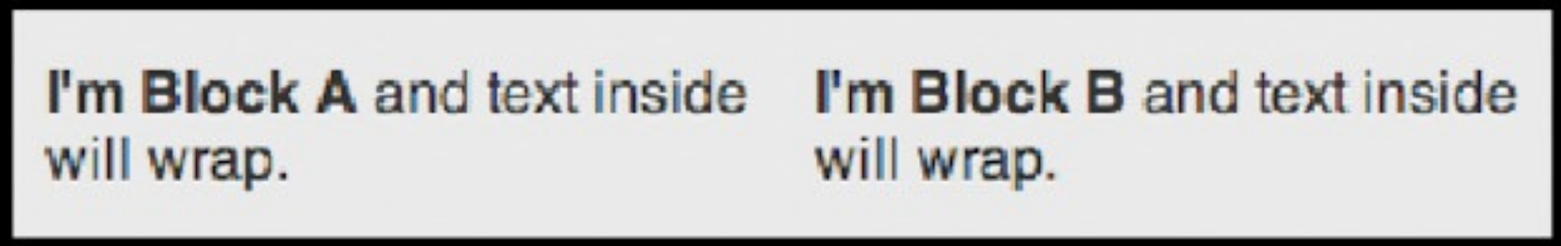

```
<fieldset data-role="controlgroup" data-type="horizontal">
```

Extras - Layout Grids

Layout grids (columns)

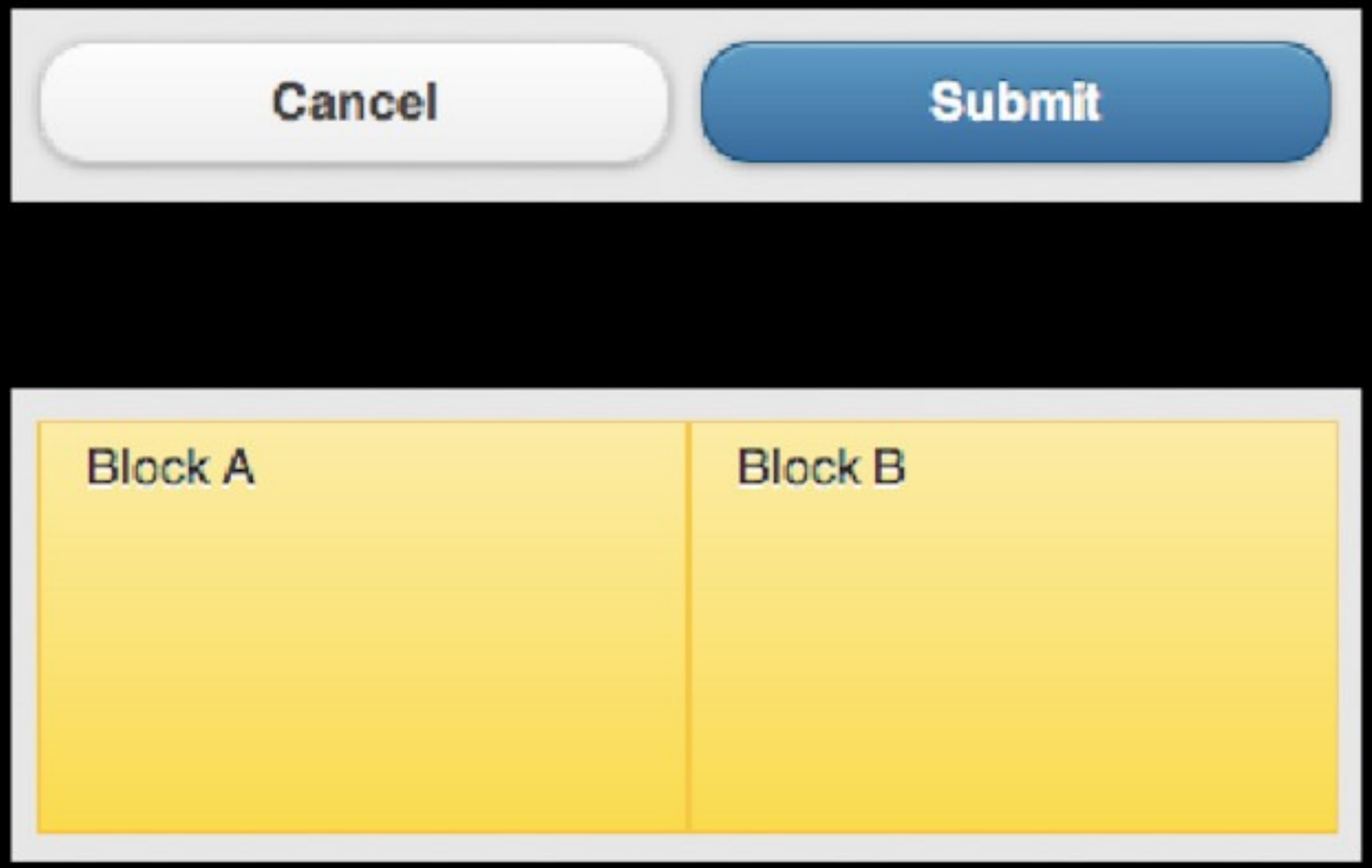
- The jQuery Mobile framework provides a simple way to build CSS-based columns through a block style class convention called `ui-grid`.
- Grids are 100% width, completely invisible (no borders or backgrounds) and don't have padding or margins, so they shouldn't interfere with the styles of elements placed inside them.
- Four preset configurations layout:
  - two-column - using the class of `ui-grid-a`
  - three-column - using the class of `ui-grid-b`
  - four-column - using the class of `ui-grid-c`
  - five-column - using the class of `ui-grid-d`

Layout grids (columns)

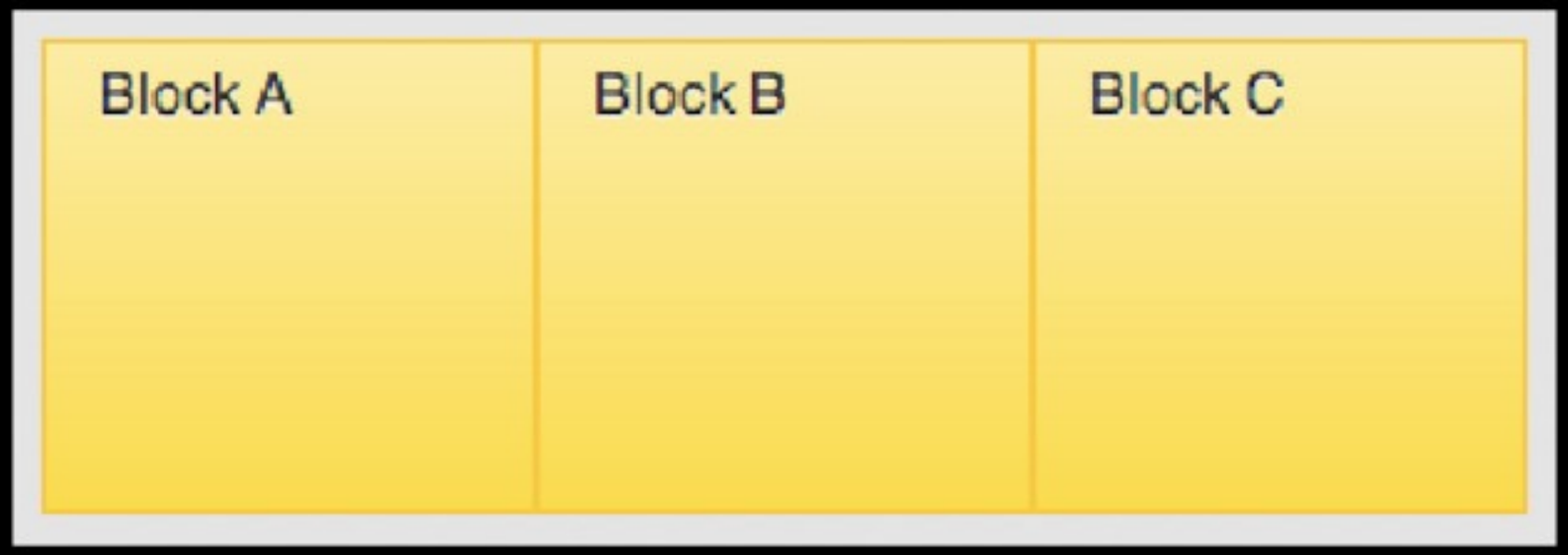



```
<div class="ui-grid-a">
    <div class="ui-block-a">
        <strong>I'm Block A</strong> and text inside will wrap
    </div>
    <div class="ui-block-b">
        <strong>I'm Block B</strong> and text inside will wrap
    </div>
</div><!-- /grid-a -->
```

Layout grids (columns)



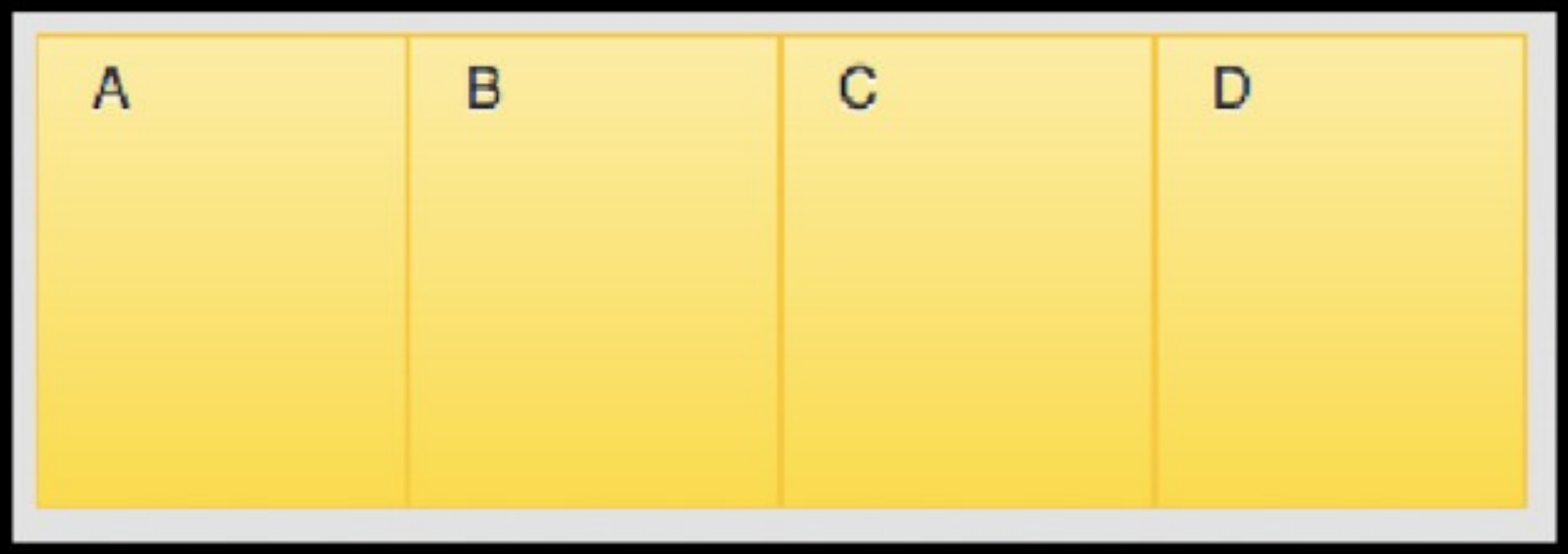
3 and 4-column grids



Hmm

No

Yes



5-column grids

Multiple row grids

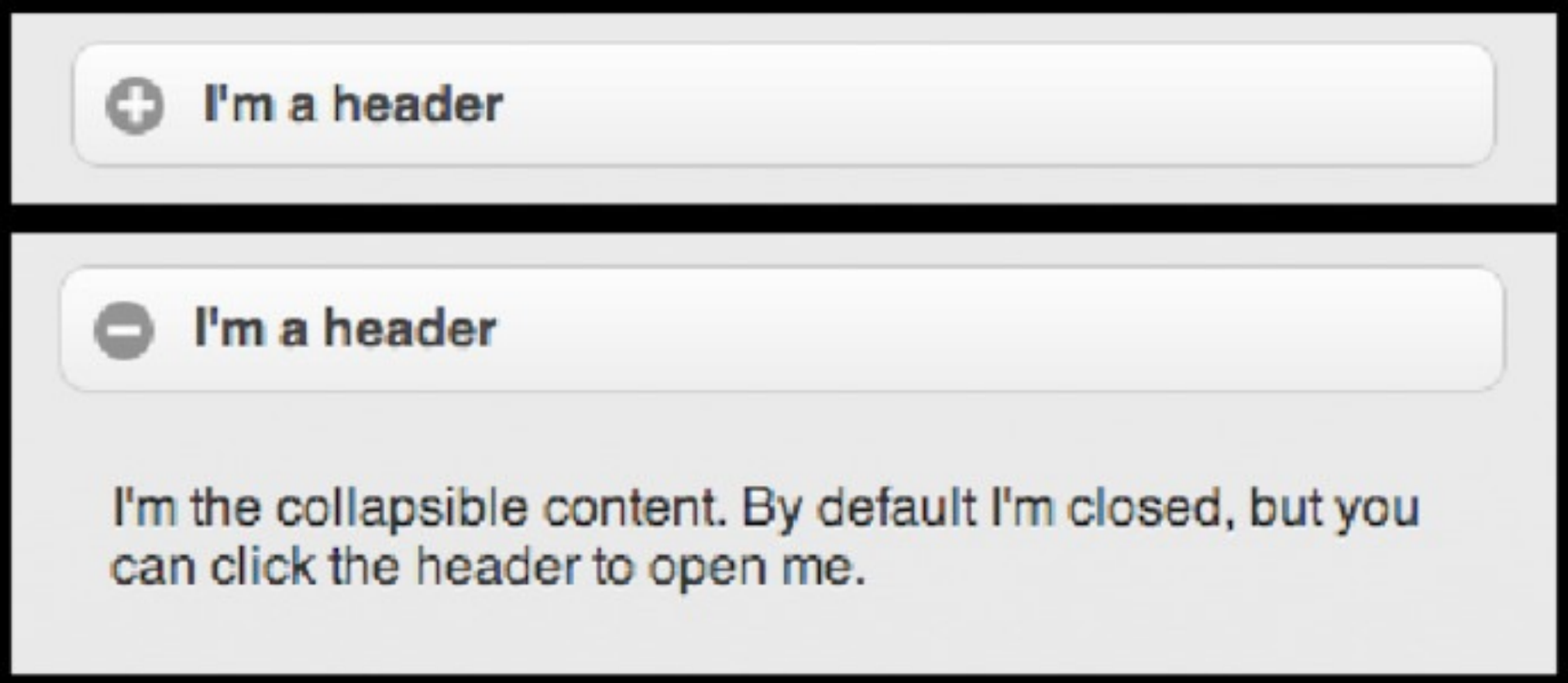
A	B	C
A	B	C
A	B	C

Collapsible content markup

- To create a collapsible block of content, create a container and add the data-role="collapsible" attribute.
- Using data-content-theme attribute allows you to set a theme for the content of the collapsible. (set it to “none” to have no theme)
- Directly inside this container, add any header element (H1-H6). The framework will style the header to look like a clickable button and add a "+" icon to the left to indicate it's expandable.
- After the header, add any HTML markup you want to be collapsible. The framework will wrap this markup in a container that will be hidden/shown when the heading is clicked.
- By default, the content will be collapsed.

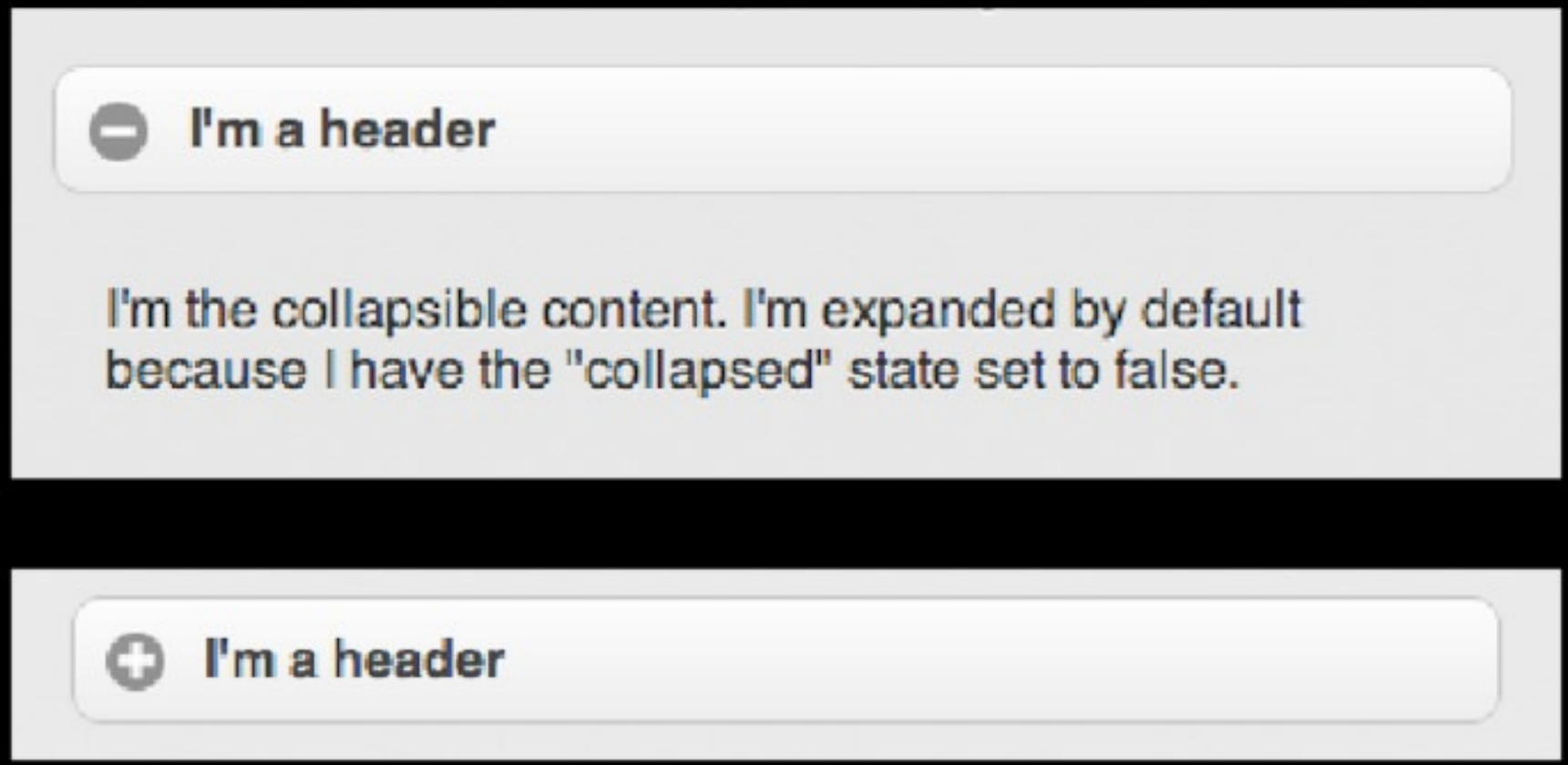
Collapsible content markup

```
<div data-role="collapsible” data-content-theme=“false”>
    <h3>I'm a header</h3>
    <p>I'm the collapsible content. By default I'm closed,
but you can click the header to open me.</p>
</div>
```



Collapsible content blocks

```
<div data-role="collapsible" data-collapsed="false” data-
content-theme=“false”>
```



Theming Collapsibles

- To provide a stronger visual connection between the collapsible header and content, add the data-content-theme attribute to the wrapper and specify a theme swatch letter. If this is not set, it will inherit the theme from its parent.

- This will apply the swatch's border and *flat* background color to the content block and changes the corner rounding to square off the bottom of the header and round the bottom of the content block instead to visually group these elements.

Collapsible content blocks


```
<div data-role="collapsible”> (or with data-content-theme=“a”)
    <h3>Header swatch A</h3>
    <p>I'm the collapsible content with a themed content block
set to "C".</p>
</div>
```

Collapsible content blocks

I'm a header

I'm the collapsible content. By default I'm open and displayed on the page, but you can click the header to hide me.

I'm a nested collapsible with a child collapsible

I'm a child collapsible.

Nested inside again.

Three levels deep now.

Section 3: Form elements

Section 4: Collapsed list

I'm a header

I'm the collapsible content. By default I'm open and displayed on the page, but you can click the header to hide me.

I'm a nested collapsible with a child collapsible

Section 3: Form elements

Textarea:

Input slider:

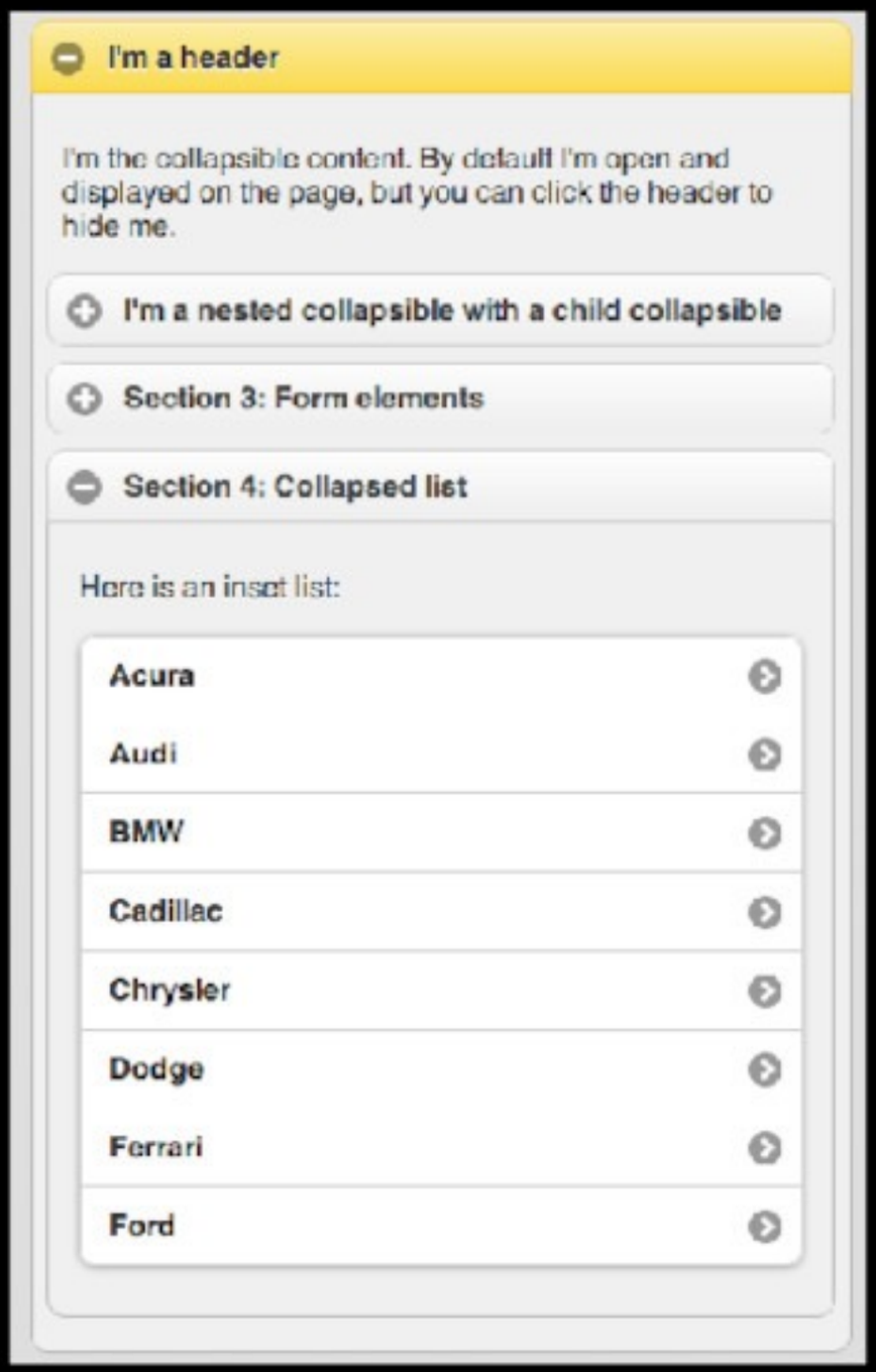
0

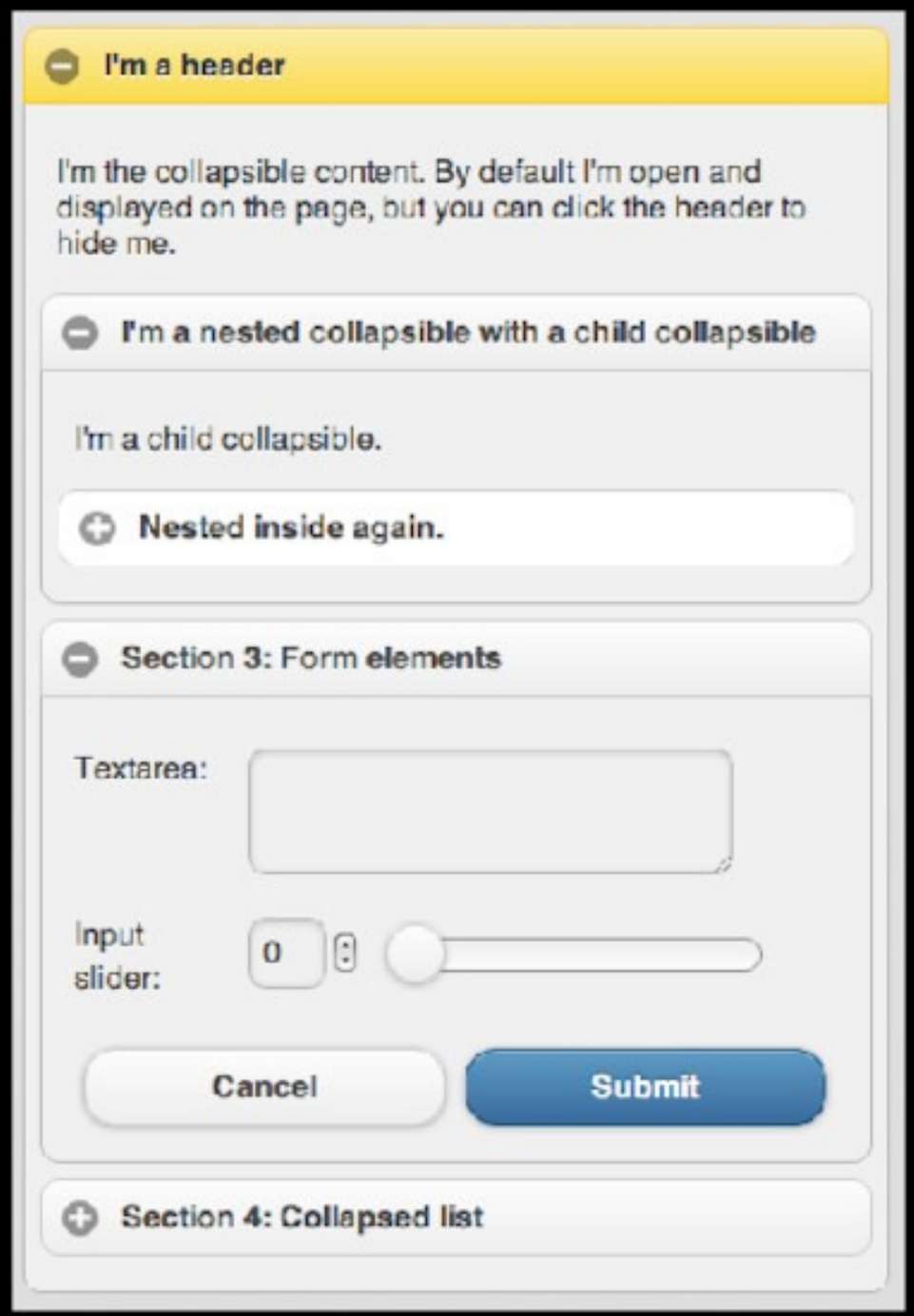
Cancel

Submit

Section 4: Collapsed list

Collapsible content blocks



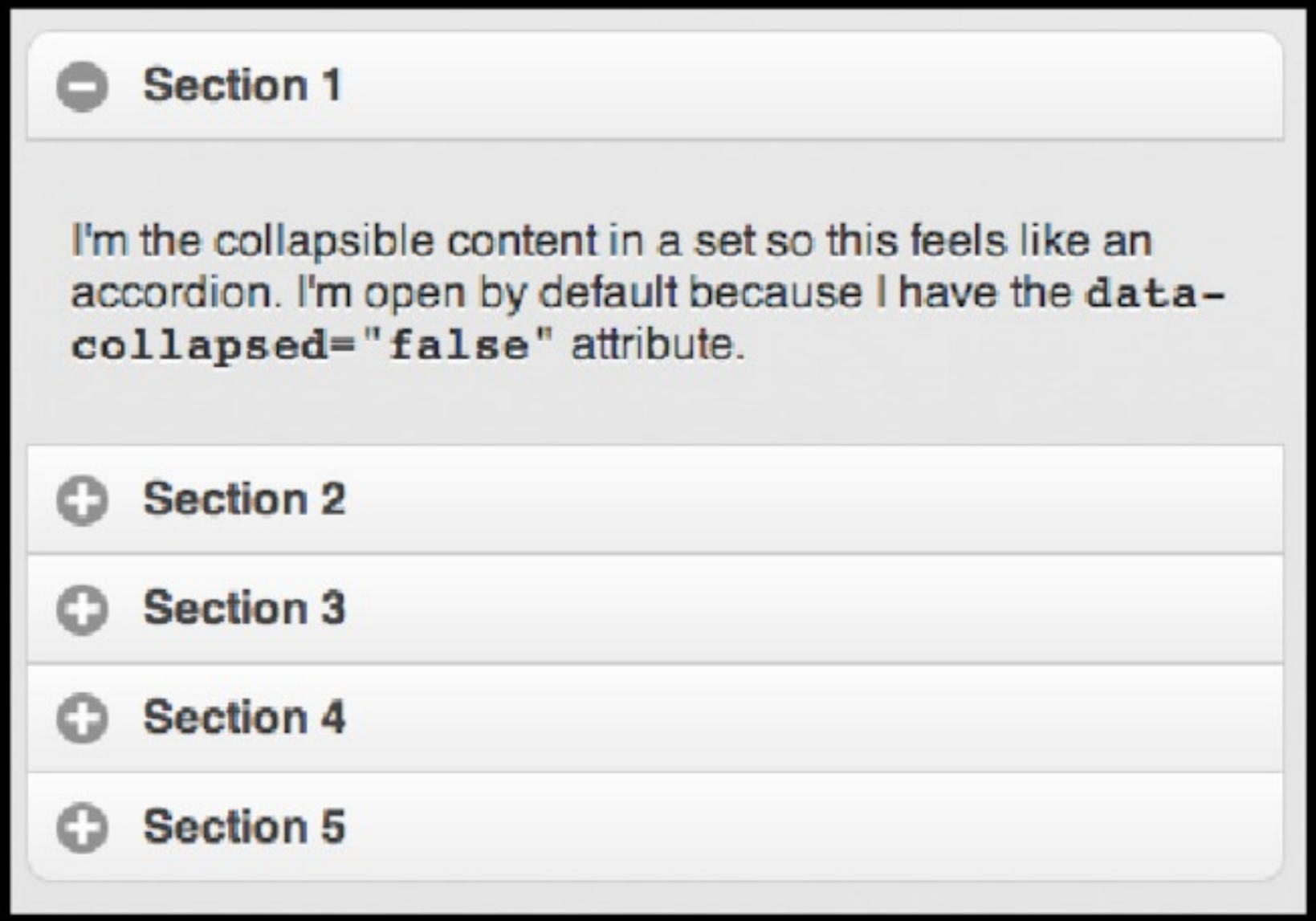


Collapsible set

- Collapsible sets start with the exact same markup as individual collapsibles. By adding a parent wrapper with a data-role="collapsible-set" attribute around a number of collapsibles, the framework will style these to looks like a visually grouped widget and make it behave like an accordion so only one section can be open at a time.

- By default, all the sections will be collapsed. To set a section to be open when the page loads, add the data-collapsed="false" attribute to the heading of the section you want expanded.

Collapsible set

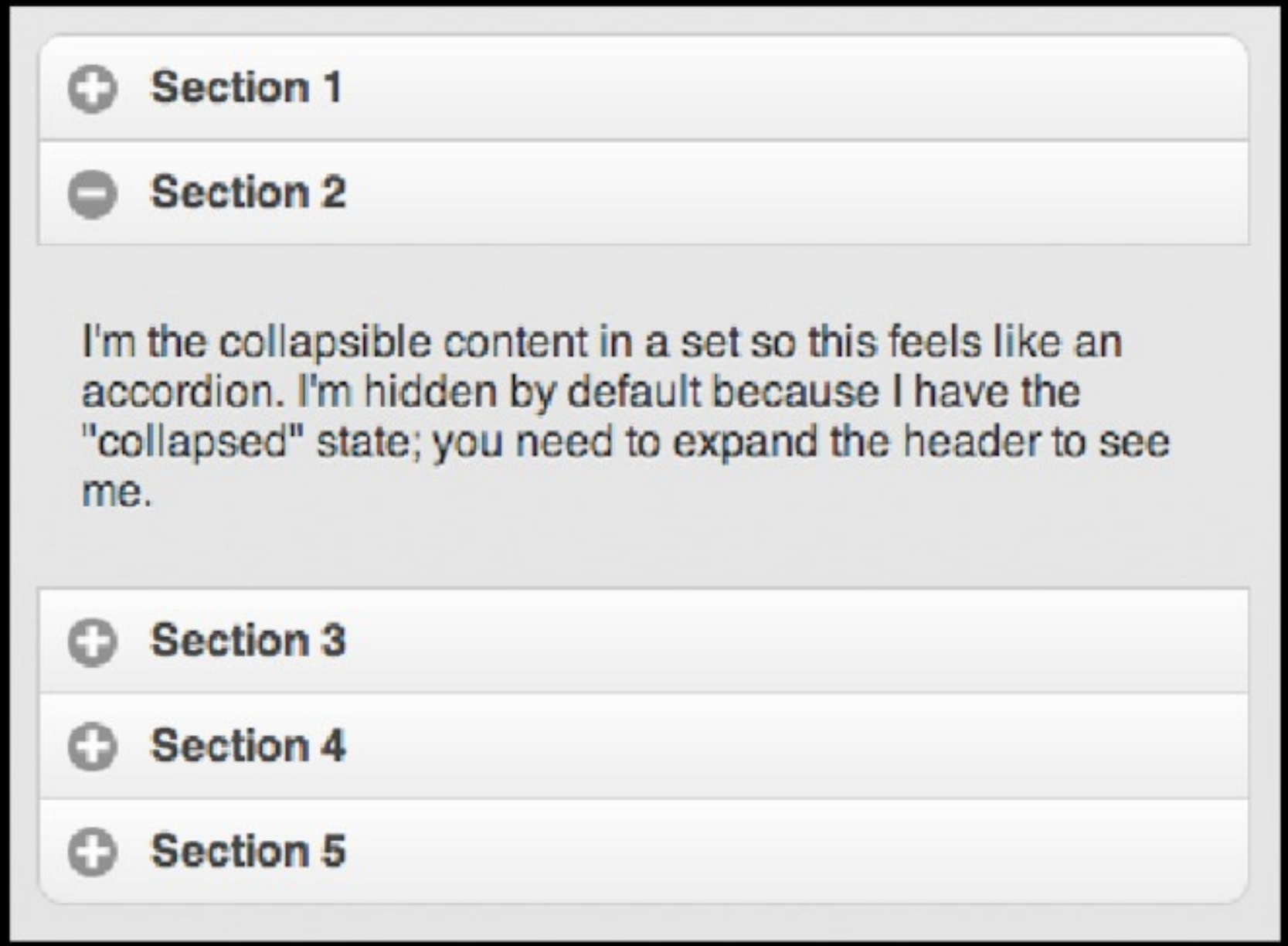


Collapsible set

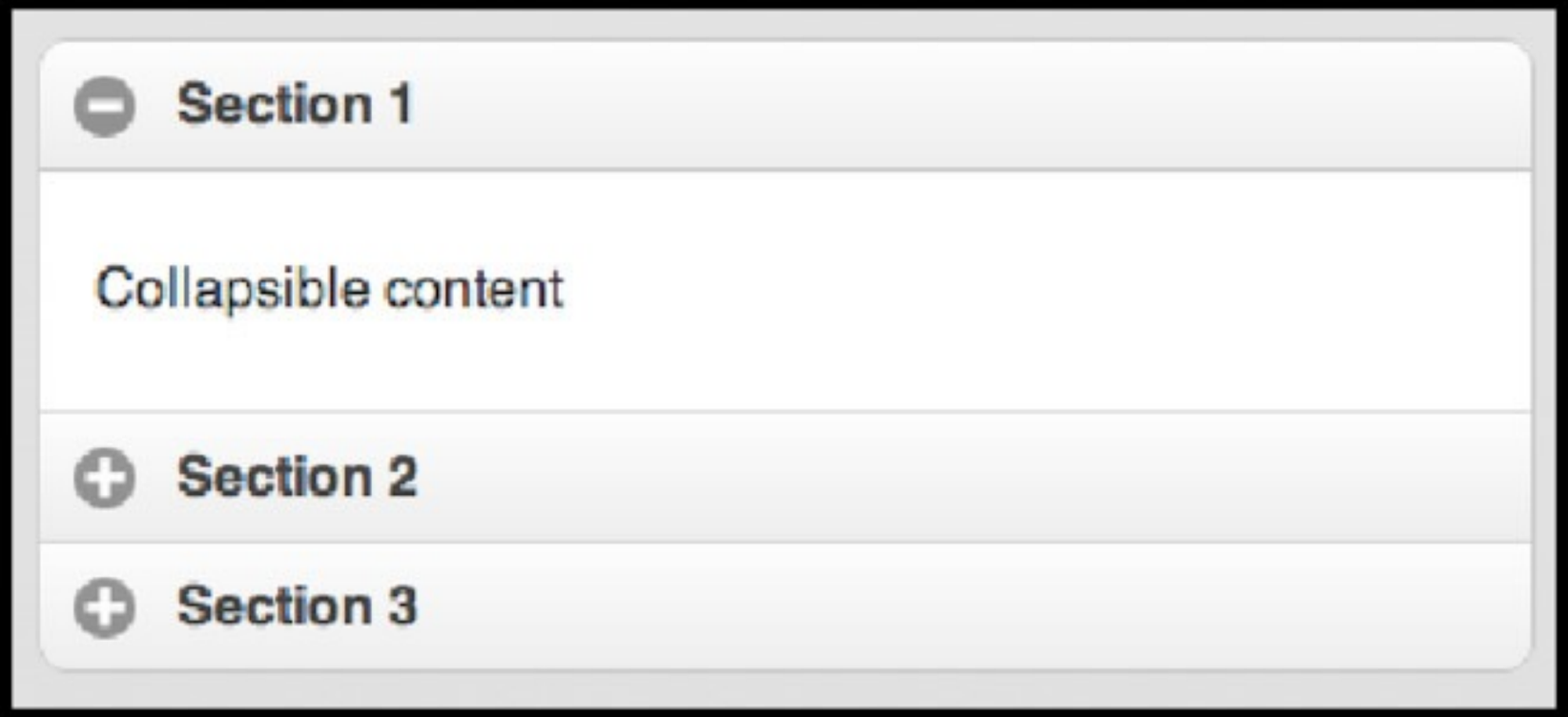
```
<div data-role="collapsible-set">

    <div data-role="collapsible" data-collapsed="false">
    <h3>Section 1</h3>
    <p>I'm the collapsible set content for section B.</p>
    </div>

    <div data-role="collapsible">
    <h3>Section 2</h3>
    <p>I'm the collapsible set content for section B.</p>
    </div>

</div>
```

Collapsible set

Collapsible set

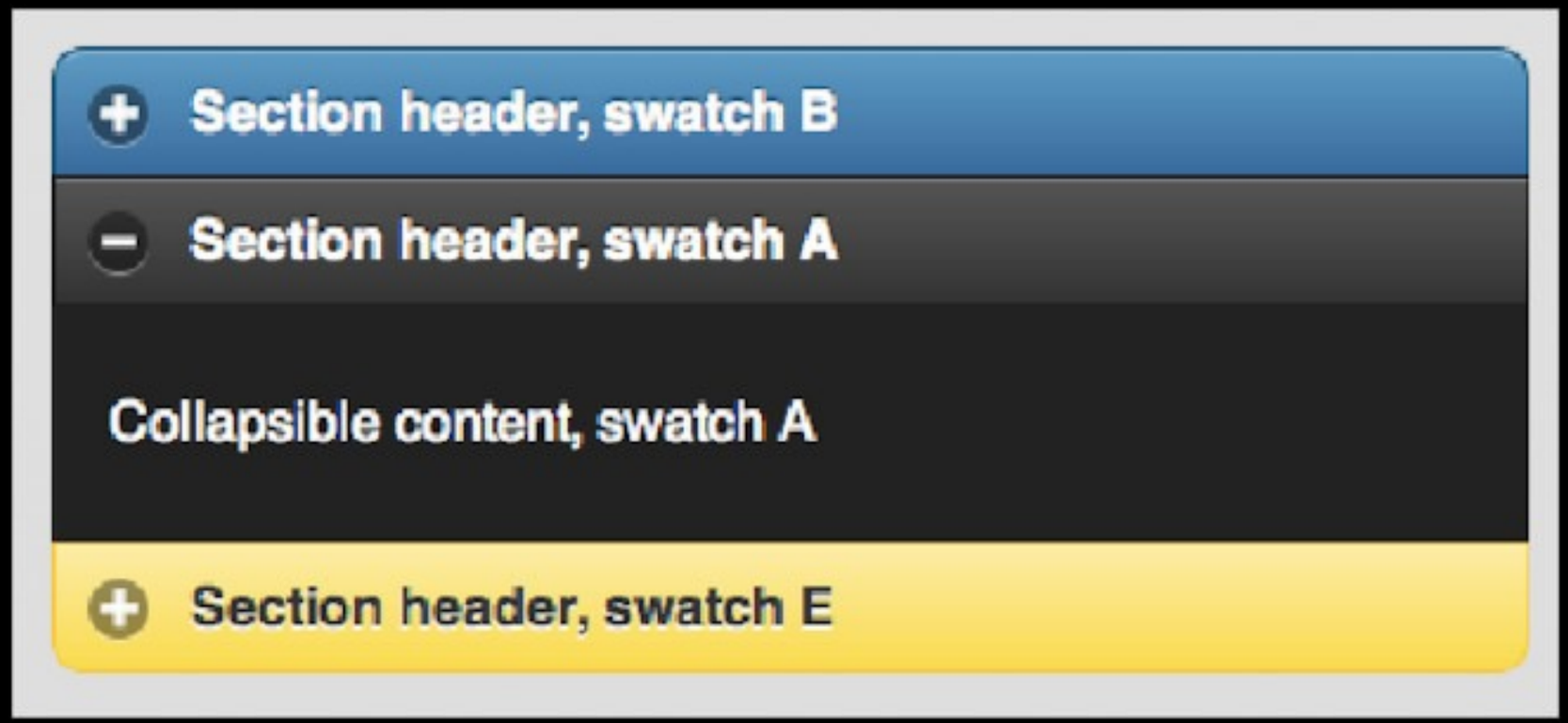


Collapsible set

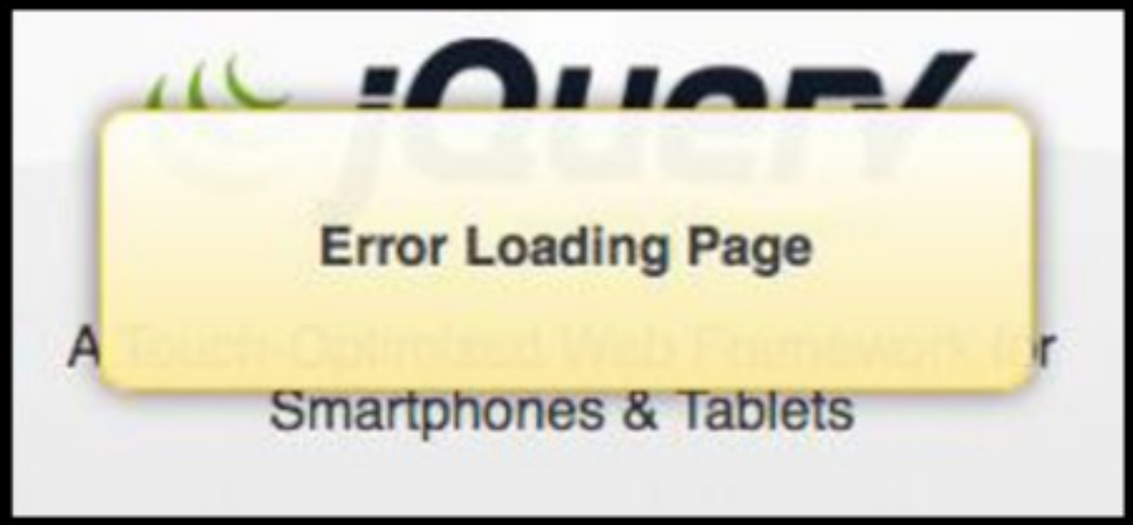



```
<div data-role="collapsible-set" data-theme=“a"
data-content-theme=“a">
```

Collapsible set



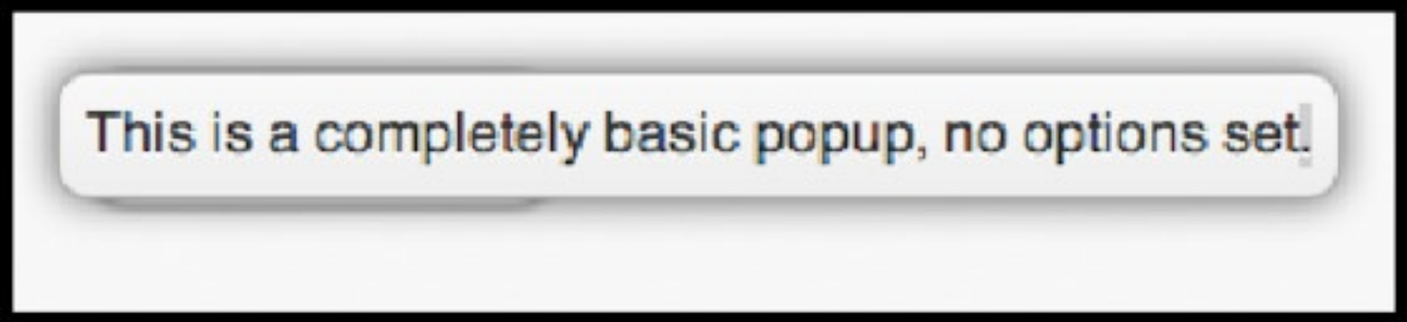
Custom messages



Extras - Popups

Popups

```
<a href="#popupBasic" data-rel="popup">Open Popup</a>

<div data-role="popup" id="popupBasic">
    <p>This is a completely basic popup, no options
set.<p>
</div>
```



Popups

```
<a href="#positionWindow" data-rel="popup" data-position-
to="window">

<div data-role="popup" id="positionWindow">
    <p>I am positioned to the window.</p>
</div>
```

Show popup demo

Menu

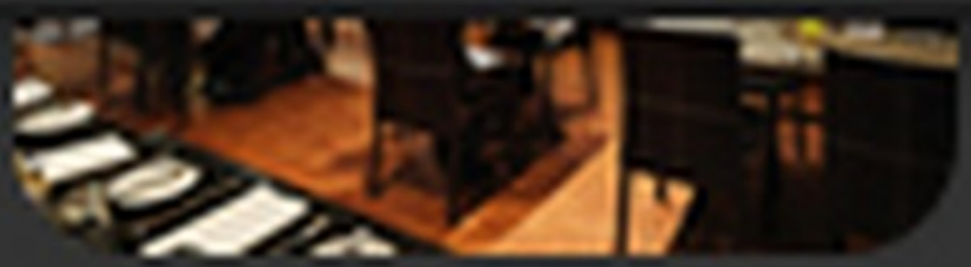
The Ritz

Classic Fren

Menu

action

Sort by cuisine



Classic Fren

Panels

- A panel must be a sibling to the header, content and footer elements inside a jQuery Mobile page. You can add the panel markup either *before* or *after* these elements, but not in between.

```
<div data-role="page">

    <div data-role="panel" id="mypanel">
        <!-- panel content goes here -->
    </div>

    <div data-role="header">
    </div>

    <div role="main" class="ui-content">
    </div>

    <div data-role="footer">
    </div>

</div><!-- /page -->
```

Panels

- The **position** of the panel on the screen is set by the data-position attribute. The defaults to left, meaning it will appear from the left edge of the screen. Specify data-position="right" for it to appear from the right edge instead.
- The **display mode** of the panel is set by the data-display attribute. The defaults to reveal, meaning the panel will sit under the page and reveal as the page slides away. Specify data-display="overlay" for the panel to appear on top of the page contents. A third mode, data-display="push" animates both the panel and page at the same time.
- The api docs: http://api.jquerymobile.com/panel/

Extras - JQuery Interaction

codeschool Try JQuery
http://try.jquery.com/

Useful links

- Api Documentation:
  - http://api.jquerymobile.com
- Demo Centre:
  - http://demos.jquerymobile.com/1.4.5/

CW1 - Image Carousel

Various Methods...
fotorama
nivoslider
owl carousel
jcarousel

CW1 - JCarousel

Basic - JCarousel
http://sorgalla.com/jcarousel/

CSS and JS in head
DIV with class of jcarousel-wrapper
pictures from xml
links for previous and next with class of jcarousel controls
view-source:http://sorgalla.com/jcarousel/examples/basic/

CW1 - RWD

Various Methods...

CW1 - XSLTPROC

To generate an HTML file for CW1 submission.

Previously we have been viewing the xml, which gets styled by xsl.

Creating an HTML file and running your JQuery Mobile application in the iOS Simulator on the mac

1. Check your xml and xsl file do not generate any errors
2. Open a terminal on the mac (search for terminal in Spotlight)
3. Launch the terminal
4. Locate you project folder on the mac
5. In the terminal type cd
6. Then drag your folder to the terminal – this is place the absolute directory path to your project folder into the terminal
7. Type RETURN
8. Now type: xsltproc yourXMLfile.xml and press RETURN
9. You should see an your html file printed out onto the terminal (this is not yet in a file though)
10. Press the ↑ k nd back.
11. At the en ion.html and press RETU
12. This lled jq
13. L
14. T eed to launch
15.
16.
17. h'
18.
19. T
20. W white screen) then me button
21. You w aunch Safari
22. Locate y
23. Drag the file
24. You now should see y running in the iOS simulator.
25. You can change the device type and rotate the device by pressing ⌘➔

https://cf.nearpod.com/neareducation/new/Webpage/309166170/iconoriginal.pdf

Google Maps

Mobile Geolocation

With HTML 5 and Google Maps API

Google Maps Overview

1. Style - Map fullscreen options
2. Script - Map API with personal developer key
3. Script - Initialise Function for Map Object, Other Functions and Variables for Map Object.
4. Script - Map Object, Functions, ID, Original Variables
5. Script - DOMListener - Load, Initialise, Custom Functions.
6. DIV - Map Page or Div with Object iD

Geolocation

- Geolocation allows your visitors to share their current location.
- Depending on how they' re visiting your site, their location may be determined by any of the following:
  - IP address
  - wireless network connection
  - cell tower
  - GPS hardware on the device

Geolocation API Specification

- W3C http://dev.w3.org/geo/api/spec-source.html
- One point to note, as the W3C Geolocation spec states: " No guarantee is given that the API returns the device' s actual location."

Browser Support

- Geolocation is supported in:
  - Safari 5+
  - Chrome 5+
  - Firefox 3.5+
  - IE 9+
  - Opera 10.6+
  - iOS (Mobile Safari) 3.2+
  - Android 2.1+

Privacy Concerns

- Not everyone will want to share their location with you, as there are privacy concerns inherent to this information. Thus, your visitors must opt in to share their location.
- Nothing will be passed along to your site or web application unless the user agrees.
- The decision is made via a prompt at the top of the browser.

Blocking of the Geolocation Prompt in Chrome

- Be aware that Chrome may block your site from showing this prompt entirely if you're viewing your page locally, rather than from an internet server. If this happens, you'll see an icon in the address bar alerting you to it.
- There's no way around this at present, but you can either test your functionality in other browsers, or deploy your code to a testing server (this can be a local server on your machine, a virtual machine, or an actual internet server).

Google Map JavaScript API V3

- https://developers.google.com/maps/documentation/javascript/tutorial
- https://developers.google.com/maps/location-based-apps

Beware of map height!

- The map canvas needs a height to display on the page.
- CSS: height: 100%; means height is 100% of the element's parent => the parent needs a height also.

Controls

- The maps displayed through the Google Maps API contain UI elements to allow user interaction with the map. These elements are known as *controls* and you can include variations of these controls in your Google Maps API application. Alternatively, you can do nothing and let the Google Maps API handle all control behavior.
- https://developers.google.com/maps/documentation/javascript/controls

Overlays

- Overlays are objects on the map that are tied to latitude/longitude coordinates, so they move when you drag or zoom the map. Overlays reflect objects that you "add" to the map to designate points, lines, areas, or collections of objects.
- https://developers.google.com/maps/documentation/javascript/overlays

Geocoding

- Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers or position the map. (Google)
- https://developers.google.com/maps/documentation/javascript/geocoding

Get the user's current location

- https://developers.google.com/maps/documentation/javascript/examples/map-geolocation

Directions service

- You can calculate directions (using a variety of methods of transportation) by using the DirectionsService object. This object communicates with the Google Maps API Directions Service which receives direction requests and returns computed results. You may either handle these directions results yourself or use the DirectionsRenderer object to render these results.
- https://developers.google.com/maps/documentation/javascript/directions

Streetview

- Google Street View provides panoramic 360 degree views from designated roads throughout its coverage area. Street View's API coverage is the same as that for the Google Maps application (http://maps.google.com/)
- https://developers.google.com/maps/documentation/javascript/streetview