

Exercise: Asynchronous Programming

1. Forecaster

Write a program that **requests** a weather report **from a server** and **displays** it to the user.

Use the skeleton from the provided resources.

When the user writes the name of a location and clicks “**Get Weather**”, make a **GET** request to the server at address **`https://judgetests.firebaseio.com/locations.json`**. The response will be an array of objects, with the following structure:

```
{
  name: locationName,
  code: locationCode
}
```

Find the object, corresponding to the name that the user submitted in the input field with ID “**location**” and use its **code** value to make **two more GET requests**:

- For current conditions, make a request to:

`https://judgetests.firebaseio.com/forecast/today/{code}.json`

The response from the server will be an object with the following structure:

```
{
  name: locationName,
  forecast: { low: temp,
              high: temp,
              condition: condition }
}
```

- For a 3-day forecast, make a request to:

`https://judgetests.firebaseio.com/forecast/upcoming/{code}.json`

The response from the server will be an object with the following structure:

```
{
  name: locationName,
  forecast: [{ low: temp,
               high: temp,
               condition: condition }, ... ]
}
```

Use the information from these two objects to compose a forecast in HTML and insert it inside the page. Note that the **<div>** with ID “**forecast**” must be set to **visible**. See the examples for details.

If an **error** occurs (the server doesn’t respond or the location name cannot be found) or the data is not in the correct format, display “**Error**” in the **forecast section**.

Use the following codes for weather symbols:

- Sunny `☀ // ☀`
- Partly sunny `⛅ // ⛅`
- Overcast `☁ // ☁`
- Rain `☔ // ☔`
- Degrees `° // °`

Examples


When the app starts, the **forecast div** is **hidden**. When the user **enters a name** and **clicks** on the button **Get Weather**, the requests being.

Get Weather


```
><div id="request">...</div>
▼<div id="forecast" style="display:none">
  ▶<div id="current">...</div>
  ▶<div id="upcoming">...</div>
</div>
```


New YorkGet Weather


Current conditions

 New York, USA
8°/19°
Sunny

Three-day forecast

 6°/17°
Partly sunny

 3°/9°
Overcast

 3°/7°
Overcast

```

▶<div id="request">...</div>
▼<div id="forecast" style="display: block;">
  ▼<div id="current">
    <div class="label">Current conditions</div>
    ▼<div class="forecasts">
      <span class="condition symbol">☀</span>
      ▼<span class="condition">
        <span class="forecast-data">New York, USA</span>
        <span class="forecast-data">8°/19°</span>
        <span class="forecast-data">Sunny</span>
      </span>
    </div>
  </div>
  ▼<div id="upcoming">
    <div class="label">Three-day forecast</div>
    ▼<div class="forecast-info">
      ▼<span class="upcoming">
        <span class="symbol">☁</span>
        <span class="forecast-data">6°/17°</span>
        <span class="forecast-data">Partly sunny</span>
      </span>
      ▶<span class="upcoming">...</span>
      ▶<span class="upcoming">...</span>
    </div>
  </div>
</div>
</div>

```

Hints

The server at the address listed above will respond with valid data for location names "London", "New York" and "Barcelona".

2. Fisher Game

Each catch should have:

- **angler** - **string** representing the name of the person who caught the fish
- **weight** - **floating-point number** representing the weight of the fish in kilograms
- **species** - **string** representing the name of the fish species
- **location** - **string** representing the location where the fish was caught
- **bait** - **string** representing the bait used to catch the fish
- **captureTime** - **integer number** representing the time needed to catch the fish in minutes

HTML Template

Use the skeleton from the provided resources.

Attach handlers to the [Load], [Update], [Delete] and [Add] buttons, which make the appropriate GET, PUT, DELETE and POST requests.

You are given an example catch in the template to show you where and how to insert the catches. Notice that the **div** containing the catch has an attribute **data-id** that should store the **_id** of the entry given by Kinvey.

Create the following REST services to access your data:

- **List All Catches**
 - Endpoint - **`https://fisher-game.firebaseio.com/catches.json`**
 - Method: **GET**
 - Returns (**Object of objects**)
- **Create a New Catch**
 - Endpoint: **`https://fisher-game.firebaseio.com/catches.json`**
 - Method: **POST**
 - Request body (JSON): **`{"angler":"...", "weight":..., "species":"...", "location":"...", "bait":"...", "captureTime":...}`**
- **Update a Catch**
 - Endpoint: **`https://fisher-game.firebaseio.com/catches/{catchId}.json`**
 - Method: **PUT**
 - Request body (JSON): **`{"angler":"...", "weight":..., "species":"...", "location":"...", "bait":"...", "captureTime":...}`**
- **Delete a Catch**
 - Endpoint: **`https://fisher-game.firebaseio.com/catches/{catchId}.json`**
 - Method: **DELETE**
- Pressing the **[Load]** button should **list all** catches.
- Pressing the **[Update]** button should send a **PUT** request, updating the catch in firebase.
- Pressing the **[Delete]** button should delete the catch both from firebase and from the page.
- Pressing the **[Add]** button should submit a new catch with the values of the inputs in the fieldset with **id="addFrom"**.

Screenshots

The screenshot shows a web application titled "Biggest Catches". On the left, there is a large rectangular area labeled "Catches" with a horizontal line, intended for displaying a list of catches. On the right side, there is a vertical panel. At the top of this panel is a button labeled "LOAD". Below it is a section titled "Add Catch" which contains a form with several input fields: "Angler" (with "Paulo Amorim" entered), "Weight" (with "636" entered), "Species" (with "Atlantic Blue Marlin" entered), "Location" (with "Vitynia, Brazil" entered), "Bait" (with "trolled ping" entered), and "Capture Time" (with "80" entered). At the bottom of this form is a button labeled "ADD".

Biggest Catches

Catches

Angler	<input type="text" value="Paulo Amorim"/>
Weight	<input type="text" value="636"/>
Species	<input type="text" value="Atlantic Blue Marlin"/>
Location	<input type="text" value="Vitona, Brazil"/>
Bait	<input type="text" value="trolled pink"/>
Capture Time	<input type="text" value="80"/>
<input type="button" value="UPDATE"/>	
<input type="button" value="DELETE"/>	

LOAD

Add Catch	
Angler	<input type="text"/>
Weight	<input type="text"/>
Species	<input type="text"/>
Location	<input type="text"/>
Bait	<input type="text"/>
Capture Time	<input type="text"/>
<input type="button" value="ADD"/>	

SoftwareHQ