GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-12-resources` repo to your computer
2. Open the `11-advanced-apis > starter-code` folder in your code editor
3. If you haven't already,
   - download Postman from https://getpostman.com
   - sign up for a Flickr account at https://flickr.com (or sign in with an existing Yahoo account)

LEARNING OBJECTIVES

At the end of this class, you will be able to

- Request data from a web service.
- Implement the geolocation API to request a location.
- Use Postman to construct and test an API request.
- Process a third-party API response and share location data on your website.
- Search documentation needed to make and customize third-party API requests.

AGENDA

- Configure Flickr account
- Implement geolocation
- Set up Postman
- Create and send API call
- Handle API response

WEEKLY OVERVIEW

WEEK 6	Asynchronous JS & callbacks / Advanced APIs
WEEK 7	Project 2 Lab / Prototypal inheritance
WEEK 8	Closures & the Module Pattern / CRUD & Firebase

EXIT TICKET QUESTIONS

1. Where does the event loop fit into this lesson?
2. I am assuming asynchronous code is the same as when people discuss parallelizing code?
3. I don't completely get the "promise" concept.
4. I'd like to see some additional web applications that utilize the concepts that we learned today
5. how to use the document ready stuff
6. I get confused by what names are "standard" JS (like 'response') and what names we decide

Suggestion: Walking through some of the more complicated homework solutions in depth

Promises & Fetch

PROMISES

traditional callback:

```
doSomething(successCallback, failureCallback);
```

callback using a promise:

```
doSomething().then(
  // work with result
).catch(
  // handle error
);
```

MULTIPLE CALLBACKS — TRADITIONAL CODE

```
doSomething(function(result) {
  doSomethingElse(result, function(newResult) {
    doThirdThing(newResult, function(finalResult) {
      console.log('Got the final result: ' + finalResult);
    }, failureCallback);
  }, failureCallback);
}, failureCallback);
```

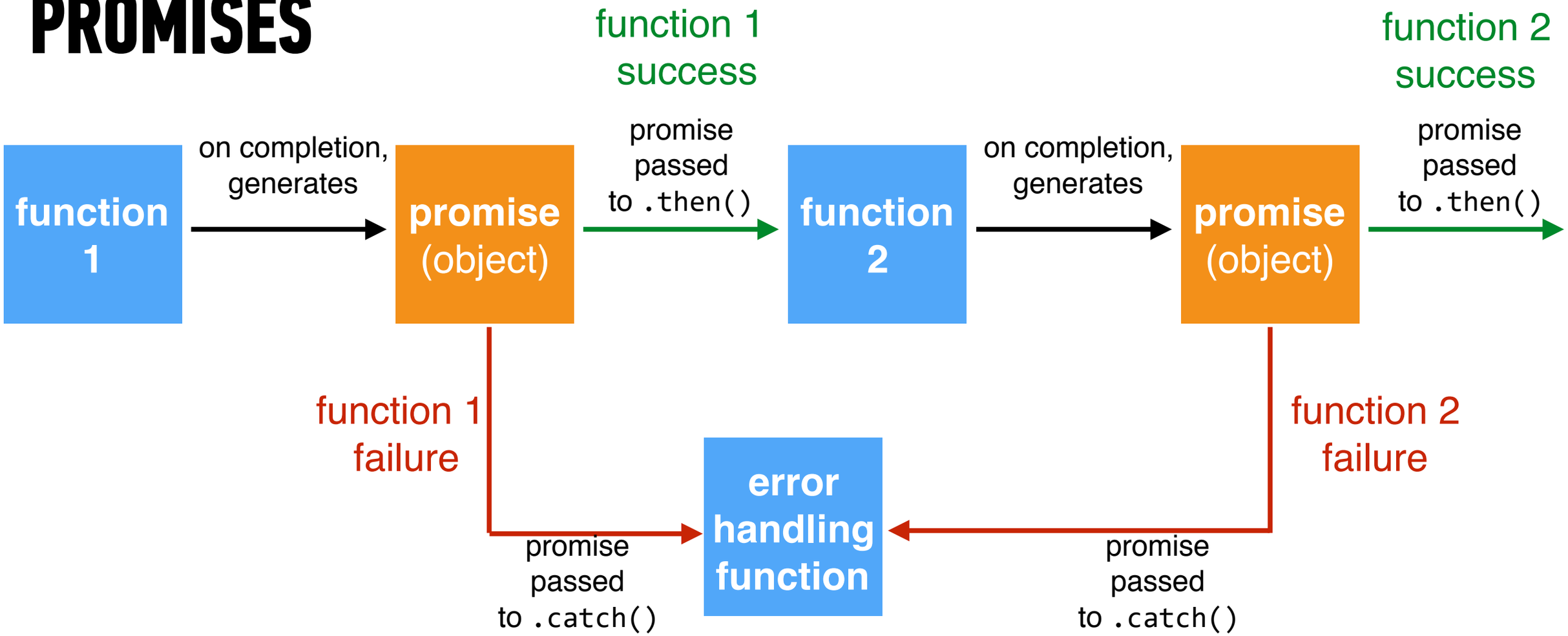
MULTIPLE CALLBACKS WITH PROMISES

```
doSomething().then(function(result) {
  return doSomethingElse(result);
})
.then(function(newResult) {
  return doThirdThing(newResult);
})
.then(function(finalResult) {
  console.log('Got the final result: ' + finalResult);
})
.catch(function(error) {
  console.log(‘There was an error');
});
```

ERROR HANDLING WITH PROMISES

```
doSomething().then(function(result) {
  return doSomethingElse(result);
})
.then(function(newResult) {
  return doThirdThing(newResult);
})
.then(function(finalResult) {
  console.log('Got the final result: ' + finalResult);
})
.catch(function(error) {
  console.log(‘There was an error');
});
```

PROMISES


function 1 success
function 2 success
function 1
on completion, generates
promise (object)
promise passed to .then()
function 2
on completion, generates
promise (object)
promise passed to .then()
function 1 failure
function 2 failure
error handling function
promise passed to .catch()
promise passed to .catch()

FETCH

```
fetch(url).then(function(response) {
  if(response.ok) {
    return response.json();
  } else {
  throw 'Network response was not ok.’;
  }
}).then(function(data) {
  // DOM manipulation
}).catch(function(error) {
  // handle lack of data in UI
});
```

ADVANCED APIS

BUILDING OUR APP

1. Configure our systems for development and testing
2. Get user's location
3. Create request to 500px with user's location info
4. Parse API response and add returned images to view

BUILDING OUR APP

Our app

2

- Get user's location

3

- Create request containing user's location info

flickr.com server

- Parse API response
- Add returned images to view

4

ENDPOINTS

- Examples from openweathermap.org

By geographic coordinates

API call:

api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}

Parameters:

lat, lon coordinates of the location of your interest

By city name

API call:

api.openweathermap.org/data/2.5/weather?q={city name}

api.openweathermap.org/data/2.5/weather?q={city name},{country code}

By ZIP code

Description:

Please note if country is not specified then the search works for USA as a default.

API call:

api.openweathermap.org/data/2.5/weather?zip={zip code},{country code}

EXERCISE

OBJECTIVE

- Search documentation needed to make and customize third-party API requests.

TIMING

4 min

1. Read the documentation for at least 2 endpoints ("API methods") from the list at https://www.flickr.com/services/api/
2. Identify an endpoint that will let us find photos based on a user's location.

Get User's Location

Call the Flickr endpoint

Handle the Response

EXERCISE

OBJECTIVE

- Process a third-party API response and share location data on your website.

TIMING

15 min

1. Create a handleResponseSuccess callback function to do the following:
   - Iterate through your response data, creating an `img` element each time with the given image URL from the API.
   - Add the class `image` to the `img` element
   - Append the new `img` element to the element with the class `images`, which already exists in the HTML.

Customize Search Results

EXERCISE

OBJECTIVE

- Search documentation needed to make and customize third-party API requests.

TIMING

until 9:20

Search the API documentation as necessary to modify your API request to do the following:

- Return 30 photos instead of the default 100
- Sort results by relevance

Bonus 1: Return URLs for larger images (Hint: Check out the extras argument at https://www.flickr.com/services/api/flickr.photos.search.html and look at the Size Suffixes section at https://www.flickr.com/services/api/misc.urls.html).

Bonus 2: Instead of landscapes, return photos from a different category (see popular tags at https://www.flickr.com/photos/tags/)

Exit Tickets!

(Class #11)

LEARNING OBJECTIVES - REVIEW

- Request data from a web service.
- Implement the geolocation API to request a location.
- Use Postman to construct and test an API request.
- Process a third-party API response and share location data on your website.
- Search documentation needed to make and customize third-party API requests.

NEXT CLASS PREVIEW

In-class lab: Feedr

- Familiarize yourself with the API documentation for news sources.
- Fork and clone your starter code.
- Strategize ways to hide the loader and replace the content of the `#main` container with that of the API.
- Look up other news sources that might be useful for the project.

Q&A