GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Remove your API keys from your homework, then submit it and create a pull request
2. Pull changes from the `svodnik/JS-SF-10-resources` repo to your computer
3. Open the `12-advanced-apis > starter-code` folder in your code editor

LEARNING OBJECTIVES

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

- Generate API specific events and request data from a web service.
- Implement the geolocation API to request a location.
- Process a third-party API response and share location data on your website.
- Make a request and ask another program or script to do something.
- Search documentation needed to make and customize third-party API requests.

AGENDA

- Configure 500px account and tools
- Implement authorization
- Implement geolocation
- Create and send API call
- Handle API response

WEEKLY OVERVIEW

WEEK 8	Advanced APIs / Project 2 Lab
WEEK 9	Closures & the module pattern / CRUD & Firebase
WEEK 10	Deploying your app / Final Project Lab

EXIT TICKET QUESTIONS

1. Resources for learning more about asynchronous programming
2. JavaScript is asynchronous, what languages are synchronous?
3. Can I make any function into an async function? Confused.
4. Need more clarification about promises / .then()?
5. Tradeoffs of using callback functions versus Promises? Would you use both?
6. Unsure about what arguments to pass in fetch functions
7. Error Handling: What is try? What is a constructor function?
8. What is the difference between Web Services and API?

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);
```

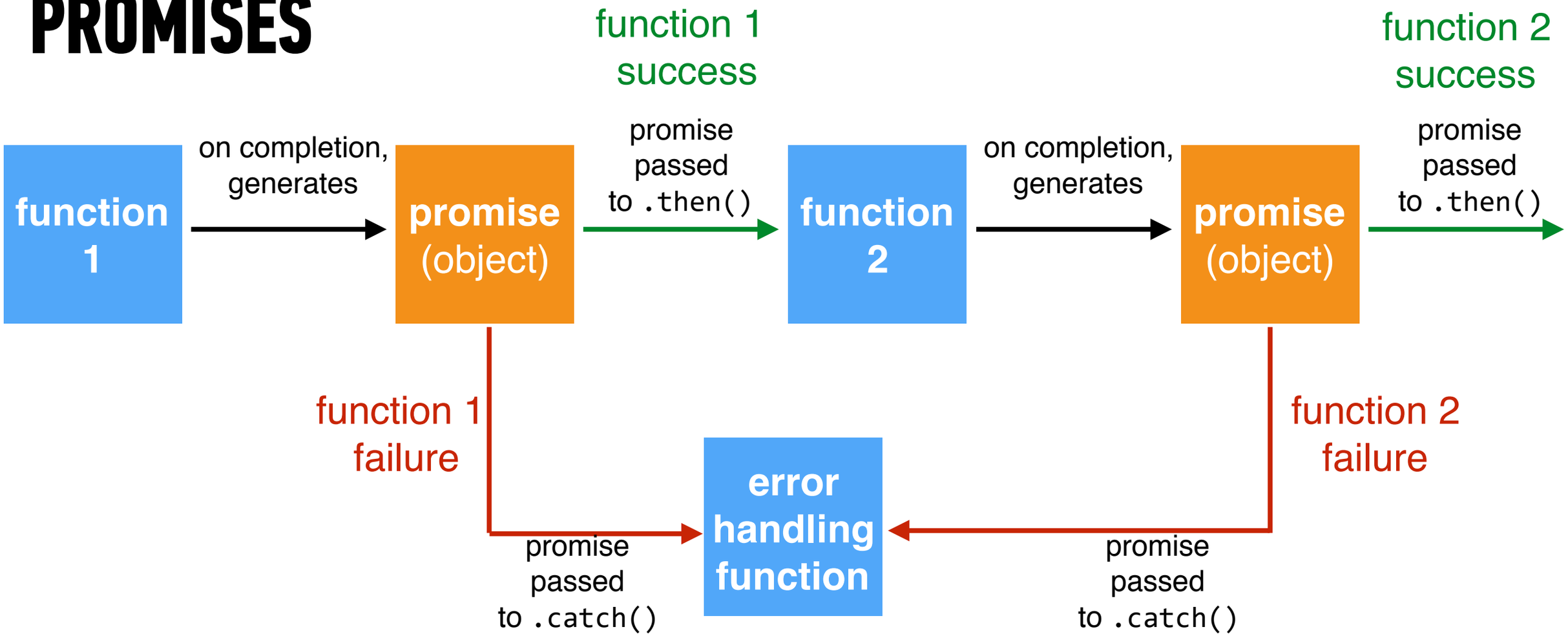
MULTIPLE CALLBACKS WITH PROMISES

```
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

ACTIVITY

TYPE OF EXERCISE

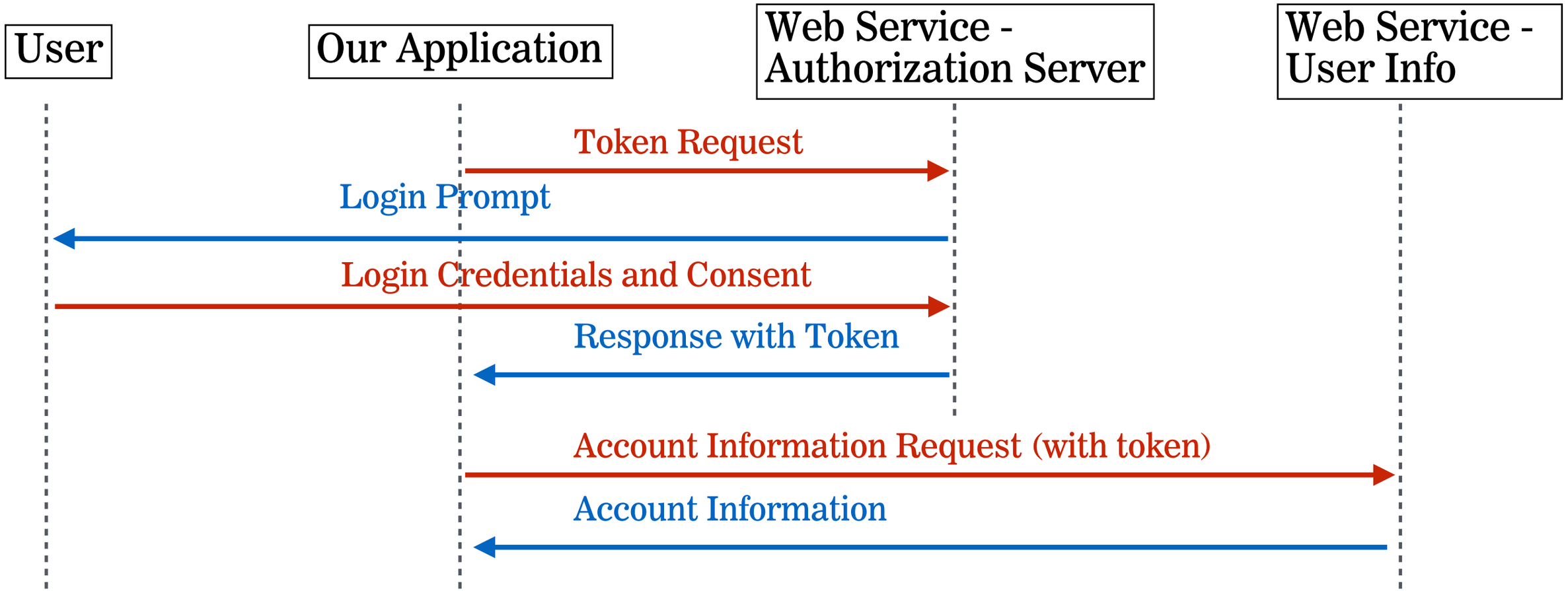
- Turn & Talk

TIMING

2 min

1. Think about a time you've granted an app access to your information from a different web service.
2. Find a partner or two, share your answers, and discuss.

OAuth Flow


User
Our Application
Web Service - Authorization Server
Web Service - User Info
Token Request
Login Prompt
Login Credentials and Consent
Response with Token
Account Information Request (with token)
Account Information

BUILDING OUR APP

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

BUILDING OUR APP

Our app

2 & 3

- Configure initial app view
- Get user's location

4

- Create request containing user's location info

500px.com server

- Parse API response
- Add returned images to view

5

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 from the list at https://github.com/500px/api-documentation#endpoints
2. Identify an endpoint that will let us find photos based on a user's location.

Create Conditional Views

EXERCISE

OBJECTIVE

- Write code to change the view of your single page app after user login

TIMING

5 min

1. Add code to your application within the section that runs after the user logs in.
2. This code should hide the login prompt and instead show the section of the page that will display the results of our API request.
   HINT: This involves DOM manipulation. Think about how you could use the jQuery `hide()` and `show()` methods.

Get User's Location

Call the 500px 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

1. Search the API documentation as necessary to modify your API request to do the following:
   - Sort photos results by highest rated first
   - Return 28 photos instead of the default 20
2. BONUS: Display the current user's information on the site after a successful login. You'll need to look into the Users API or JavaScript SDK documentation to accomplish this.
   If your bonus code isn't working, try 1) disabling extensions, or 2) allowing third-party cookies (In Chrome, Preferences (scroll to bottom) > Advanced > Privacy & Security > Content Settings > Cookies > turn off "Block third party cookies")

Exit Tickets!

(Class #12)

LEARNING OBJECTIVES - REVIEW

- Generate API specific events and request data from a web service.
- Implement the geolocation API to request a location.
- Process a third-party API response and share location data on your website.
- Make a request and ask another program or script to do something.
- 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