High-Level Program Design

JavaScript Development Accelerator

GENERAL ASSEMBLY

Upskill the Broader Organization

JavaScript Development Accelerator

Instructor-Led | Onsite or Remote | ~60 Hours

Overview:

Employees who are already proficient with HTML and CSS will gain an understanding of JavaScript development to build more complex websites. Participants will design and build their own custom front-end applications and practice plugging those applications into third party APIs.

Business Outcomes:

- Expand technical competencies beyond your technology team.
- Improve efficiency by upskilling designers, marketers, and other employees to build rich, interactive websites.
- Enable any employee to create a high-fidelity prototype and collaborate with engineers once ready for production.

~60 Hours of JavaScript Training

JavaScript Fundamentals	The Browser & APIs	Persisting Data	Advanced Topics	Applied Practice
Learn the basics of JavaScript and object-oriented programming.	**Use JavaScript to interact** with the browser, the Document Object Model, and APIs.	**Use advanced programming topics** and persist user data via API calls to a back-end service provider.	**Explore advanced JavaScript frameworks** and app deployment strategies.	**Build a single-page application** that consumes data from at least one API and persists user data via Firebase.

Why JavaScript Accelerator?

- **Validated tools and approaches** for building a single-page application, developed in partnership with top organizations.
- **Built with subject matter experts** with industry experience in software development.
- **60 hours** of expert-led, hands-on learning, including:
  - **Projects, labs, and assignments** that mimic real-world tasks and workflows.
  - **Case studies and examples** that demonstrate how businesses use the concepts.
  - **Regular feedback and touchpoints** with instructors and peers to ensure students meeting learning goals.

Learner Persona

This product is specifically designed for the following audience:

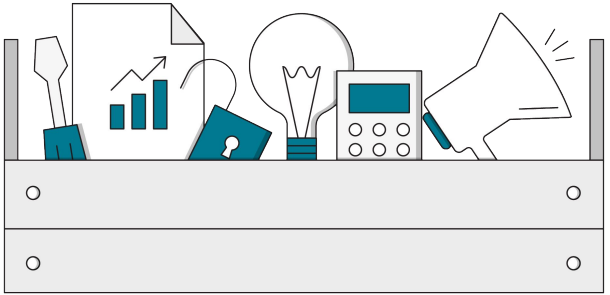
- **Career Accelerator:** Employees who want to build more complex websites and single-page applications — a versatile skill set that complements experiences in design, data, and other tech-adjacent roles.

Anatomy of JavaScript Accelerator

Unit	What's Covered	Project
Pre-Work	Engage in online, self-paced learning to gain an initial understanding of JavaScript and object-oriented programming.	N/A
Unit 1: Fundamentals of JavaScript	Learn the fundamentals of JavaScript and object-oriented programming by working with JavaScript on the command line.	Using the provided scaffolding, build a basic Slack bot that responds to user input and run it locally from your machine in the class Slack channel.
Unit 2: The Browser and APIs	Use JavaScript to interact with the browser, the Document Object Model, and APIs.	Build a simple, single-page application that consumes data from a third-party API like Twitter, Facebook, or Instagram.
Unit 3: Persisting Data and Advanced Topics	Use advanced programming topics and persist user data via API calls to a back-end service provider.	Begin working on your final project: a single-page application that consumes data from at least one API and persists user data via Firebase.
Unit 4: Building and Deploying Your App	Work on your final project and deploy your app using GitHub Pages or Heroku.	Present your capstone piece: a single-page application that consumes data from at least one API and persists user data via Firebase.

Tools Used in the Course

- Google Chrome
- CodePen
- GitHub Enterprise
- Slack and Zoom



JavaScript Accelerator

What You'll Learn

Module Overview

Fundamentals of JavaScript

15 hours 

Why?

Being a developer can be overwhelming at first. There is a multitude of tools and languages to learn. This is what makes JavaScript a fantastic language to start with — it's versatile and can manage a variety of complicated tasks; it's everywhere, and it's easy to learn! In this unit, we'll practice the foundation of Javascript to help you hit the ground running.

Learning Objectives

- Explore how the web works and the client server model.
- Compare JavaScript in the browser versus the command line.
- Run JavaScript code on the command line using Node.js.
- Gain an introduction to working with variables and conditionals.
- Understand data types.
- Work with collections and loops, and iterate over collections.
- Write functions.
- Discover the concept of scope.

Datatypes

For this lesson, we're going to use the terminal and Node to run some basic scripts to understand the types of data we're working with. Open the terminal and type in `node`.

Part 1: `typeof( )`

We don't yet know what type of data we're working with, so let's ask the computer. To do this, we can use `typeof()`. Let's try it out in the terminal with the following:

```
typeof(37) === 'number';
=> true

typeof({}) === 'object';
=> true

typeof('hi there') === 'string';
=> true
```

`typeof()` returns a string with the type of the operand, or expression of the object you're looking at.

> Note: At this point we haven't explained exactly what objects are. Provide a brief overview of objects as a collection of properties, and of a property as an association between a key and a value. Objects in JavaScript are used in two ways: >1. As simple structured data store, similar to arrays—the main difference being that instead of accessing values by index, we access them by a key. >2. As a fundamental programming paradigm that helps us structure and categorize our code. More about objects in the second half of this class.

Interacting with Your Hubot

To create instructions for your Hubot, you need to add a JavaScript file to the `scripts` folder. You can add multiple scripts files to that folder and all will execute when the Hubot is run. Let's look at a few commands that will help us build our Hubot scripts.

Listening

.hear

The `.hear` command listens for a specific phrase anywhere in the Slack room. You don't have to mention your Hubot in order to get a response.

In the example below, when the bot hears "Hello!", it will respond, "Hi there!":

```
bot.hear(/Hello!/, function(res) {
  return res.send("Hi there!");
});
```

```
tim: Hello!
hubot: Hi there!
```

Module Overview

The Browser and APIs

15 hours 

Why?

Short for “JavaScript Object Notation,” JSON is a way to store information in an organized and accessible way. In this unit, we’ll dive into how JavaScript shares data and interacts with the browser, web servers, and applications.

Learning Objectives

- Get acquainted with objects and methods.
- Work with JSON-formatted data.
- Explore the jQuery library and its capabilities.
- Understand the Document Object Model (DOM) and manipulate objects in the DOM.
- Handle forms and user input.
- Use events and listeners.
- Gain an introduction to AJAX.
- Make API calls, consuming and incorporating API data.
- Compare and contrast asynchronous and synchronous JavaScript.
- Leverage callbacks.

Setting Object Properties

```
// We can set object properties via the key in dot notation (more common for simple scenarios)
myHouse.windows = 6;
myHouse.address = "Tedi Manor, Gotham City";

// We can also set object properties via square brackets with the key as a string.
// We use the square bracket notation when a property name has either a special character
// like a space or a hyphen, or when the property name starts with a number.
// This notation is also used when our property names are dynamically determined
myCar["num-of-wheels"] = 4;
myCar["doors"] = 2;
```

We access object properties the same way as setting them:

```
myHouse.windows; //returns 6
myHouse.address; // returns "Tedi Manor, Gotham City";

myCar["num-of-wheels"]; // returns 4;

var numDoors = "doors";
myCar[numDoors]; // returns 2;
```


Event Delegation and Best Practices: Codealong

We started covering mouse events with the click event. We can add additional mouse events in the same manner.

```
var $thingListItems = $('#fav-list li');

$thingListItems.on('mouseenter', function(e) {
  $(this).removeClass('inactive');
  $(this).siblings().addClass('inactive');
});

$thingListItems.on('mouseleave', function(e) {
  $(this).siblings().removeClass('inactive');
});
```

The above code listens for two events:

1. User's mouse set to enter the list item element. In this case, it removes the 'inactive' class from itself (if it exists) and adds it to its sibling list items. *Note: element and sibling class switching is a common best practice to toggle visual effects on user actions.*
2. User's mouse set to leave the list item element. This removes the 'inactive' class from all elements on the same level.

While the above code works great for existing elements, if we add new elements to the DOM, the events will not fire up for the newly added elements.

Module Overview

15 hours 

Persisting Data and Advanced Topics

Why?

In this comprehensive and hands-on unit, we'll explore advanced API topics as well as CRUD, short for "create, read, update, and delete" and Firebase, Google's mobile platform. These tools will enable you to develop and continuously improve your application.

Learning Objectives

- Dive into authentication, working with tokens and API keys.
- Utilize OAuth.
- Get acquainted with prototypal inheritance, prototypes, and constructors.
- Explore the concept of "this."
- Handle anonymous functions.
- Understand CRUD.
- Gain an introduction to Firebase and write user data to it.
- Retrieve and update user data.

Class vs. Prototype

The purpose of prototypal inheritance is to offer a modeling object, which contains certain behaviors, to other objects to be prototyped off of, inheriting the model's behaviors. For those of you familiar with other object-oriented languages, such as Ruby, this functionality may sound very similar to what a class does. However, Javascript is not a class-based language, but instead a prototype-based language. Let's discuss the differences by comparing Ruby against Javascript.

Class:

In Ruby, classes are objects that have unique responsibilities and methods:

- manufacture new objects
- define the behavior of the objects they manufacture

Prototypal Inheritance:

Similar to Ruby, Javascript also uses objects, function objects (generally everything in Javascript is an object except primitives), to perform inheritance. However unlike Ruby, Javascript does not have a special function object that can both manufacture a new object as well as define the behavior of the object it creates. Instead, Javascript has two different features, constructors and prototypes, which accomplish these tasks:

- a *constructor* function manufactures new objects
- a *prototype* property defines the behavior of new objects manufactured by the constructor

Now that we know the key differences between a prototype-based and a class-based language, let's take a deeper dive into the mechanics of a constructor and a prototype.

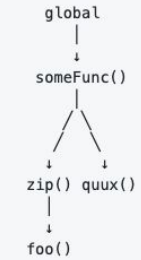
Scope Tree

Looking at the nesting from top-down, a tree of scopes is formed.

This code

```
function someFunc() {
  function zip() {
    function foo() {
    }
  }
  function quux() {
  }
}
```

Produces this tree



Remembering that inner scopes can access outer scope's variables, but *not* vice-versa (`foo()` can access `zip()`'s variables, and `zip()` can access `someFunc()`'s variables), then it makes more sense to look at the tree from bottom-up, which forms a chain, also known as...

Module Overview

Building and Deploying Your App

15 hours 

Why?

It's showtime! Well, almost. In this unit, we'll focus on refining your application with advanced JavaScript frameworks and explore strategies and tools for deploying your app. You'll get the support and feedback you need to successfully deploy a high-quality app.

Learning Objectives

- Get started with advanced JavaScript frameworks.
- Explore app deployment strategies.
- Deploy your app to GitHub Pages or Heroku and use a custom domain name.
- Use Firebase with Heroku.

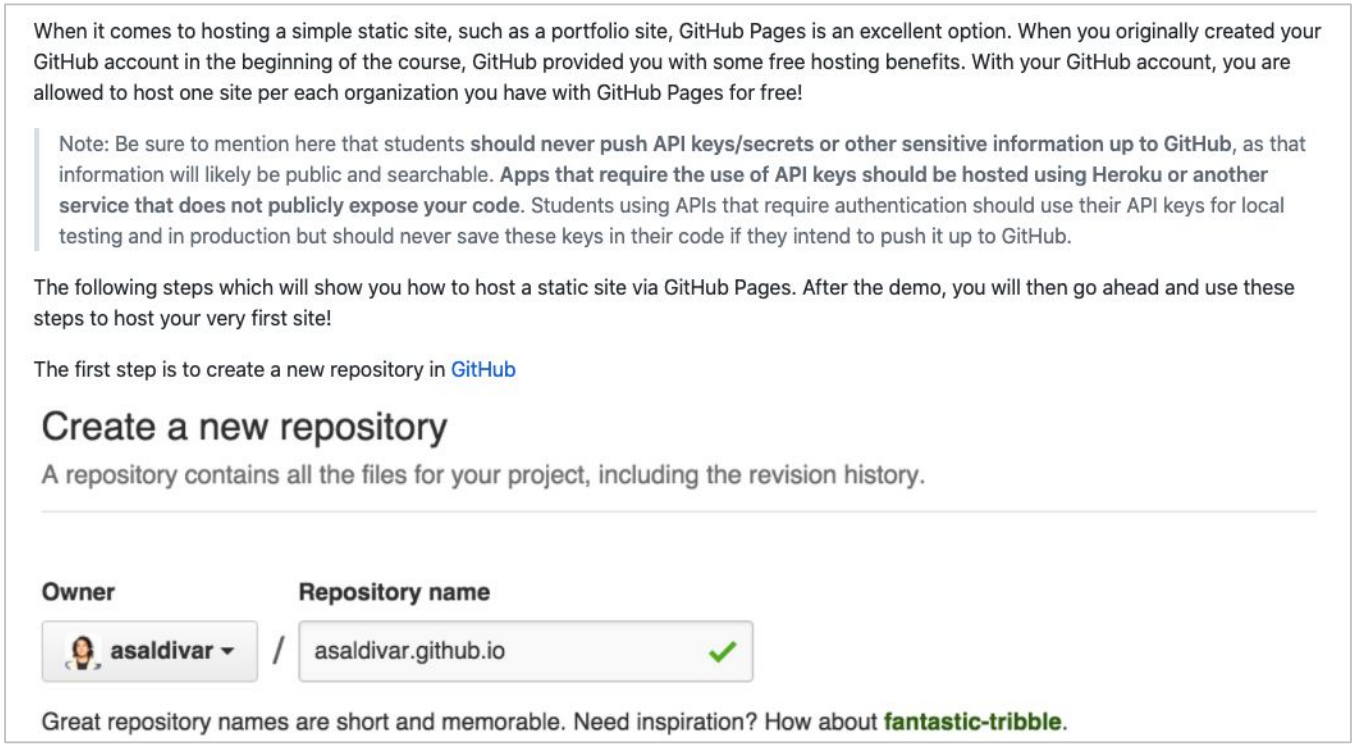
Deploy with GitHub Pages - Demo

When it comes to hosting a simple static site, such as a portfolio site, GitHub Pages is an excellent option. When you originally created your GitHub account in the beginning of the course, GitHub provided you with some free hosting benefits. With your GitHub account, you are allowed to host one site per each organization you have with GitHub Pages for free!

> Note: Be sure to mention here that students **should never push API keys/secrets or other sensitive information up to GitHub**, as that information will likely be public and searchable. **Apps that require the use of API keys should be hosted using Heroku or another service that does not publicly expose your code**. Students using APIs that require authentication should use their API keys for local testing and in production but should never save these keys in their code if they intend to push it up to GitHub.

The following steps which will show you how to host a static site via GitHub Pages. After the demo, you will then go ahead and use these steps to host your very first site!

The first step is to create a new repository in GitHub



Deploy with Heroku

Heroku is one of the most widely used hosting services for web applications in the world and provides many useful features. It is not just a hosting service but an entire cloud platform service. What this means is that instead of just hosting your code it provides features such as:

- Instant Deployment with Git push - build of your application is performed by Heroku using your build scripts.
- Plenty of Add-on resources (applications, databases etc).
- Full Logging and Visibility - easy access to all logging output from every component of your app and each process (dyno).
- Environment variables, which allow you to store sensitive information (like API keys) separately from your code.

To summarize the benefits of Heroku in one line, Heroku gives you an environment where you just push code and some basic configuration and as a result get a running application.

> Note: This is a great time to remind students that they **should never push API keys/secrets or other sensitive information up to GitHub!** Students should be able to push their API credentials up to Heroku without incident, however, this is not conisidered a best practice. Consider showing students how to use environment variables (known as "config vars" in Heroku parlance) using the Heroku dashboard as part of the below lesson on Heroku.

With that all said, let's go to Heroku and sign up!

Now that you have signed up, let's talk about how we can utilize Heroku to host our applications. When thinking of Heroku, it is best to view it in the same way as you do GitHub. It is simply a *remote*, a.k.a. a cloud, you push your code up to for hosting purposes.

For example, if you run the command `git remote -v` you will be returned with the remote that your application is connected to:

```
origin  https://github.com/generalassembly-studio/JS_Materials.git (fetch)
origin  https://github.com/generalassembly-studio/JS_Materials.git (push)
```


GA