



# Phase 1 Project Guidelines

 [\\_ \(https://github.com/learn-co-curriculum/phase-1-javascript-project-mode\)](https://github.com/learn-co-curriculum/phase-1-javascript-project-mode)   
[\\_ \(https://github.com/learn-co-curriculum/phase-1-javascript-project-mode/issues/new\)](https://github.com/learn-co-curriculum/phase-1-javascript-project-mode/issues/new)

## Learning Goals

- Design and architect features across a frontend
- Communicate and collaborate in a technical environment
- Integrate JavaScript and an external API
- Debug issues in small- to medium-sized projects
- Build and iterate on a project MVP



## Introduction

Welcome to JavaScript Project Mode!

You've worked so hard to get here and have learned a ton. Now it's time to bring it all together!

For this project, you're going to build a Single Page Application (**SPA**). Building this application will be challenging because it will integrate everything you've learned up to this point. Your frontend will be built with HTML, CSS, and JavaScript and will communicate with a public API.

## Project Requirements

1. Your app must be a HTML/CSS/JS frontend that accesses data from a public API or from a db.json file using json-server. Your API or db.json should return a collection of at least 5 objects with each object having at least 3 attributes. All interactions between the client and the API should be handled asynchronously and use JSON as the communication format. Try to avoid using an API that requires a key. APIs that are free and require no authorization will be easiest to use. For ideas, see this [list of no-auth APIs](https://mixedanalytics.com/blog/list-actually-free-open-no-auth-needed-apis/)  [\\_ \(https://mixedanalytics.com/blog/list-actually-free-open-no-auth-needed-apis/\)](https://mixedanalytics.com/blog/list-actually-free-open-no-auth-needed-apis/). If you would like to use an API that requires a key, please consult with your instructor on how to protect that key. **NEVER push your API key to github!**
2. Your entire app must run on a single page. There should be NO redirects or reloads. In other words, your project will contain a single HTML file.
3. Use at least 3 distinct [event listeners](https://developer.mozilla.org/en-US/docs/Web/Events)  [\\_ \(https://developer.mozilla.org/en-US/docs/Web/Events\)](https://developer.mozilla.org/en-US/docs/Web/Events) (3 events of different types) that enable interactivity. What this means is that, if you had 3 click events, that would only count as 1 distinct event and you would need to add at least 2 more. Think search or filter functionality, toggling dark/light mode, upvoting posts, etc. Each of your event listeners should also have its own unique callback function. These must be added using

JavaScript's `.addEventListener()` method. Events embedded into HTML elements and CSS will not count toward the total. Please ask your instructor if you have questions regarding this requirement.

4. Your project must implement at least one instance of array iteration using available array methods ( `map` , `forEach` , `filter` , etc). Manipulating your API data in some way should present an opportunity to implement your array iteration.
5. Follow good coding practices. Keep your code DRY (Do not repeat yourself) by utilizing functions to abstract repetitive code.

## Stretch Goals

1. Use `json-server`  (<https://www.npmjs.com/package/json-server>) in your project to persist your app's interactivity.

## Strategy, Timeline, and Tips

### Planning

- Plan out your features
- Develop user stories
  - “As [ a user ], I want [ to perform this action ] so that [ I can accomplish this goal ].”
  - Features should not need you there to explain them to users
- Plan out the structure of your JSON requests

### Project Pitches

Before you start working on your project, you'll pitch your project idea to your instructors for approval and feedback.

For your project pitch, you should include:

- The basic story of your application
- The core features of your MVP
- The API data you'll be using and how you'll use it
- Challenges you expect to face
- How you are meeting the requirements of the project

Feel free to send this pitch to your instructor via slack asynchronously.

### MVP ASAP

- Build a Minimum Viable Product (MVP) as quickly as possible.
  - Pick an API and explore it early on to ensure it will work for your need

## Instructor Guidance

You should strive to solve problems independently, but you also shouldn't waste your time stuck on a problem. A good guideline for a small bug is the rule of 10s:

- 10 minutes debugging the code
- 10 minutes using Google and StackOverflow to try to find an answer
- 10 minutes asking your fellow students for help
- Asking an instructor

If you seek out instructor guidance on your design from the start, they might help steer you into design and architectural decisions that will help you down the road. That will also give the instructors context for what your app is supposed to do, so you won't need to explain everything to them when asking for help debugging.

## Guidelines for Staying Organized


**Write down** the decisions you make about your project. This will not only help you think more clearly, it will also help you communicate your project to instructors when asking for help. In addition to writing everything down, we also recommend the following to help stay organized and on track:

- Describe/sketch your ideas (use diagrams!).
- Start by creating a frontend directory with the basic files you'll need
- Next, build enough code to get some API data to work with. Don't worry about building all of your async code yet, just get to the point where you can access one endpoint on an API, then start working on getting that data displayed.
- Then, continue to build additional async code and frontend features.
- Continue building features one by one.

Check in with your instructors to make sure your scope and timeline are manageable.

## JSON Server Instructions

**Note:** Using `json-server` is a stretch goal, so make sure you have a working MVP before trying to set up `json-server` !

You can use this [json-server template](https://github.com/learn-co-curriculum/json-server-template)  (<https://github.com/learn-co-curriculum/json-server-template>) to generate your backend code. Using this template will make it easier to deploy your backend later on.

If you prefer, instead of using the template, you can create a `db.json` file with a structure in the root of your project that looks like this:

```
{
  "toys": [
    {
      "id": 1,
      "name": "Woody",
      "image": "http://www.pngmart.com/files/3/Toy-Story-Woody-PNG-Photos.png",
      "likes": 8
    },
    {
      "id": 2,
      "name": "Buzz Lightyear",
      "image": "http://www.pngmart.com/files/6/Buzz-Lightyear-PNG-Transparent-Pict",
      "likes": 14
    }
  ]
}
```

Then, assuming you have `json-server` installed globally, you can run this command to run the server:

```
$ json-server --watch db.json
```

Whatever top-level keys exist in your `db.json` file will determine the routes available. In the example above, since we have a key of `toys` pointing to an array of toy objects, `json-server` will generate the following routes:

- `GET /toys`
- `POST /toys`
- `GET /toys/:id`
- `PATCH /toys/:id`
- `DELETE /toys/:id`

You can consult the [json-server docs](https://www.npmjs.com/package/json-server)  (<https://www.npmjs.com/package/json-server>) for more information.

## Resources

- [Public APIs](https://github.com/public-apis/public-apis)  (<https://github.com/public-apis/public-apis>)
- [Fun APIs](https://apilist.fun/)  (<https://apilist.fun/>)
- [json-server](https://www.npmjs.com/package/json-server)  (<https://www.npmjs.com/package/json-server>)