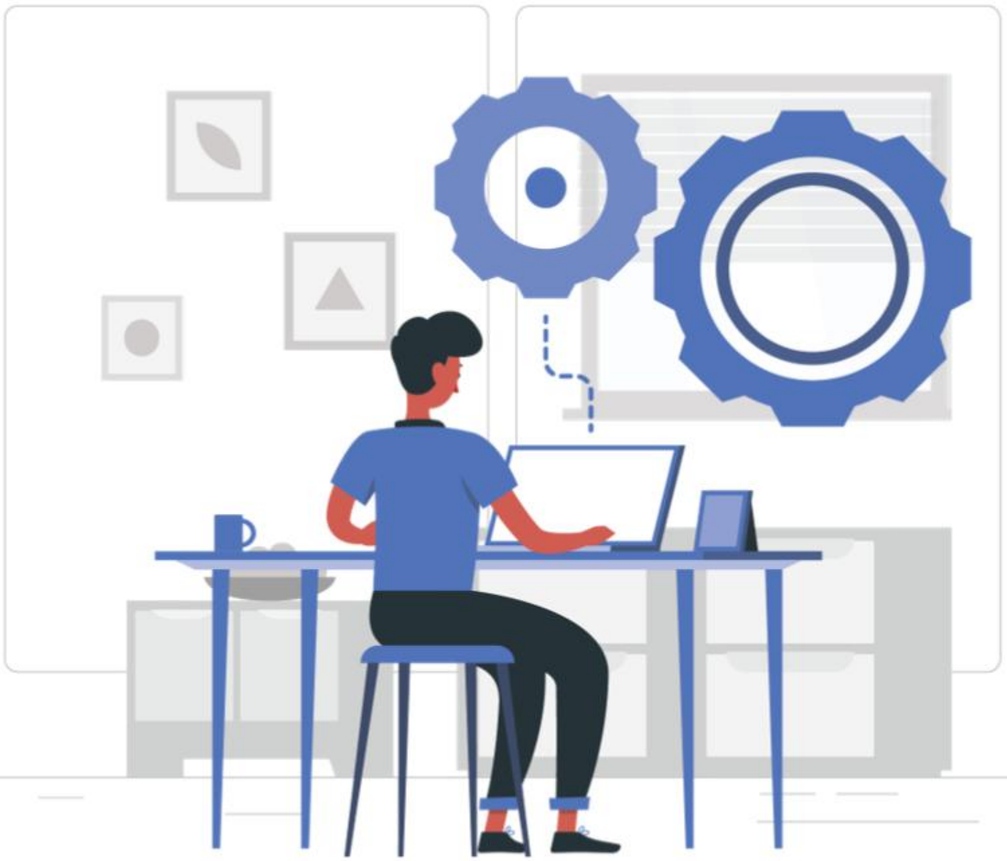

Practice

Model Real World Data Using Objects

Practices

- Practice 1: Create simple JSON
- Practice 2: Create JSON to represent a collection
- Practice 3: Create complex JSON
- Practice 4: Parse JSON to perform aggregations
- Practice 5: Convert JavaScript object to JSON

PRACTICE

Practice 1: Create Simple JSON

The details of flights from Chicago to New York are provided in a tabular format in the upcoming slides.

Create simple JSON data with the flight details provided and save it in a `.json` file.

Tasks

- The solution for this practice should be written in the **.json** files located inside the folder **p1-simple-json** of the boilerplate.
- Create separate JSON for each of the rows with the flight details provided below and save them in separate .json files (p1-submission-1.json, p2-submission-2.json, p3-submission-3.json):

from	to	departure-days	flight no	airlines	departure-time	arrival-time	stop overs	isDirect	departure-from	arrival-at
Chicago	New York	M-Tu-W-Th-F	DL551	Delta Airlines	06:00:00	09:13:00		TRUE	ORD	LGA
Chicago	New York	M-T-W-Th	UA2202	United Airlines	19:46:00	22:56:00		TRUE	ORD	EWR
Chicago	New York	Th	AA1378	American Airlines	11:21:00	14:30:00		TRUE	ORD	JFK

Note:

- departure-from and arrival-at are the airport codes.
- For the cells left blank, the value in the JSON should be `null`.
- In a table, a row represents an object.
- In this case, the row represents a flight.
- Flight is an example of an object in the real world that provides the details of a particular flight from Chicago to New York on specified departure days.

PRACTICE

Practice 2: Create JSON to Represent a Collection

The details of multiple flights from Chicago to New York are provided in a tabular format in the upcoming slide.

Create a single JSON that represents collections of flights and save it in a .json file.

Tasks

- The solution for this practice should be written in the file **p2-submission.json** located inside the folder **p2-json-for-collection** of the boilerplate.
- The details of multiple flights flying from Chicago to New York are provided in the tabular format below:

from	to	departure-days	flight no	airlines	departure-time	arrival-time	stop overs	isDirect	departure-from	arrival-at
Chicago	New York	M-Tu-W-Th-F	DL551	Delta Airlines	06:00:00	09:13:00		TRUE	ORD	LGA
Chicago	New York	M-T-W-Th	UA2202	United Airlines	19:46:00	22:56:00		TRUE	ORD	EWR
Chicago	New York	T-W-T	UA265	United Airlines	18:00:00	21:15:00		TRUE	ORD	LGA
Chicago	New York	Th	AA1378	American Airlines	11:21:00	14:30:00		TRUE	ORD	JFK
Chicago	New York	M-Tu-W-Th-F	DL556	Delta Airlines	08:45:00	11:59:00		TRUE	ORD	LGA
Chicago	New York	Tu	AA552	American Airlines	18:23:00	21:32:00		TRUE	ORD	EWR

- Instead of creating a separate JSON for each of these rows, create a single JSON that represents this table.

Note: The table is a collection of rows, with each row providing details of a particular flight option.

PRACTICE

Practice 3: Create Complex JSON

The details of flights from Chicago to New York are provided in a tabular format in the upcoming slide.

Create JSON with the details provided and save it in a `.json` file. Use JSON objects to model flight details.

Tasks

- The solution for this practice should be written in the file **p3-submission.json** located inside the folder **p3-json-with-objects** of the boilerplate.
- The details of flights from Chicago to New York are provided in the tabular format below:

from	to	departure-days	flight no	airlines	departure-time	arrival-time	stop overs	isDirect	departure-from	arrival-at
Chicago	New York	M-Tu-W-Th-F	DL551	Delta Airlines	06:00:00	09:13:00		TRUE	ORD	LGA
		M-T-W-Th	UA2202	United Airlines	19:46:00	22:56:00		TRUE	ORD	EWR
		T-W-T	UA265	United Airlines	18:00:00	21:15:00		TRUE	ORD	LGA
		Th	AA1378	American Airlines	11:21:00	14:30:00		TRUE	ORD	JFK
		M-Tu-W-Th-F	DL556	Delta Airlines	08:45:00	11:59:00		TRUE	ORD	LGA
		Tu	AA552	American Airlines	18:23:00	21:32:00		TRUE	ORD	EWR

Tasks (Cont'd)

Note:

- The value for fields `from` and `to` are same for all flight options and need not be repeated.
- The list of flight options should be represented as a collection within the JSON along with these fields.
  - ❖ Each item in the collection is an object that represents a flight option.
- Create a JSON with the details provided to represent the flight object.

PRACTICE

Practice 4: Parse JSON to Perform Aggregations

Write JavaScript program to convert the JSON containing flight details into a JavaScript object.

Also, perform a filter operation on the JavaScript object to retrieve details of United Airlines flights from Chicago to New York.

Tasks

- The solution for this practice should be written in the file **p4-submission.js** located inside the folder **p4-js-to-json** of the boilerplate.
- Steps to convert JSON to JavaScript object:
  - Step 1: Declare a JavaScript string variable and assign it the JSON created in Practice 3.
  - Step 2: Using the `JSON.parse()` function convert the JSON string to JavaScript object.
    - The JavaScript object should be an array of objects with flight details.
- Steps to perform filter operation:
  - Step 1: Define a function `searchFlights()` that accepts an array of flights, from (city name), to (city name) and (name of the) airlines as parameters.
  - Step 2: Inside function, perform a filter operation on the array using the array method `filter()` to fetch flights flying from Chicago to New York.
  - Step 3: Chain the output of `filter()` method and invoke `map()` array method to transform the flight details.

Tasks (Cont'd)

- Step 4: The transformed result should be an array of flights from Chicago to New York where:
  - Each array item should be an object with properties `from`, `to` and `flightOptions`.
  - The flightOptions array should be filtered using the array method `filter()` to retrieve only the flights of United Airlines.

PRACTICE

Practice 5: Convert JavaScript Object to JSON

The JavaScript array given in the upcoming slide consists of a list of airports.

Convert the object to JSON and save it in a .json file.

Tasks

- Write the code as snippet in the sources panel of Chrome's developer tools.
- Steps to perform this task:
  - Step 1: Declare JavaScript object variable and assign the data given below:

```
[
    {
        "ORD": {
            "airportName": "Chicago O'Hare International Airport",
            "state": "Chicago",
            "facilities":["taxi parking","family lounge", "on-site hotel", "lost and found"]
        },
        "LWA": {
            "airportName": "LaGuardia Airport",
            "state": "New York",
            "facilities": ["terminal transits", "business lounge", "private car parking", "taxi
parking"]
        }
    }
]
```

Tasks (Cont'd)

- Step 2: Using the `JSON.stringify()` method, convert the JavaScript object to JSON.
- Step 3: Press Ctrl+Enter to run the code and test the output.
- Step 4: Once the desired output is achieved, copy and paste the solution for this exercise in **p5-submission.js** file.
- Step 5: The solution for this practice should be written in the file **p5-submission.js** located inside folder **p5-js-to-json** of the boilerplate.
- Step 6: Copy and paste the JSON data in the **p5-submission.json** file.