14.2 Lesson Plan - Objects, ES6, and Tables

Overview

Today's class will take a deeper look into JavaScript through Objects, ES6, and tables.

Class Objectives

- Students will understand and be able to use **`forEach`** and callback functions and how to use them.
- Students will create, update, and iterate JavaScript Objects.
- Students will understand how to apply **`map`** and **`filter`** to parse data.
- Students will be able to create and use arrow functions to simplify code.
- Students will understand the basic structure of a Bootstrap HTML table.

Instructor Notes

- Much of today's class will focus on delving further into JavaScript, enabling students to perform more sophisticated operations and data manipulation.

- Students may be scrambling to keep up with the new material, but remind them that they will have the next few weeks to practice these fundamentals. Today's lesson is really about exposure to modern JavaScript programming techniques and best practices.

- Concepts and syntax from ES6 will be introduced today, along with objects and functional programming. Today's topics cover most of the syntax that they will encounter in modern data applications written in JavaScript.

Sample Class Video

- View an example class lecture visit *(Note video may not reflect latest lesson plan)*
    - Class Video Pt. 2
    - Class Video Pt. 3

1. Instructor Do: Welcome Class

- Welcome the class and explain that today, we will cover objects, functional programming, and data munging in JavaScript.

- Explain that the techniques covered today are very common ways to work with data using modern JavaScript syntax and best practices.

5. Instructor Do: JavaScript Objects (0:10)

- Explain to students that JavaScript objects are similar to Python dictionaries. Assure students that their ample experience in using Python dictionaries, as well as parsing and dealing with JSON data (JavaScript Object Notation), will serve them well today.

- Take a moment to **summarize** some things we know about Python dictionaries:

    - They **organize** information in `key` and `value` pairings.
    - Unlike lists, key-value pairs are **unordered**.
    - The `key` is used to access the `value`.

- Open `index.html` in a browser and `index.js` with a text editor.

- Explain that JavaScript objects look very similar to Python dictionaries.

    - Each `key` can hold as its `value` of any one of data types, including a string, a boolean, or an array. It can even hold an object.

```
var movie = {
    name: "Star Wars",
    year: 1977,
    profitable: true,
    sequels: [5, 6, 1, 2, 3, "The Last Jedi"]
};
```

- To retrieve a value from the object, both **dot** notation and **bracket** notation can be used for the key. However, the **dot** notation is preferred.

```
console.log(movie.name);
console.log(movie.year);
console.log(movie.sequels[0]);
```

```
console.log(movie["name"]);
```

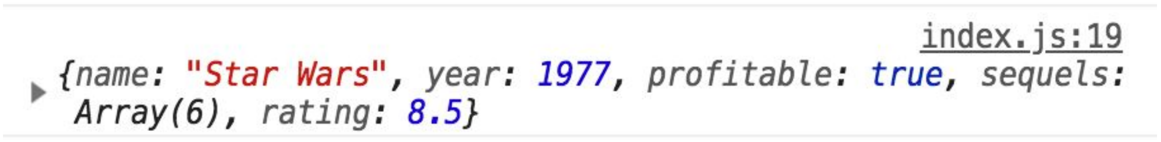
- Show that, much like in Python, a property can be added to a JavaScript object simply by specifying a key and assigning a value to it:

```
movie.rating = 8.5;
console.log(movie);
```

- Show that key-value pairs can also be deleted:

```
delete movie.sequels;
console.log(movie);
```

- Show the updated object in the browser:



```
                                                    index.js:19
▶ {name: "Star Wars", year: 1977, profitable: true, sequels:
   Array(6), rating: 8.5}
```

- Also like Python, it's possible to test whether a `key`, or a `property`, exists in an `object`:
    - If the `movie` object has a `property` named “rating,” the console prints the statement.

```
if ('rating' in movie) {
    console.log("This movie has a rating!");
}
```

- Finally, demonstrate how to loop through an object:

```
for (var prop in movie) {
    console.log(movie[prop]);
}
```

  - This code loops through the key-value pairs of the movie object.
  - The variable prop represents the object's key in each iteration.
  - During each iteration in this for-loop, the value of the key-value pair is printed to the console with movie[prop] .

- Explain that JavaScript has several more built-in methods to manipulate objects. Here, the object is a cartoon family:

```
var people = {
mom: "wilma flintstone",
dad: "fred flintstone",
daughter: "pebbles",
son: "bambam"
};
console.log(people);
```

- Explain that `Object.keys()` displays all the keys of an object:

```
console.log(Object.keys(people));
```

  - The method takes the name of the object as its argument.

- Similarly, show that `Object.values()` displays all the values of an object:

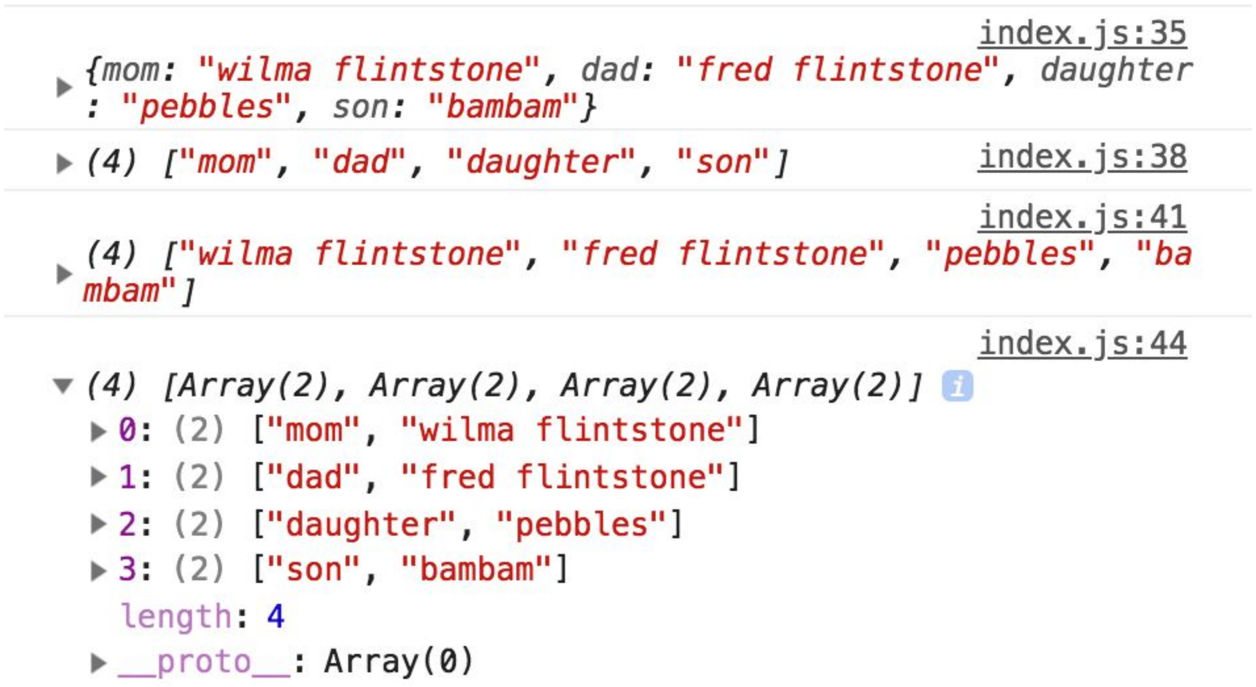
```
console.log(Object.values(people));
```

- Finally, show that to access both keys and values, `Object.entries()` can be used:

```
console.log(Object.entries(people));
```

- Each key-value pair is returned inside an array.

- Show the results in the console while briefly reiterating the above methods:



```
                                                          index.js:35
▶ {mom: "wilma flintstone", dad: "fred flintstone", daughter
  : "pebbles", son: "bambam"}
▶ (4) ["mom", "dad", "daughter", "son"]                   index.js:38
                                                          index.js:41
▶ (4) ["wilma flintstone", "fred flintstone", "pebbles", "ba
  mbam"]
                                                          index.js:44
▼ (4) [Array(2), Array(2), Array(2), Array(2)] i
  ▶ 0: (2) ["mom", "wilma flintstone"]
  ▶ 1: (2) ["dad", "fred flintstone"]
  ▶ 2: (2) ["daughter", "pebbles"]
  ▶ 3: (2) ["son", "bambam"]
    length: 4
  ▶ __proto__: Array(0)
```

End of Section