Javascript OOP

START PROJECT

Object-Oriented JavaScript Basics

This tutorial is designed to help those with basic JavaScript knowledge to jump into Object-Oriented Programming.

What are you going to learn?

- What is Object-Oriented Programming
- What is a class and how it relates to an object
- What is static typing
- How to approach a familiar problem in an object-oriented way

Not started

Ratings (13)

DIFFICULTY:	3.2
USEFULNESS:	★★★★⯪ 4.2
FUN:	★★★⯪☆ 3.4
MATERIALS:	★★★★⯪ 4.3
DURATION:	152 min

Tasks

SHOW ALL CRITERIA

Construct a programming language

Define a function called `constructProgrammingLanguage()`.

1. There is a function called `constructProgrammingLanguage()` in the `functional.js` file.
2. The function returns a new object with the following keys: `name`, `isStaticallyTyped`, and `supportsOOP`.
3. The function receives parameters for each key of the constructed object (so `name`, `isStaticallyTyped`, and `supportsOOP`).
4. If any of the keys are not provided as a function parameter or are not the desired type, the function throws a new error (`name` is a string, `isStaticallyTyped` is a boolean, and `supportsOOP` is a boolean).

CRITERIA

Construct a code editor

Define a function called `constructCodeEditor()`.

1. There is a function called `constructCodeEditor()` in the `functional.js` file.
2. The function returns a new object with the following keys: `name` and `supportedLanguages`.
3. The `supportedLanguages` key has a default value that is an empty array.
4. The function receives a parameter with the `name` of the constructed object.
5. If the `name` is not provided as a function parameter or is not a string, the function throws a new error.

CRITERIA

Add a programming language to an editor

Define a function called `addProgrammingLanguageToEditor()`.

1. There is a function called `addProgrammingLanguageToEditor()` in the `functional.js` file.

2. The function receives two parameters, one is a programming language object, and the other is a code editor.
3. Both parameters must be objects (throws an error otherwise).
4. The code editor object must have a key called `supportedLanguages`.
5. The function adds the received programming language to the `supportedLanguages` key/array of the code editor object.

CRITERIA

Remove a programming language from an editor

Define a function called `removeProgrammingLanguageFromEditor()`.

1. There is a function called `removeProgrammingLanguageFromEditor()` in the `functional.js` file.
2. The function receives two parameters, one is a programming language object, and the other is a code editor.
3. Both parameters must be objects (throws an error otherwise).
4. The code editor object must have a key called `supportedLanguages`.
5. The function removes the received programming language from the `supportedLanguages` key/array of the code editor object.

CRITERIA

Programming language class

Define a class called `ProgrammingLanguage`.

1. There is a class called `ProgrammingLanguage` in the `oop.js` file.
2. The class has a `name`, an `isStaticallyTyped`, and a `supportsOOP` property.
3. The class has a constructor method that receives parameters for each property (so `name`, `isStaticallyTyped`, and `supportsOOP`).
4. If any of the properties are not provided as a constructor parameter or are not the desired type, the method throws a new error (`name` is a string, `isStaticallyTyped` is a boolean, and `supportsOOP` is a boolean).

CRITERIA

Code editor class

Define a class called `CodeEditor`.

1. There is a class called `CodeEditor` in the `oop.js` file.
2. The class has two properties called `name` and `supportedLanguages`.
3. The class has a constructor method that receives a parameter for the `name` property.
4. If the `name` is not provided as a constructor parameter or is not a string, the method throws a new error.
5. The `supportedLanguages` property has an empty array as its default value.

CRITERIA

Add programming language method

Define a method called `addProgrammingLanguage()` in the `CodeEditor` class.

1. There is a method called `addProgrammingLanguage()` in the `CodeEditor` class.
2. The method receives one parameter which is a programming language object.
3. The passed programming language must be an instance of the `ProgrammingLanguage` class (throws an error otherwise).

4. The method adds the received programming language to the `supportedLanguages` property/array of the class.

CRITERIA

Remove programming language method

Define a method called `removeProgrammingLanguage()` in the `CodeEditor` class.

1. There is a method called `removeProgrammingLanguage()` in the `CodeEditor` class.
2. The method receives one parameter which is a programming language object.
3. The passed programming language must be an instance of the `ProgrammingLanguage` class (throws an error otherwise).
4. The method removes the received programming language from the `supportedLanguages` property/array of the class.

CRITERIA

OPTIONAL TASK: Hide supported languages

Hide the `supportedLanguages` property of the `CodeEditor` class.

1. The `supportedLanguages` property of the `CodeEditor` class is not available from outside of the object.

CRITERIA

Hints

- For testing purposes, here are a couple of examples that you can use:
  - Programming languages:
    - JavaScript (isStaticallyTyped: false, supportsOOP: true)
    - Java (isStaticallyTyped: true, supportsOOP: true)
    - C# (isStaticallyTyped: true, supportsOOP: true)
  - Code editors:
    - Visual Studio Code (supports Java, C#, and JavaScript)
    - JetBrains IntelliJ IDEA (supports Java and JavaScript)
    - JetBrains Rider (supports C# and JavaScript)

Background materials

What is Object-Oriented Programming? *(please don't be thrown off by the "Coding for Kids" phrase in the video's title and trust that it is a great introduction to what OOP is)*

Statically Typed vs Dynamically Typed Languages

JavaScript Functions

JavaScript Objects

JavaScript Errors

JavaScript Classes

JavaScript Class Constructors

Demystifying the JavaScript this Keyword

JavaScript instanceof

Aggregations and compositions

JavaScript Private Fields