JavaScript Runtime

Section 2

In this section, we dive into an exciting enhancement for our JavaScript runtime by creating a custom logging function. Our mission is to transform ordinary logging into an engaging experience with timestamps and witty, sarcastic messages.

#### **The Journey of Enhancement**

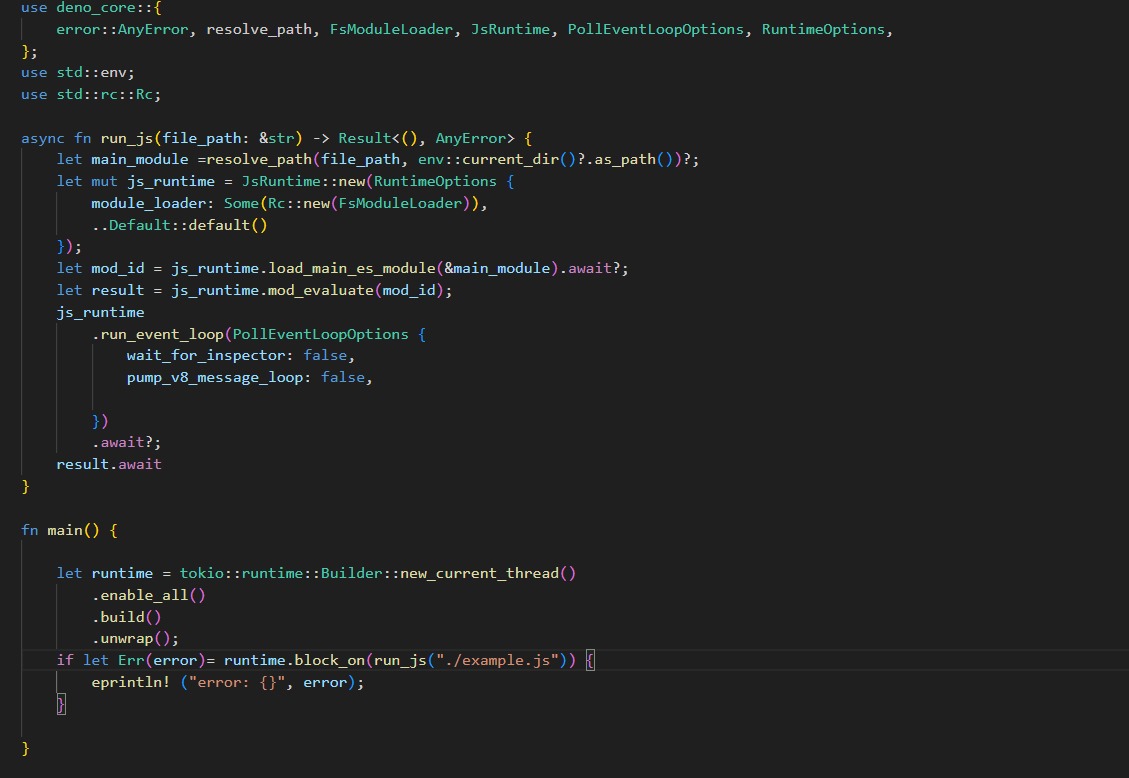
We embarked on this enhancement journey with the goal of making our runtime not just functional but also fun and expressive. We aimed to go beyond the basics and add a personal touch to our logging system. Here’s how we tackled it:

**1. Create a New Rust Project:**

* Open your terminal or command prompt.
* Navigate to the desired directory for your project.
* Run the command: cargo new my\_js\_runtime (replace "my\_js\_runtime" with your preferred name).
* This creates a new project directory with essential files like Cargo.toml and src/main.rs.

**2. Replace main.rs Content:**

* Open src/main.rs in your code editor.
* Replace its existing content with the provided run\_js function.

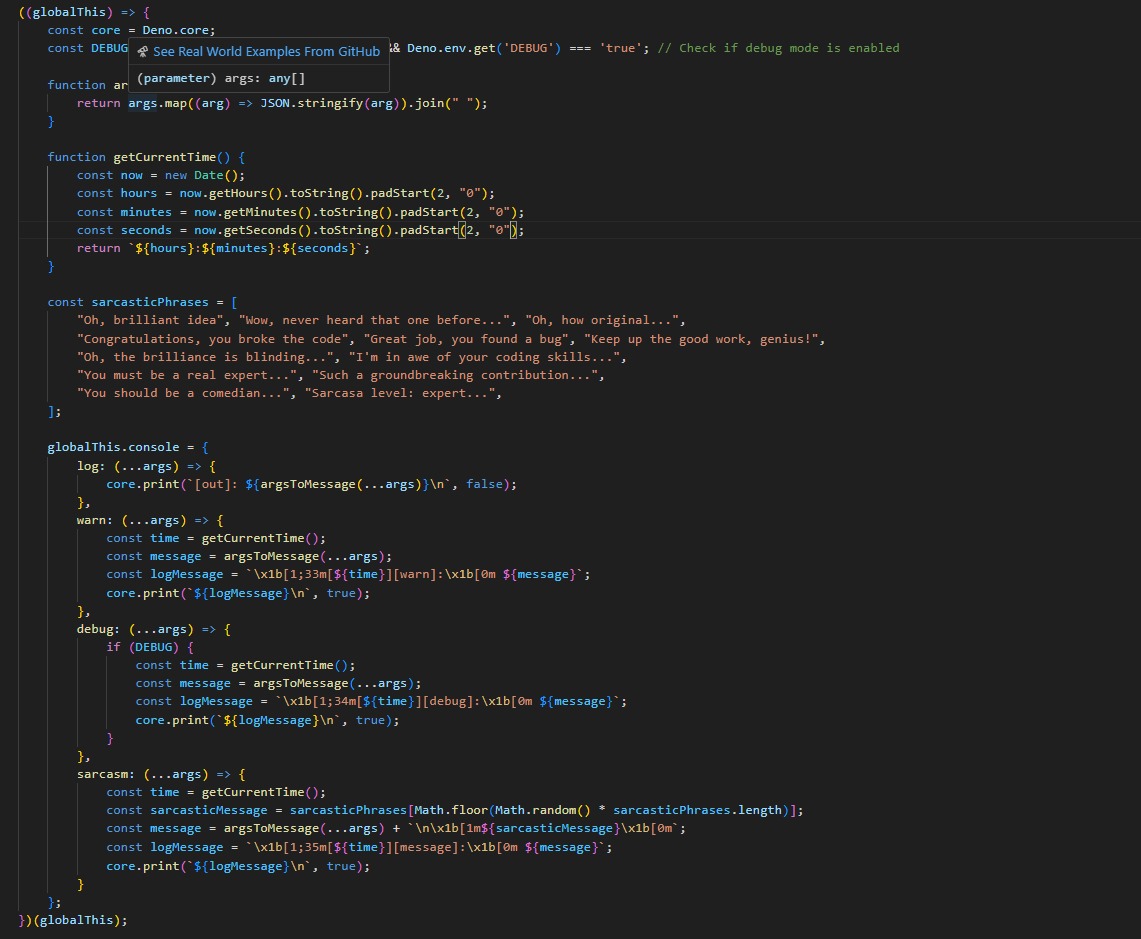


**3. Create runtime.js:**

* Navigate to your project's root directory (usually the same directory containing src).
* Create a new file named runtime.js using your preferred editor.
* Paste the content of the second code block you provided into this file. This code defines the custom console object with timestamped logs and the console.sarcasm function.

**4. Update Cargo.toml:**

* Open Cargo.toml in your editor.
* Locate the [dependencies] section (or add it if it doesn't exist).
* Add a line to specify the dependency on the deno\_core crate, which provides functionalities for interacting with JavaScript code



**5. Build and Run the Project:**

* In your terminal, navigate to your project's root directory.
* Run the command: cargo run
* If everything compiles successfully, the project will attempt to execute the example.js file (replace "example.js" in the main.rs code with your actual file path if it's different).

**6. Create example.js:**

* In the project root, create a new file named example.js using your editor.
* Add the following code to test the custom logging functions.

**7. Test the Output:**

* If the project runs successfully, you should see the following output in your terminal window.
* The [timestamp] indicates the time the log message was created.
* The sarcastic message from runtime.js appears after the regular log message.

**8. Testing runtime.js (Optional):**

This step is optional but demonstrates how to test the functionality of your runtime.js file independently.

* Create a new file named test.js in your project's root.
* Add code that simulates how your JavaScript code might interact with the custom console object from runtime.js.
* You can use tools like cargo test for unit testing in Rust projects, but this specific step focuses on testing runtime.js within the larger project context.

