Protecting the Object

Create a JavaScript object called `person` with properties `name`, `age`, and `email`. Implement the following:

- Make the `name` and `email` properties read-only.
- Make the `age` property write-only.
- Add a method called `getAge()` that returns the `age` property.
- Add a method called `setAge()` that accepts an `age` parameter and updates the `age` property.

JavaScript Prototype

Create a JavaScript object called `Vehicle` with properties `make`, `model`, and `year`. Implement the following:

- Add a method called `getDetails()` that returns a string with the vehicle's make, model, and year.
- Create a subclass called `Car` that extends `Vehicle` with an additional property `numDoors`.
- Override the `getDetails()` method in `Car` to include the number of doors in the string that is returned.
- Create an instance of `Vehicle` and an instance of `Car`, and call the `getDetails()` method on each instance, logging the result to the console.

Desired Coding Practices:

1. Code should be indented properly.
2. Code should be readable.
3. Code should handle any edge cases foreseen.
4. Add comments wherever required.
5. Follow meaningful naming conventions, avoid generic naming conventions such as a,b,c,x,y,z etc.

Hints:

Protecting the Object

- Use `Object.defineProperty() or any other Object methods` to create read-only and write-only properties.

JavaScript Prototype

- Use the `prototype` property to add methods to a JavaScript object.
- Extend a base object by creating a subclass with the `extends` keyword. Override methods from the base object by redefining them in the subclass.