1. What's a closure? Where in the code is there a closure?

A closure is a feature in JavaScript that allows a function to access and use variables that are defined outside of its own scope. Even if the function is executed and those outer variables are no longer in scope, the closure retains a reference to those variables, allowing the function to still access and use them.

In the code there is a closure being used in the computed property sortedPlanets.
In the sortedPlanets computed property, a function is defined as the getter for this property. This function utilizes the variables this.planet and this.sortBy from the outer scope. Even after the computed property is defined and the outer function has finished executing, the inner function retains access to these variables due to closure. The closure allows the sortedPlanets computed property to always have access to the current values of this.planets and this.sortBy, even if they change in the component's data over time. This ensures that the computed property remains reactive and automatically updates whenever the dependent variables change.

2. Which are the potential side-effects in any function? Could you point out any of these cases in your code? Are they expected? Can they be avoided?

Potential side effects in a function are changes or interactions that occur outside of the function's scope. In the code, examples of side effects include making a network request to fetch data and modifying component data based on the response.
These side effects are expected in this code because it's intended to fetch data from an API and update the component's state. To handle potential issues, the code includes error handling for the network request and manages the received data.
To properly handle side effects, such as error handling and data handling, and consider optimizations, you need to make sure your code behaves correctly and minimizes unintended consequences.