Zahava Gopin

WEB 330: Discussion 5.1

1/30/2023

Professor Krasso

JavaScript Generators

“A generator is a process that can be paused and resumed and can yield multiple values” (Rascia, 2020). A generator has two parts: a generator function and a generator object.

Using generator functions is a great way to avoid asynchronous code like async and await, which helps with readability. “Generators can maintain state, providing an efficient way to make iterators, and are capable of dealing with infinite data streams, which can be used to implement infinite scroll on the frontend of a web application, to operate on sound wave data, and more” (Rascia, 2020).

The most common way to create a generator function is by placing an asterisk next to the word function. Generator functions can be called within expressions or be methods of objects and classes. Even when return is called in the generator function a value is not returned right away. Instead, an iterable generator object is created. That object ca be assigned to a variable, which when called, goes into a suspended state. To actually get a value you need to use an iterator. “An iterator is an object that has a next() method available, which is used for iterating through a sequence of values. The next() method returns an object with value and done properties. Value represents the returned value and done indicates whether the iterator has run through all its values or not” (Rascia, 2020). The generator creates the values, and the iterator retrieves them.

Once all the values of the generator are called, the state of done is true and the function switches from suspended to closed. There are two other ways to close a function; return () and throw().

Resources:

Rascia, T. (2021, August 27). *Understanding generators in JavaScript*. DigitalOcean. Retrieved January 30, 2023, from https://www.digitalocean.com/community/tutorials/understanding-generators-in-javascript