QAP 1 – Full Stack JavaScript (Node.js) – By: Ellen Dalton

Task #1 – Understanding core global objects

**Events**

In Node.js, the events module is a core module that helps you build applications where certain actions or occurrences (events) can trigger responses (functions). The events module allows developers to create, emit, and listen for events within their applications. The main object within the events module is the ‘EventEmitter’, which provides methods to handle events, such as ‘on’ and ‘emit’. These are described below and examples are given in my code:

on(eventName, listener)

* on is used to add a listener function to the specified event. The listener will be executed whenever the named event is emitted.

emit(eventName, [args])

* emit emits the specified event, triggering the execution of all listeners attached to that event. Arguments can be passed to the listeners.

**Filesystem**

The "fs" (file system) module is an essential part of Node.js. It allows developers to perform various operations on files and directories, such as reading or writing files, creating directories, checking file existence, and more.

One common use case for the "fs" module is reading and writing files. Developers can use methods like ‘fs.readFile’ to asynchronously read the contents of a file, or ‘fs.writeFile’ for asynchronously writing files. These examples are shown in my example code. Additionally, the "fs" module supports file and directory manipulation operations. For example, developers can use ‘fs.mkdir’ to create a new directory or ‘fs.readdir’ to read the contents of a directory.

Error handling is an integral part of working with the file system in Node.js. Asynchronous operations typically involve callbacks that receive error information as their first parameter, allowing developers to handle errors gracefully.

**URL**

In Node.js, the ‘url’ module provides utilities for parsing and formatting URLs. It allows you to extract various components of a URL and manipulate them.

It provides a set of functions to work with URLs, allowing you to parse a URL into its individual components, construct URLs, and perform other operations.

One key function is ‘url.parse()’ which takes a URL string as input and returns an object containing properties like ‘protocol’, ‘hostname’, ‘port’, ‘pathname’, ‘query’, and ‘hash’. This object makes it easy to access and manipulate different parts of the URL.

In addition to parsing, the url module also provides functions like ‘url.format()’, which is used to construct a URL from its components.

Examples of these methods can be seen in my example code.