Write a blog post that answers the following questions. You should post it to your github.io account and attach the link in blackboard. Your blog post should NOT list each question and then answer it. Your post should address these three questions in the body of the post.

1. What is Node.js?
2. What makes node different than developing with more traditional server side languages?
3. What is the concept of non blocking IO?

A non-blocking IO allows the client to continue execution of code after making requests to the server by following an asynchronous model. Asynchronous means that the order of executions is not sequential within the code. This is performed by using callback functions. When the client makes a request to the server, the server immediately responds with a callback to the client , which allows the next queued function to be executed. When the server is finished with the initial request, it returns the data to the client. This follows the concept of non-blocking IO because code execution is never halted.

A server side platform which follows event-driven and non-blocking IO which allows for the creation of real-time applications. Node.js uses code libraries which are written in JavaScript and run on Google’s V8 engine. Node.js allows for asynchronous IO with HTTP and direct TCP sockets, allowing web applications to act as a standalone web server. It allows for thousands of connections simultaneously, but utilizes only one thread.

Until recently, web applications have utilized stateless request and response methods. This simply means that only the client can initiate a connection with the server, but the server cannot be the first to initiate connection with the client. Sequential server side languages such as PHP follow blocking IO processes which means that the client must receive a response from the server before it can continue execution. While the client waits for a response from the server, memory is frozen and the thread process stalls. If the client needs to make another request to the server during this period, a new thread must be created by the server. This consumes even more processing power and memory. Node differs from this approach by allowing client execution to continue after requests are made to the server, which greatly improves overall efficiency in applications which require real time communication with little computational requirements. Development is also different because code is not necisarily executed in a sequential order. This means that the order in which functions are written is irrelevant to their order of execution, while languages like PHP follow a sequential logic. Another major difference is that Javascript can be used for both the Server and Client side development whereas older languages required JavaScript knowledge and a separate server side language knowledge.