Node-Red for IoT, easy to develop simple html

Node Red : For more classical ETL works and maybe IOT projects.

It's standalone. Isn't designed to be distributed(as far as I know).

Apache Nifi is a little bit complicated but It's built for scalability.

You can have multiple nifi nodes and have higher throughput.

Nifi uses far more resources than Node Red.

The most important thing to understand is that one is "generally" not better than the other. The selection depends on the case in hand.

1. **Apache**generally uses PHP as a scripting language, which is extremely easy for the beginners and it is built for Web. **Node js** is simply a javascript v8 engine which was not originally intended to serve web.
2. **Apache and PHP** are old and stable. Everything you need to develop a working web application is available with a lot of support. **Node Js**is the new kid in the block ( .. not so new.. but still..). It is not essentially built for developing Web applications. http based Web application can be considered as one use of Node Js.
3. Many famous cms (Wordpress, Drupal, Joomla etc) and development frameworks (Yii, Laravel, Code Ignitor, Cake PHP) are build in **PHP**. These frameworks are very helpful in organizing the code base for a large application. **Node Js**also has web frameworks like Express and sails js but they are new and the support is not as readily available as compared to PHP frameworks.
4. **Apache** is thread and process based i.e each request is handled by a separate thread or process (depending upon configuration), which means if the process is waiting for the I/O, whole thread is blocked. **Node JS**has asynchronous, event driven I/O. Every nodejs instance runs in a single thread and due to its asynchronous nature, it can handle far more number of concurrent requests as compared to apache.
5. **Apache** is thread based, so if an unexpected problem occurs while processing a request, only that particular thread will crash leaving rest of the requests and threads intact. **Node Js**handles multiple requests using one single thread. If a problem occurs whole node js instance will crash along with any global data that was stored in javascript variables or arrays. It can be automatically restarted using "forever" or some other modules but the current requests and the data is gone.
6. when **Apache** gets a request which is CPU intensive, other request do not get blocked because of the context switching between the threads. when **NodeJs**gets a CPU intensive request, all the other requests get blocked till this CPU intensive request stops for an I/O. its a good idea to delegate CPU intensive requests to a worker or some other process while using Node Js.
7. **PHP** does not support web sockets natively (to my knowledge). There are libraries which can help implement web sockets, but again due to thread based model, a decent number of live web socket connections will eat up your server's resources. **NodeJs**is a perfect candidate for real time communication over internet. Combine it with [Socket.IO](http://socket.io/) and you will get a good scalable web socket server along with fallback communication mediums such as flash socket and long polling. Its light weight and it can scale pretty good because its all running in one single thread.