Interview Q&A sections:

Related: Big-Data

Q1: Difference between Apache spark and Hadoop?

Ans: **Apache spark**: lightning fast cluster computing system/framework with high-level API for scala, java, R, python.

* Speed: Runs applications 100 times faster in memory and 10 times faster accessing data process from disk than Hadoop. Because of reducing the number of read/write cycle to disk and storing intermediate data in-memory Spark makes it possible.
* Difficulty: Spark is easy to program as it has tons of high-level operators with RDD – Resilient Distributed Dataset.
* Easiness: Installing Spark on a cluster will be enough to handle all the requirements like performing batch, interactive and Machine Learning and Streaming all in the same cluster. As a result makes it a complete data analytics engine. Thus, no need to manage different component for each need.
* Real-time stream: can process real time data i.e. data coming from the real-time event streams at the rate of millions of events per second, e.g. Twitter data /facebook share.
* Fault tolerance: Spark is fault-tolerant so no need to restart the apps from scatch.
* Better for large scale data like GB to PB etc.
* Spark can use Hadoop components like HDFS.

**Hadoop**: Hadoop is an open source, Scalable, and Fault tolerant framework written in Java. It processes large volumes of data on a cluster of commodity hardware. Hadoop is not only a storage system but is a platform for large data storage as well as processing.

* **Hadoop MapReduce**– MapReduce reads and writes from disk, as a result, it slows down the processing speed.
* In MapReduce, developers need to hand code each and every operation which makes it very difficult to work. MapReduce only provides the batch engine. Hence, we are dependent on different engines. For example- Storm, Giraph, Impala, etc. for other requirements. So, it is very difficult to manage many components.
* MapReduce fails when it comes to real-time data processing.
* Like Apache Spark, MapReduce is also fault-tolerant. Same here
* Better for small size data like 100mb than spark

**JavaScript:**

**Q1: let, var, const**

Let: variable in block/private method

Const: variable fixed/constant not changeable later

Var: variable can be changed and work outside any block/private class or method

Q2: Closures:

A **closure** is the combination of a function and the scope object in which it was created. Closures let you save state. Means you can temporary have state of function reference until there is no other ref or function call for this. it is an expression that can reference variables within its scope (when it was first declared), be assigned to a variable, be passed as an argument to a function, or be returned as a function result.

So its like a function inside function where need 2 parameter but got one but still works until get another value as like ref. or waiting

function makeAdder(x) {

return function(y) {

return x + y;

};

}

var add5 = makeAdder(5);

var add10 = makeAdder(10);

console.log(add5(2)); // 7

console.log(add10(2)); // 12