**Question 1. (a)**

[**What is the difference between statically typed and dynamically typed languages?**](https://stackoverflow.com/questions/1517582/what-is-the-difference-between-statically-typed-and-dynamically-typed-languages)

**Answer:**

**A programming language is statically-typed if the type of a variable is known at compile time.**

**A language is dynamically-typed if the type of a variable is checked during run-time.**

**Statically-typed :**

We call a language as statically-typed if it follows type checking during compilation. So, every detail about the variables and all the data types must be known before we do the compiling process.

In this type of language, once a variable is assigned a type, it can’t be assigned to some other variable of a different type. If we try to do so, the compiler will raise some errors, and we need to fix them. Hence, for a declared variable, the data type is fixed. Some examples of statically-typed languages are Java, C, C++, C#, Swift, Scala, Kotlin, Fortran, Pascal, Rust, Go, COBOL, etc.

**Dynamically typed :**

We call a language as dynamically-typed if type checking takes place while the program runs (run-time). In this type of language, there is no need to specify the data type of each variable while writing code.

It means that you can write pretty quickly since you do not have to specify types every time. Some languages do allow you to provide type information but do not require it.

Most of the modern programming languages are dynamically-typed. Some examples of dynamically-typed languages are Python, Javascript, Ruby, Perl, PHP, R, Dart, Lua, Objective-C, etc.

---------------------------------------------------------------------------------------

**Question 1 (b)**

**What’s the difference between Scripting and Programming Languages?**

**Answer :**

Basically, all scripting languages are programming languages. The theoretical difference between the two is that scripting languages do not require the compilation step and are rather interpreted. For example, normally, a C program needs to be compiled before running whereas normally, a scripting language like JavaScript or PHP need not be compiled.  
   
Generally, compiled programs run faster than interpreted programs because they are first converted native machine code. Also, compilers read and analyze the code only once, and report the errors collectively that the code might have, but the interpreter will read and analyze the code statements each time it meets them and halts at that very instance if there is some error. In practice, the distinction between the two is getting blurred owing to improved computation capabilities of the modern hardware and advanced coding practices.  
   
Another point to be noted is that while classifying a language as scripting language or programming language, the environment on which it would execute must be taken into consideration. The reason why this is important is that we can design an interpreter for C language and use it as a scripting language, and at the same time, we can design a compiler for JavaScript and use it as a non-scripting(compiled language). A live example of this is V8, the JavaScript engine of Google Chrome, which compiles the JavaScript code into machine code, rather than interpreting it.  
   
Some scripting languages traditionally used without an explicit compilation step are JavaScript, PHP, Python, VBScript.  
   
Some programming languages traditionally used with an explicit compilation step are C, C++.  
   
   
**Applications of Scripting Languages :**

1. To automate certain tasks in a program

2. Extracting information from a data set

3. Less code intensive as compared to traditional programming languages

**Applications of Programming Languages :**

1. They typically run inside a parent program like scripts

2. More compatible while integrating code with mathematical models

3. Languages like JAVA can be compiled and then used on any platform

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

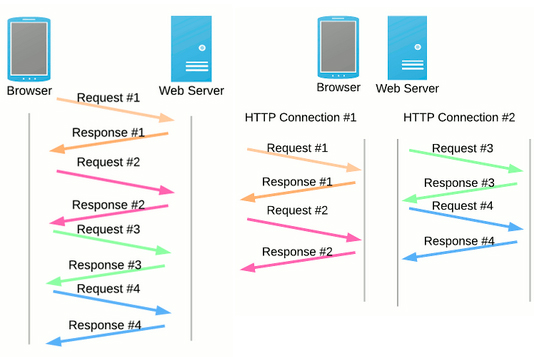
---------------------------------------------------------------------------------------\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 3**

**Difference between HTTP1.1 vs HTTP2 ?**

**Answer :**

HTTP2, the new Web protocol slated to go live any day now, aims to be a faster, more efficient protocol. HTTP1.1 is the current predecessor and has been around for about 15 years. The problem with HTTP1.1 is that can only load requests one at a time, one request per one TCP connection. Basically, this made browsers run parallel requests to multiple TCPs for the same Web asset. This clogs up “the wire” with multiple duplicate data requests, and can hurt performance if too many requests are made.

Image Credit: The Moz Blog

In short, HTTP1.1 makes loading a web page more resource-intensive than ever. This led to the industry utilizing Best Practices like concatenation, data inlining, and domain sharding in an attempt to fix the underlying problems of the protocol.

## HTTP & SPDY: The Basis Of HTTP2

Enter SPDY in 2012. SPDY was the next open-source protocol that was developed, this time by Google, in an attempt to reduce web page load latency and increase security. SPDY modified the way HTTP requests and responses are sent over the wire and made an effective base for HTTP2. Just recently Google announced it will be removing SPDY in favor of HTTP2 and that we can [expect HTTP2](http://arstechnica.com/information-technology/2015/02/http2-finished-coming-to-browsers-within-weeks/%3Futm_source=sendgrid%26utm_medium=email%26utm_campaign=top10) to come to browsers in a few weeks.

These are the high-level differences between HTTP1 and HTTP2:

* HTTP2 is binary, instead of textual
* HTTP2 is fully multiplexed, instead of ordered and blocking
* HTTP2 can, therefore, use one connection for parallelism
* HTP2 uses header compression to reduce overhead
* HTTP2 allows servers to “push” responses proactively into client caches

 HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, HTTP/2 is able to use a single [TCP](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) connection to send multiple streams of data at once so that no one resource blocks any other resource. HTTP/2 does this by splitting data into binary-code messages and numbering these messages so that the client knows which stream each binary message belongs to.