Applications are of 4 types

1. desktop application—These are single user application
2. Mobile application---The application runs mobile is mobile application
3. Web application---Any application which runs on server, and displays o/p in the browser in HTML form
4. Web service---Any application which runs on server, and displays o/p in json format

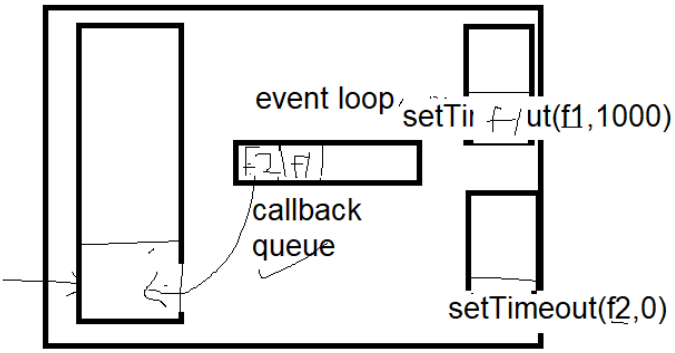
Download tomcat 9

https://tomcat.apache.org/download-90.cgi#9.0.89

How the program works in nodejs-→ refer file Nodejs architecture.doc for more details

```
console.log("begin the file")
setTimeout(f1,1000)
setTimeout(f2,0)
console.log("after set timeout f1")

f1()
f2()
```



Nodejs use modular approach, so there are multiple built in modules are there

1. buffer --- all buffer related functions are in buffer module, when you need to use buffer module, you need not import the module
2. fs-→ all file system related functions(like read, write,delete, copy file functions) are in fs module, to use these functions, you need to import the module
3. http-→ all server related functions are in this module

You may create a user defined modules, any module is a javascript file which has many public/exported functions, which can be used in another file

To import user defined module always use relative path

e.g --→ const m1=require(“./module1”)

To create user defined modules, in the .js file, the functions added in exports object, can be used outside the file, those are public functions

1. create a file by name module1.js

module1.js	usemodule1.js

exports.f1=function(){
console.log(“in f1”)
}
exports.f2=function(){
console.log(“in f2”)
}
	const m1=require(“./module1”)
	m1.f1()
	m1.f2()
	console.log(“in usemodule 1 file”)

functions in fs module

to use functions in fs module, you need to import fs module

const fs=require(“fs”)

var data=fs.readFileSync(“test.txt”)

var data1=fs.readFileSync(“test2.txt”)

- every function in fs module comes in 2 formats, Synchronous and asynchronous
- every synchronous function will block the code, further code will not get executed, unless the current statement execution is over
- every asynchronous function, will run parallely, in these functions the last parameter is always a callback function, and the first parameter of every callback function is err, and the next parameter is data to be processed

readFile
 	
in async function printing data happens in the callback function
	const fs=require(‘fs’)
	fs.readFile(“test.txt”,function(err,data){
	if(err)
	console.log(err);
	else
	console.log(data.toString(),data.length)
	})

readFileSync in sync function printing data happens after the synchronous function is over	const fs=require(“fs”) var data=fs.readFileSync(“test.txt”) console.log(data);
write	writes data in the file in asynchronous way
writesync	writes data in the file in synchronous way