Okay, I've looked at the docs for V8 again.. and it doesn't look good.. There are two things: and "isolate" and a "context"..
The isolate sounds like a complete separate environment, including different heap. This sounds bad because what we want is for multiple scripts within the same application to work on the same variables, in parallel. So, if each script has to run in a different isolate, then each will have a different heap.. so, the question is whether a script running in one isolate can access variables on the heap that is inside a different isolate?
Do you know what a heap is? It is the memory where "new" things are placed.. in C, each call to malloc creates a chunk of data that is "on the heap". In javascript, each invocation of "new" creates an object that is "on the heap". So, two different isolates will create "new" objects in two different heaps.. the question is whether scripts inside one isolate are able to read and write objects on the heap of a different isolate.
That can be tested!

The 2nd code I use one global isolate that can be used by multiple isolate, and this worked and so they have the same heap and can use the same variable.

Regarding the first code, I create for every thread one isolate, and this compiled and worked but without printing the script of the other thread,

So now you need to test to make two different isolate and try to test if an isolate are able to read and write objects on the heap of a different isolate?
so I'll have a look to see how we could do this to test that.

If so, the next question is: will objects in one isolate be mistakenly garbage collected?
The js engine checks how many objects contain a handle by which they can access a given object on the heap. So, for a given heap object, it counts how many other objects have a handle to it. If there are zero, then nothing can access the heap object, so it gets deleted. Soooooo.. if a script running in one isolate is accessing a heap object that is inside a different isolate, then the garbage collector may not see that "foreign" object, and mistakenly believe nothing is accessing the heap object and delete it.
That can also be tested.

So this mean that the garbage collector only see the current heap.

First case: if one script accessing an object in different isolate, this mean that the script reads and writes the object and if doesn't create and allocate a heap memory to it in its isolate, so why garbage collector would delete it? like this
handle<String> = string_of_different_isolate, it will saved inside the stack and deleted at the end of handle scope, unless we don't use persistent handle.

2nd case: And if it creates a heap memory like this,
handle<String> s = String::New();
s = string_of_different_isolate;
so here it creates a heap memory to it and has a handle so why garbage collector will delete it.

3rd case: if the accessed object is a shared object between the 2 scripts then this mean it will be created inside one isolate and the other isolate can see it and access it within the creating isolate.

actually I confused in this part, may be cause I don't know how this object will be accessed through different isolates, I'll see V8 API maybe there is something that do this.

If both tests pass... then it might be possible that a separate thread can be safely assigned to each isolate. Then, that thread is the only one to ever run scripts inside that given isolate. The scripts in different isolates work on common data. That way, multiple threads are simultaneously working on common data.
so, that's one possible path: multiple isolates, with a different thread assigned to each isolate.
The other possibility is that a context is enough.. I don't hold much hope for this.. a context is only a separate set of global variables ("built-in" variables). However, multiple contexts exist inside the same isolate, and so share the same heap (I think)..
So, some tests can be done there.. assign a different thread to each context, and run the scripts in the different threads/contexts.. see what happens..
So, how about doing that next? Let's see.. try starting with making two contexts..One question I have is about whether a source string, and the compiled script made from that exist in particular isolates and contexts:So, a script's source string appears to be "created" inside an isolate:

Handle<String> source = String::NewFromUtf8(isolate, "'Hello' + ', World!'");

But when it is compiled, there is no mention of isolate nor context:

Handle<Script> script = Script::Compile(source);

However, in the hello world example, before these actions were performed, the context was set:

Context::Scope context_scope(context);

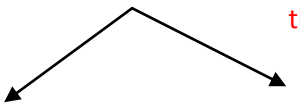
This command make this context the "current" context.. so the creation of the string and the compile both happened inside this context..
I don't know if the compiled script is now bound, somehow, to the context (and isolate) inside which it was compiled..

mmm.. I think this case is our 2nd code, as I used only, one isolate and used two different context; one inside the main thread and the other inside the thread function.

In any case, do this:-] Create one isolate.-] create the C++ strings of the two javascript functions.-] create two "child" threads of the current thread.-] give both threads the same birth function, but pass one string to one, the other string to the other-] inside the birth function:-] create a context, inside the isolate-] set the context active-] compile the string that was passed in-] execute the resulting compiled script-] end the thread.
I expect that to break.. run it a whole bunch of times, if it doesn't..
Once it breaks, try creating the contexts in the main thread, and compiling the scripts in the main thread.. run them both in the main thread, one after the other.. then create the two child threads, and pass one compiled script to one, the other to the other.. also pass

the contexts to the child scripts.. have the threads set the contexts, then run the compiled scripts.. do that a bunch of times, see if it crashes..
How does that sound?

so the **first test is:**
1- New isolate inside the main thread
2- 2 New C++ strings
3- create 2 threads with the same function



1- new context
2- compile the string
3-execute the compiled string

and run it many times

2nd test:
1- new isolate
2- create 1st context
3- New C++ string contains src script
4- compile it
5- run it
6- create 2nd context
7- New C++ string contains src script
8- compile it
9- run it
10-create two child thread (compiledScript, context)