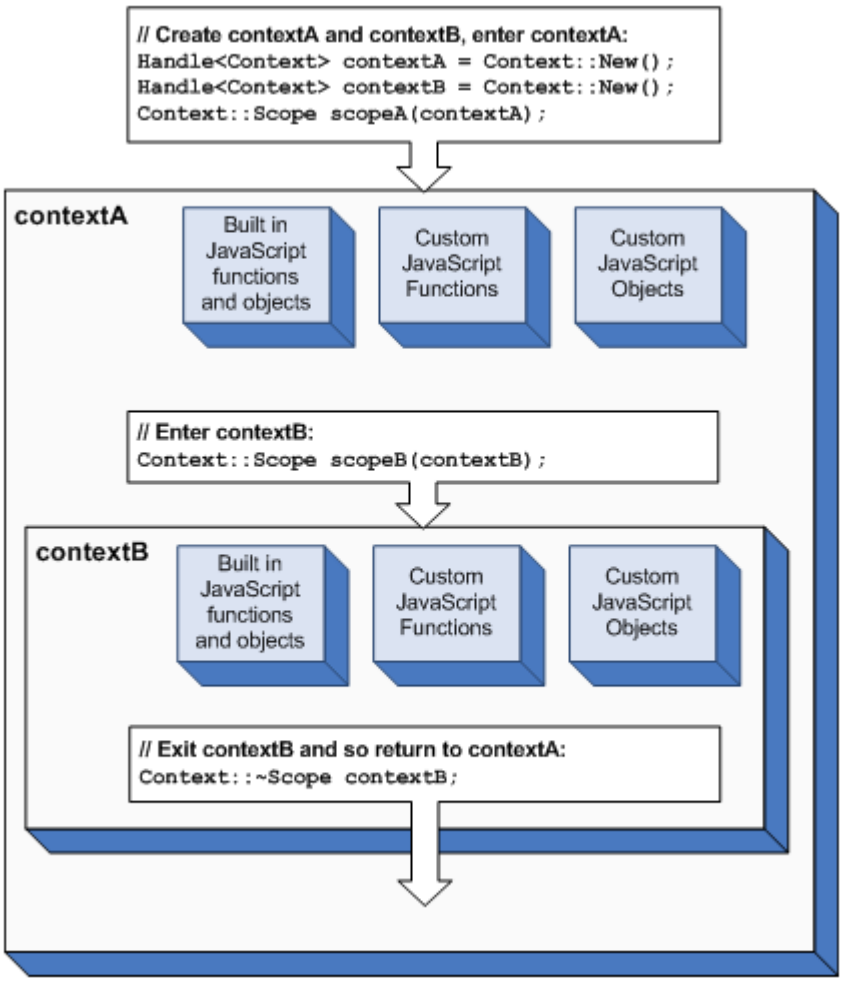
1- Your C++ code accesses V8 through the V8 API by including the header include/v8.h.
2- All objects are accessed using "handle", which provides reference to javascript object in the heap
3- The V8 garbage collector reclaims memory used by objects that can never again be accessed.
4- Handle scope is a container that holds lots of handles. When the handle scope's destructor is called all handles created within that scope are removed from the stack.
   Destructor : HandleScope::~HandleScope
5- There are two type of handle:
   a- local handle: can be deleted by calling its destructor or deleting its handle scope which handle is created in.
   Handle<SomeType>.
   Pitfalls: *you cannot return a local handle directly from a function that declares a handle scope*.
   Solution : The proper way to return a local handle is to call the Closemethod on the handle scope, passing in the handle whose value you want to return, and returning the new handle you get back from the Close call
   b- Persistent handle: is deleted by calling its destructor only, and can keep a reference to an object for more than one function call.
   Persistent<SomeType>, Persistentconstructor , Persistent::Dispose, Persistent::MakeWeak

   They look like global object and local object.

6- To return something from a function like local handle, you use handle scope.close(returned handle) this will return new handle contains the value of the returned handle and the local handle will be reclaimed by garbage collector. "The `Close` method copies the value of its argument into the enclosing scope, deletes all its local handles, and then gives back the new handle copy which can safely be returned."
7- context is an execution environment that allows separate, unrelated, JavaScript applications to run in a single instance of V8
   When you have created a context you can enter and exit it any number of times. While you are in context A you can also enter a different context, B, which means that you replace A as the current context with B. When you exit B then A is restored as the current context. This is illustrated below:



8- Template is used to wrap C++ functions and data structures within JavaScript objects so that they can be manipulated by JavaScript scripts.

You can create a set of templates and then use the same ones for every new context you make. You can have as many templates as you require. **However you can only have one instance of any template in any given context.**

8.1- In JavaScript there is a strong duality between functions and objects, to create a new type of object, you will create a new function & create instances using the function as a constructor.

8.2 Function template: You create a JavaScript instance of the template by calling the template's `GetFunction` method from within the context in which you wish to instantiate the JavaScript function.

8.3 Object templates

Each function template has an associated object template. This is used to configure objects created with this function as their constructor. You can associate two types of C++ callbacks with object templates:

- **accessor** callbacks are invoked when a **specific object property** is accessed by a script, Accessors are configured through an object template, using the `SetAccessor` method. This method takes the name of the property with which it is associated and two callbacks to run when a script attempts to read or write the property.

To see accessing static global https://developers.google.com/v8/embed#static_global

To see accessing dynamic variable
https://developers.google.com/v8/embed#dynamic

Difference between external object and handle in V8?
answer: External objects can only be used to store reference values in internal fields. JavaScript objects can not have references to C++ objects directly so the external value is used as a "bridge" to go from JavaScript into C++. In that sense external values are the opposite of handles since handles lets C++ make references to JavaScript objects.

Also, each JavaScript object template keeps a reference to the C++ object for which it is a wrapper with an internal field. cannot be accessed from within JavaScript, they can only be accessed from C++ code. An object can have any number of internal fields, the number of internal fields is set on the object template as follows:

```
point_templ->SetInternalFieldCount(1);
```

But what is exactly the role of internal field or what did they mean by internal field? Actually I don't sure of what I guess is right or wrong.

- **Interceptor** callbacks are invoked when **any** object property is accessed by a script

There are two types of interceptors:

- named property interceptors - called when accessing properties with string names. An example of this, in a browser environment, is document.theFormName.elementName.

- indexed property interceptors - called when accessing indexed properties. An example of this, in a browser environment, is document.forms.elements[0].

9- Exception

V8 will throw an exception if an error occurs - for example, when a script or function attempts to read a property that does not exist, or if a function is called that is not a function.

Also, V8 returns an empty handle if an operation did not succeed. It is therefore important that your code checks a return value is not an empty handle before continuing execution. Check for an empty handle with the `Handle` class's public member function `IsEmpty()`.

10- Inheritance :
javascript uses prototypal inheritance instead of classes inheritance, which is objects inherits from another objects, or instance by adding **prototype** keyword on the object before adding the custom property to it. The same approach is used in V8 by

introducing PrototypeTemplate method to FunctionTemplate, you can set properties, and associate C++ functions with those properties, on a `PrototypeTemplate` which will then be present on all instances of the corresponding`FunctionTemplate`.
Also, V8's `FunctionTemplate` class provides the public member function `Inherit()` which you can call when you want a function template to inherit from another function template