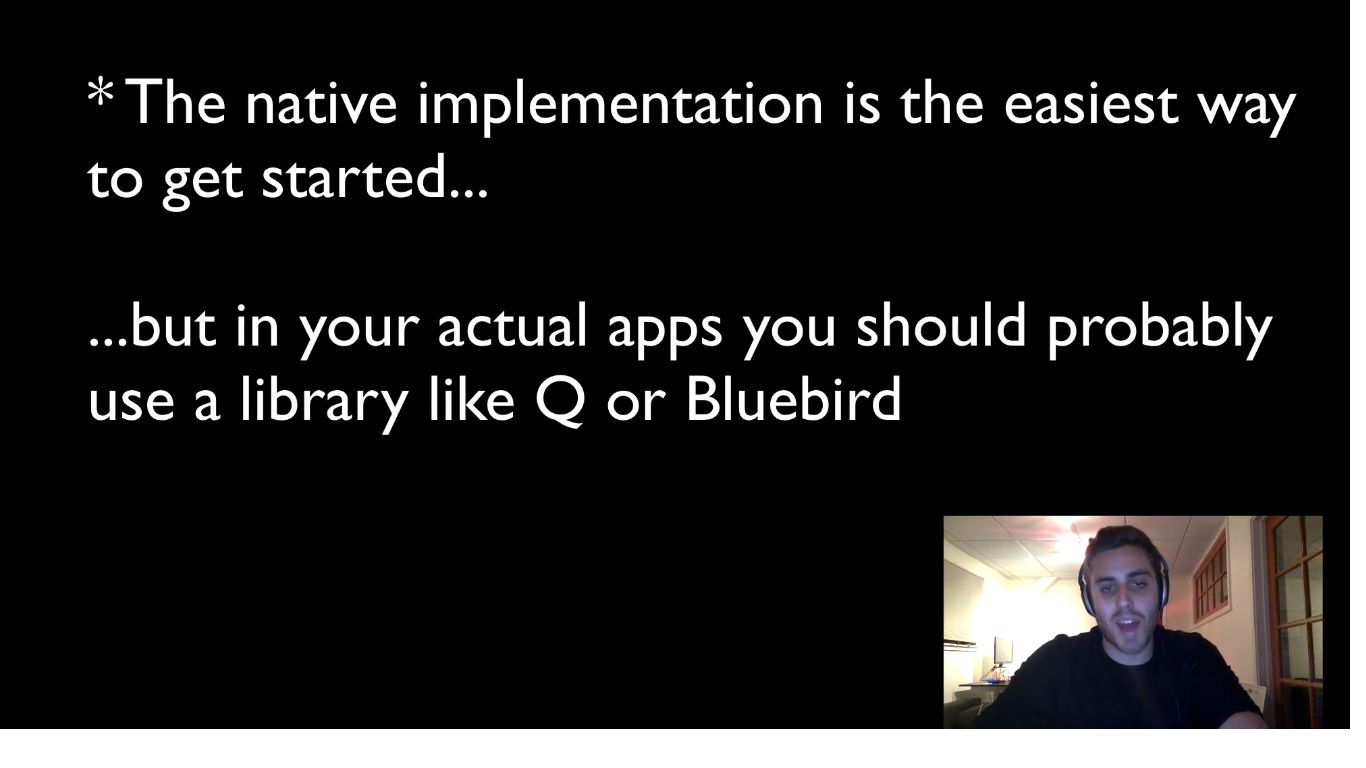
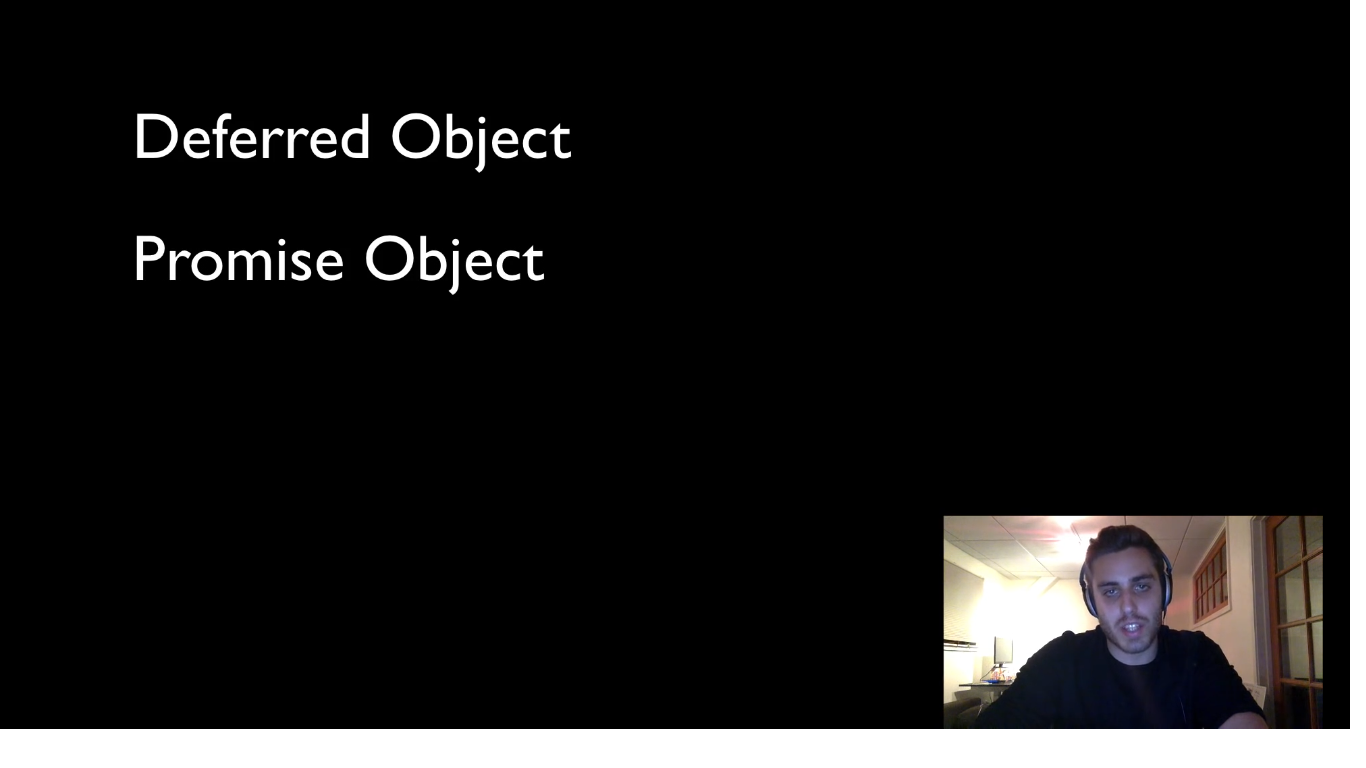
How to use Promises

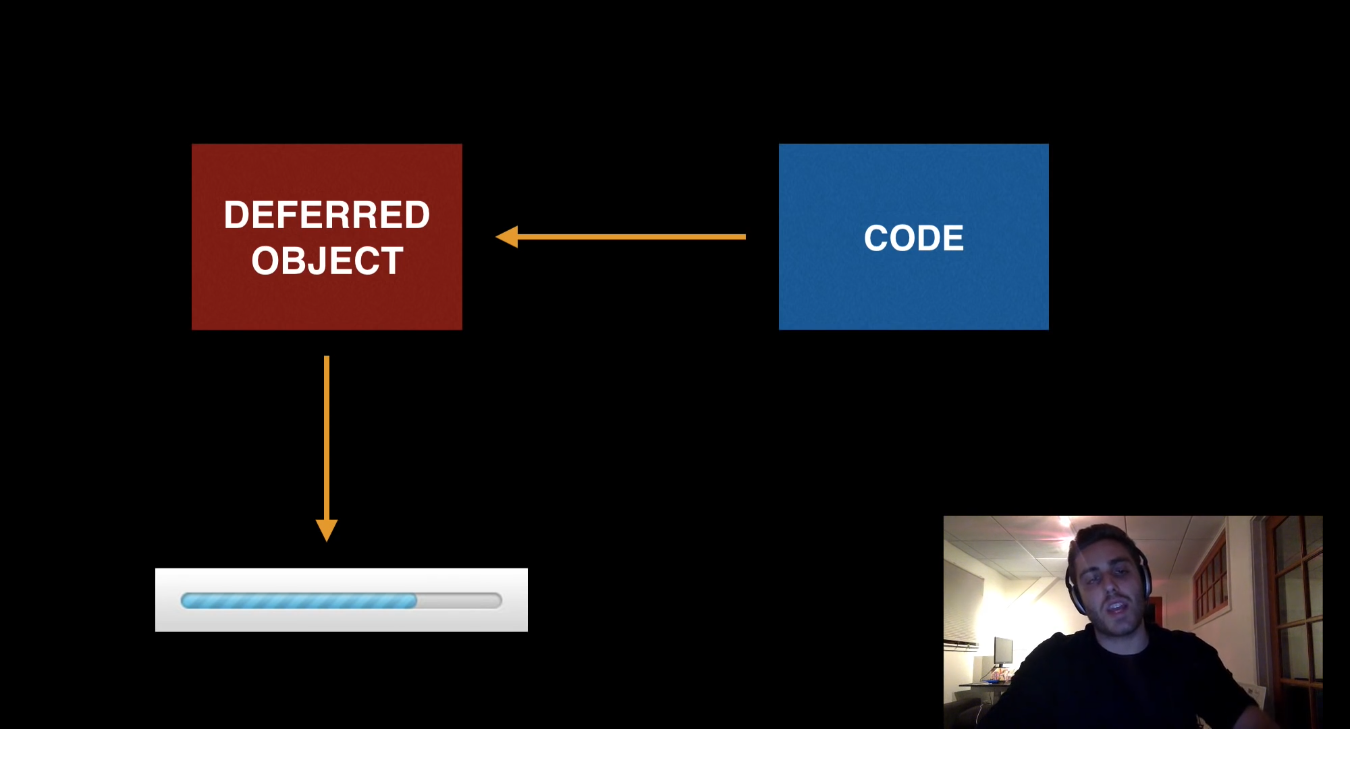
At high levels there are two objects : 1. Deffered object and 2. Promise Object

You can think of a Promise like a progress bar, which is currently in progress and it is personified as js object.

­­­

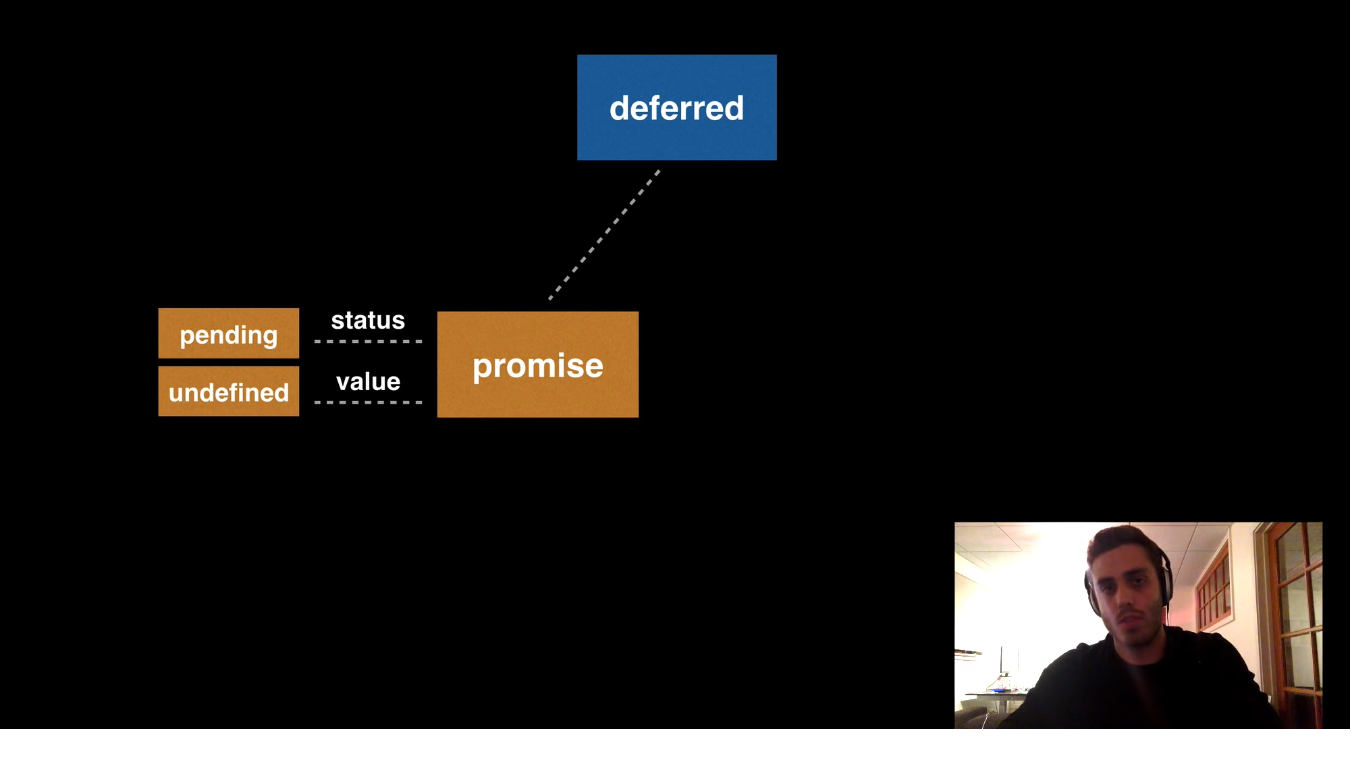
Callbacks doesn’t have that ability to access that in progress object.

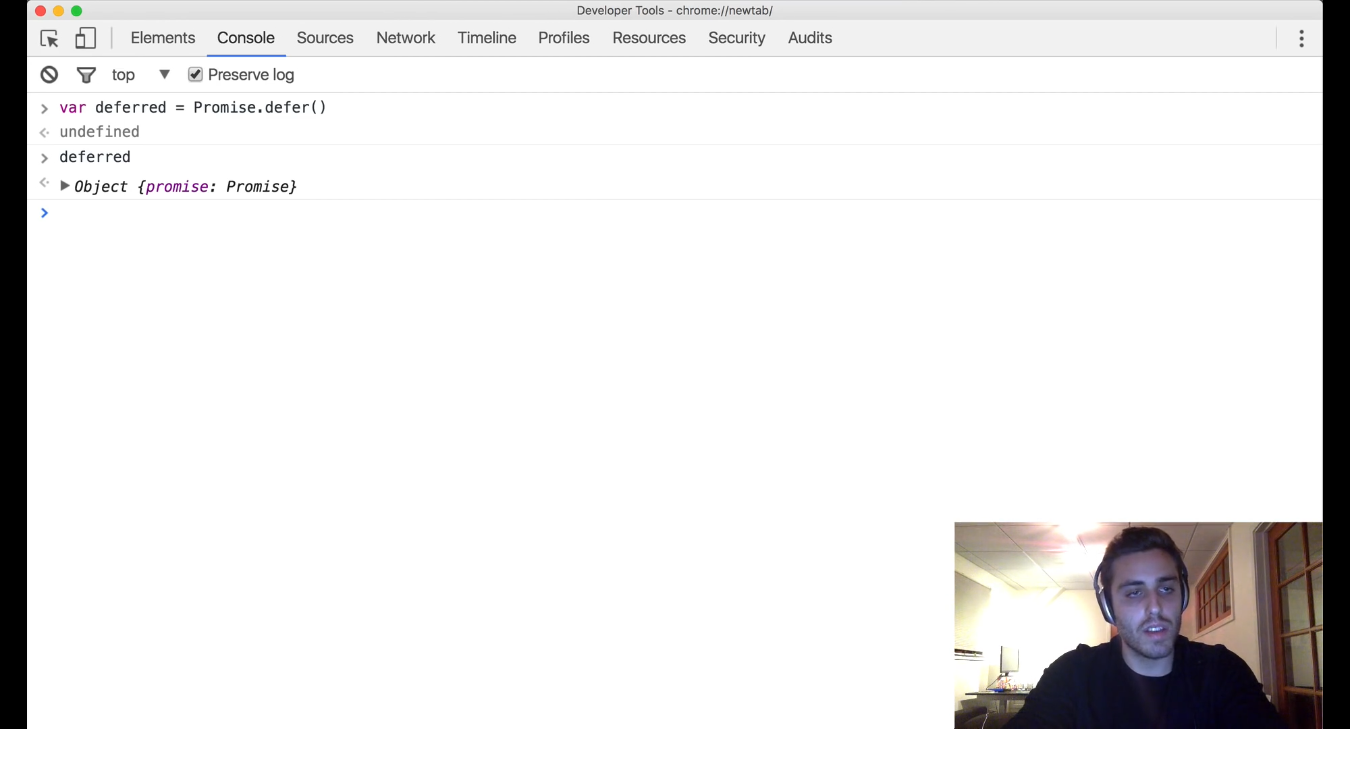
Deferred object is interface to that progress bar (promise object)



So we don’t make updates to the progress bar directly, it is updated through this deferred object.

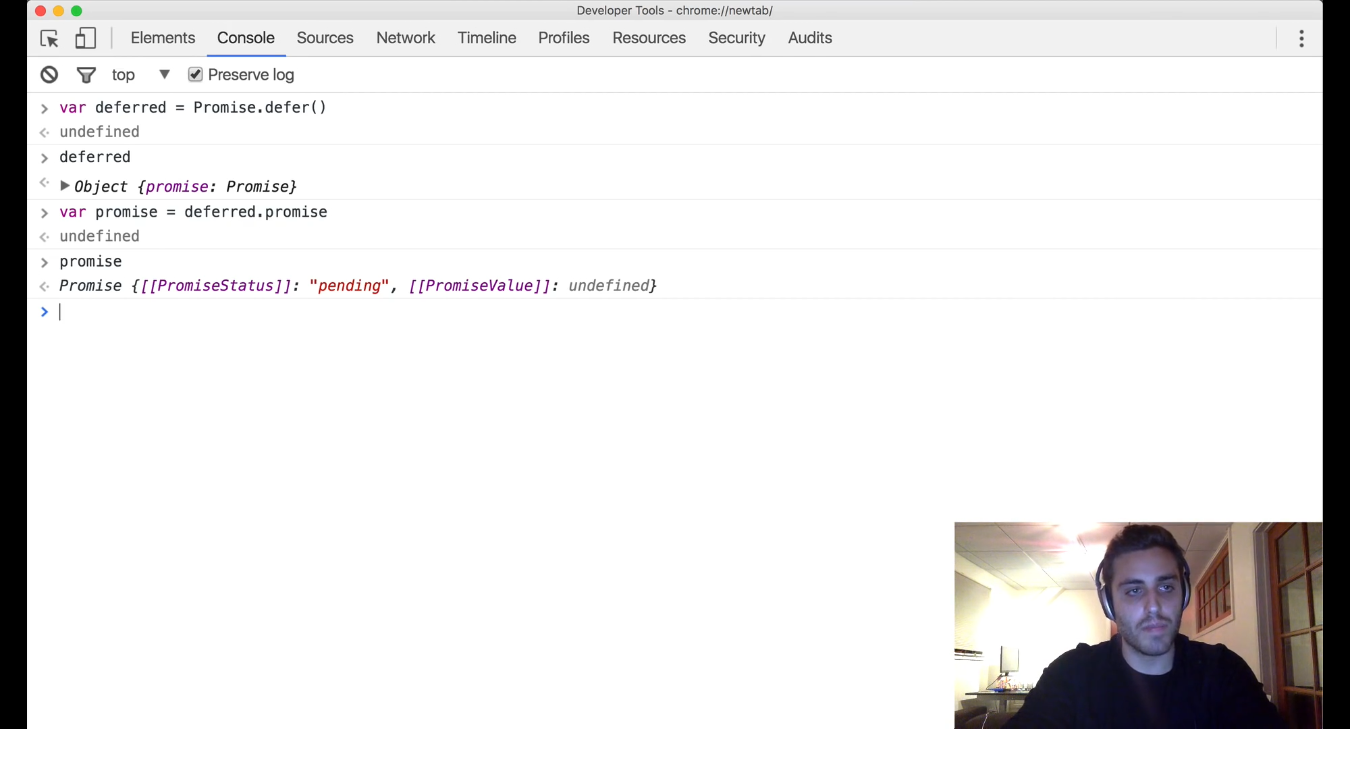
The Deferred object is just a js object, which has property on it called as promise. Which is the Promise object. Promise itself has two properties : status & values. By default the status is pending and value is undefined.





Let’s get this promise property in separate variable:

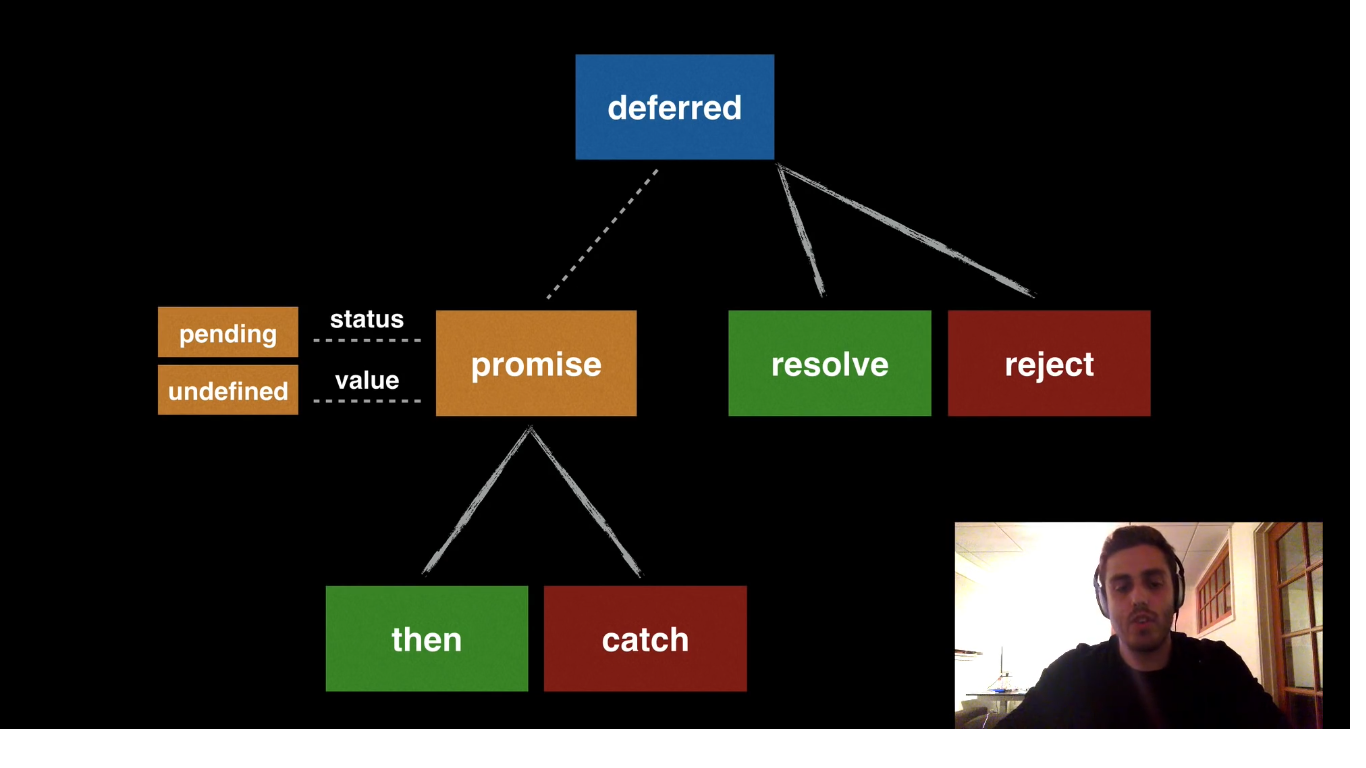
var promise = deffered.Promise, and if we see that that has status pending and value is undefined.



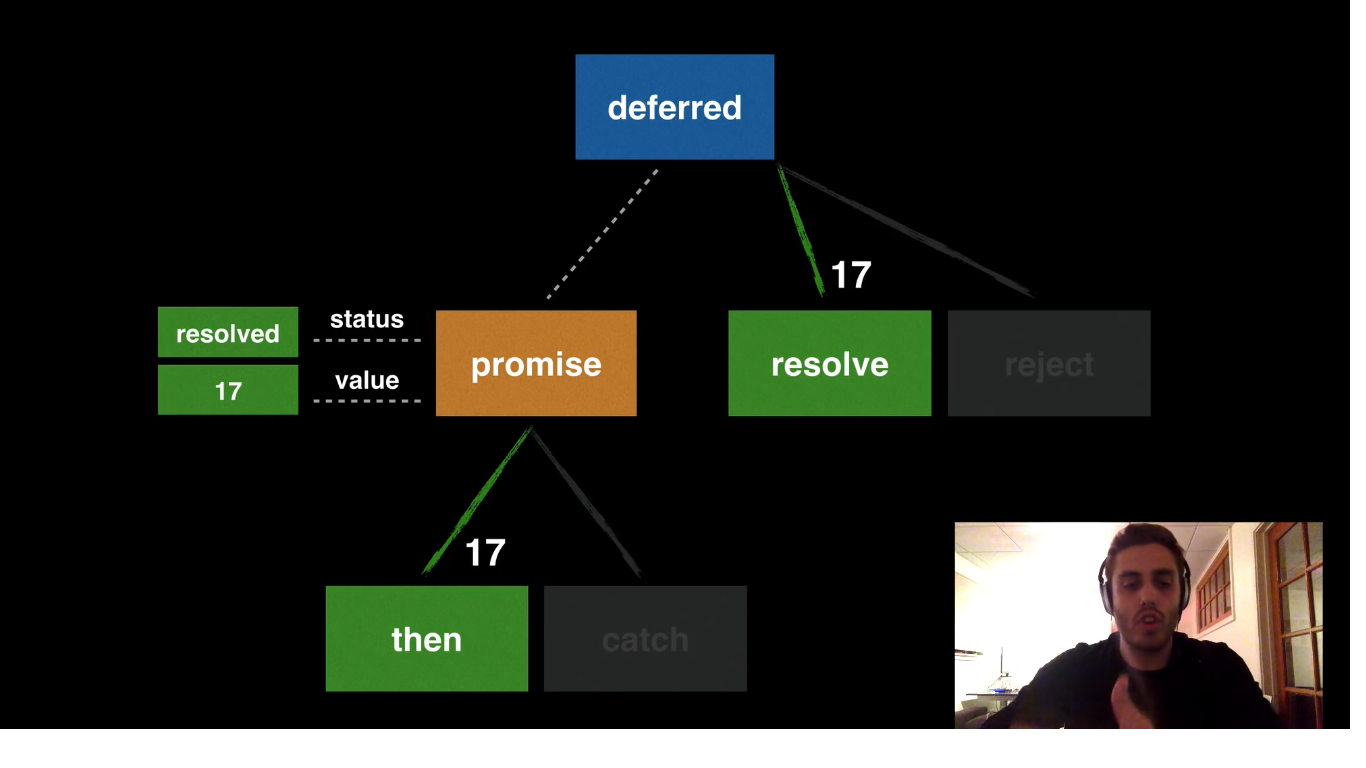
This deffred object also has two functions to call : 1. Resolve 2. Reject

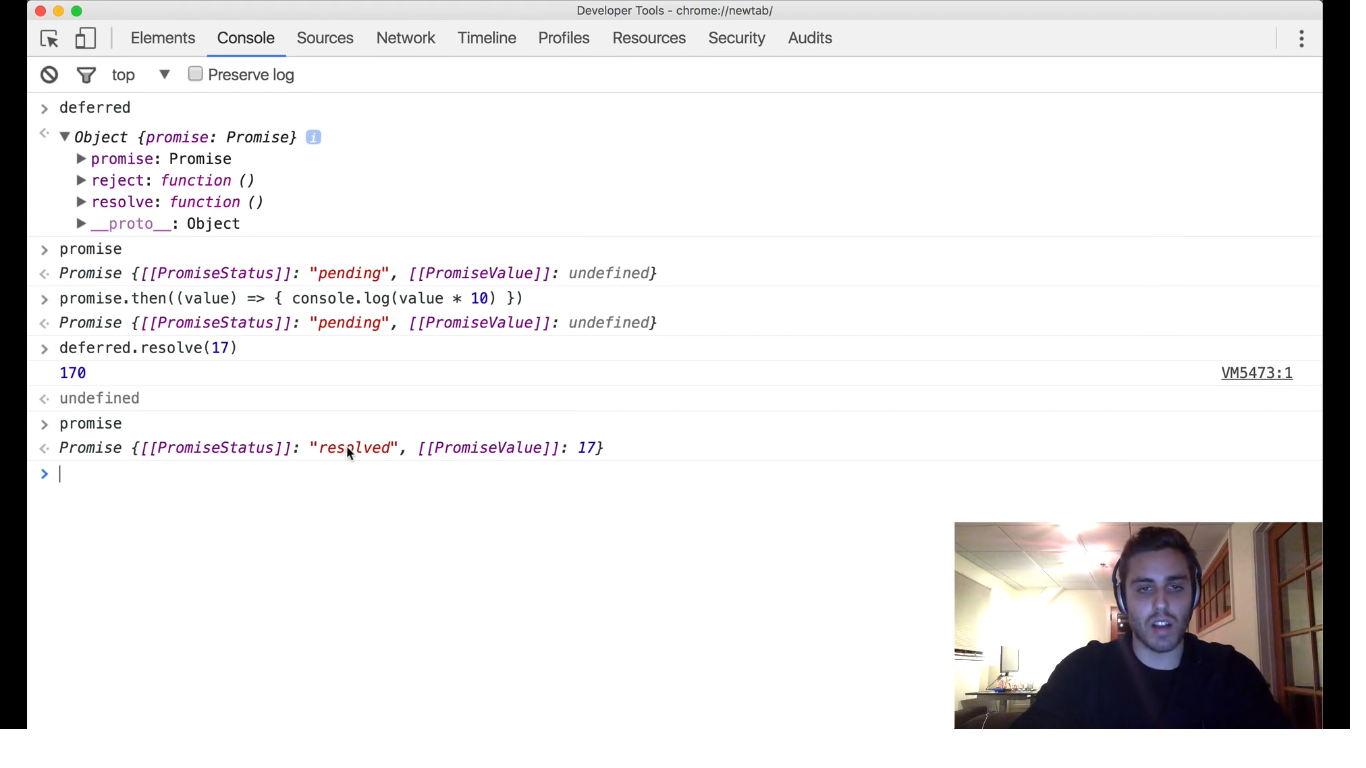
The property of deferred object i.e. Promise also has two functions to call : 1. Then 2. Catch

In the following picture, the dashes are properties and bold lines are functions.



Resolve says : Now nothing is in progress. It has finished and now we have its value. That becomes the value of the promise. And then any function waiting for this value now uses that value to continue its execution.





Normally we pass a js error object, when a promise gets rejected, to reject function. If you reject a promise you are no longer ever resolve that promise. The status of the promise gets rejected, the value becomes that error object. And we go down to catch branch.

