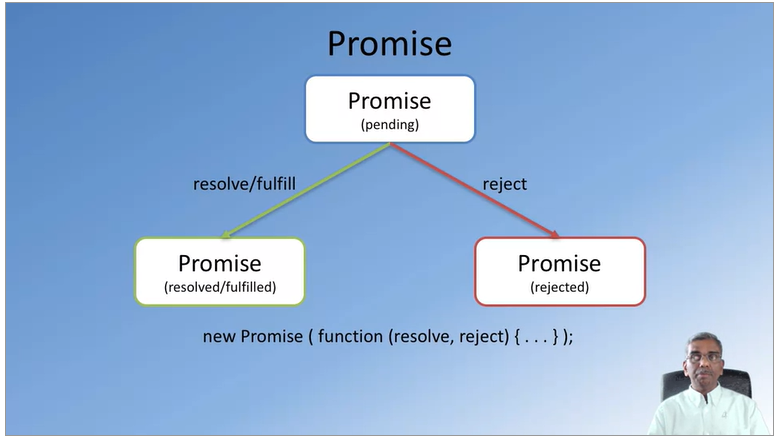
**Callback problems**

* Heavily nested callback code
  + Results from our tendency to write code top to bottom
  + Pyramid of doom
  + We can work around this problem by not using anonymous functions for the callbacks, and instead declaring the functions with specific names (there are other ways to avoid this problem
* Promises
  + A way to avoid the callback problem
  + Tries to preserve the top-down appearance of the code
  + Mechanism that supports asynchronous computation
  + Proxy for a value not necessarily known when the promise created
    - Represents a value that may be available now, or in the future, or never
    - Whenever a promise created (you create a resolve and a reject function
  + Why promises
    - Solves the callback problem
    - Promises can be chained (you can have a bunch of then calls that handles the return of the value)
    - Can immediately return:
      * Promise.resolve(result)
      * Promise.reject(error)
  + Consuming promises
    - Consumers of promise are notified of the fulfillment or rejection of the promise
      * Register the callbacks to handle fulfillment and rejection with the .then() (can be chained)
      * .catch() is used for handling errors
    - Example
      * promise.then( () => {} ).catch( () => {} );