**Problem**: Callback Hell a.k.a. the Pyramid of Doom

Be careful using asynchronous callback functions in Node.js; the layers of nesting within callback functions can become a problem when they grow too deep. At which point, it is very likely to result in errors and their results getting lost in the callback.

*The following code snippet is an example of "Callback Hell":*

|  |
| --- |
| function func1(name, callback) {  // operations that takes a bit of time and then calls the callback  }  function func2(name, callback) {  // operations that takes a bit of time and then calls the callback  }  function func3(name, callback) {  // operations that takes a bit of time and then calls the callback  }  function func4(name, callback) {  // operations that takes a bit of time and then calls the callback  }  func1("input1", function(err, result1){  if(err){  // error operations  }  else {  //some operations  func2("input2", function(err, result2){  if(err){  //error operations  }  else{  //some operations  func3("input3", function(err, result3){  if(err){  //error operations  }  else{  // some operations  func4("input 4", function(err, result4){  if(err){  // error operations  }  else {  // some operations  }  });  }  });  }  });  }  }); |

**Solution**: Use flat Promise chains

Promises are a good way to write asynchronous code without getting into nested pyramids.

*The above code can be securely written as follows using a flat Promise chain:*

|  |
| --- |
| function func1(name) {  // operations that takes a bit of time and then resolves the promise  }  function func2(name) {  // operations that takes a bit of time and then resolves the promise  }  function func3(name) {  // operations that takes a bit of time and then resolves the promise  }  function func4(name) {  // operations that takes a bit of time and then resolves the promise  }  func1("input1")  .then(function (result){  return func2("input2");  })  .then(function (result){  return func3("input3");  })  .then(function (result){  return func4("input4");  })  .catch(function (error) {  // error operations  }); |