ERROR HANDLING

LIFE BEFORE DOMAINS

OCCASIONAL ERRORS

01_server_throws.js:

```
var server = require('http').createServer();
server.on('request', function(req, res) {
  a.abc();
  res.end('Thank you for using our service!');
});
server.listen(8000);
```

GLOBAL ERROR CATCHALL

Prevent node from going down

02_uncaught_exception_handler.js:

```
process.on('uncaughtException', function(err) {
  console.log('uncaught exception here', err);
});

var server = require('http').createServer();
server.on('request', function(req, res) {
  a.abc();
  res.end('Thank you for using our service!');
});
server.listen(8000);
```

EVENT EMITTERS AND ERRORS

the `error` event is special

03_event_emitter_error.js

```
var EventEmitter = require('events').EventEmitter;

var server = require('http').createServer();
server.on('request', function(req, res) {
  ee = new EventEmitter();
  ee.emit('example', 1, 2, 3);
  ee.emit('error', new Error('something terrible has happened'));
  res.end('Thank you for using our service!');
});
server.listen(8000);
```

WITHOUT DOMAINS

04_event_emitter_handle_error.js:

```
var EventEmitter = require('events').EventEmitter;

var server = require('http').createServer();
server.on('request', function(req, res) {
  ee = new EventEmitter();

  ee.on('error', function(err) {
    res.writeHead(500);
    res.end(err.message);
  });

  ee.emit('example', 1, 2, 3);
  ee.emit('error', new Error('something terrible has happened'));
  res.end('Thank you for using our service!');
});

server.listen(8000);
```

DOMAINS

Domains provide a mechanism for catching unhandled errors emitted from arbitrary groups of IO.

DOMAINS: EVENTEMITTER

05_domains.js:

```
var EventEmitter = require('events').EventEmitter;
var domain = require('domain');

var server = require('http').createServer();
server.on('request', function(req, res) {

  var d = domain.create();

  d.once('error', function(err) {
    res.writeHead(500);
    res.end(err.message);
  });

  d.run(function() {
    ee = new EventEmitter();

    ee.emit('example', 1, 2, 3);
    ee.emit('error', new Error('something terrible has happened'));
    res.end('Thank you for using our service!');
  });

});

server.listen(8000);
```

DOMAINS: EXCEPTIONS

`domain.run` also catches exceptions

06_domains_throw.js:

```
var EventEmitter = require('events').EventEmitter;
var domain = require('domain');

var a;

var server = require('http').createServer();
server.on('request', function(req, res) {

  var d = domain.create();

  d.on('error', function(err) {
    res.writeHead(500);
    res.end(err.message); // respond with the error
  });

  d.run(function() {
    a.abc();
    res.end('Thank you for using our service!');
  });

});
server.listen(8000);
```

This also works if the error is thrown asynchronously:

07_domains_throw_async.js:

```
var EventEmitter = require('events').EventEmitter;
var domain = require('domain');

var server = require('http').createServer(function(req, res) {

  var d = domain.create();

  d.on('error', function(err) {
    res.writeHead(500);
    res.end(err.message);
  });

  d.run(function() {
    setTimeout(function() {
      a.abc();
    }, 500);
  });

}).listen(8000);
```

RESOURCE LEAKS

Domains are not a cure all

08_leak_example.js:

```
var domain = require('domain');
var net = require('net')
var connections = 0;
var server = require('http').createServer(function(req, res) {
  var d = domain.create();

  d.on('error', function(err) {
    console.log('caught error, continuing');
    res.writeHead(500);
    res.end('ERROR:' + err.message + '. ' + connections + ' hanging clie
  });

  d.run(function() {
    var conn = net.connect('http://google.com');
    connections++;

    throw new Error('simulated explosion');

    conn.end("GET / \r\n\r\n");
  });
});

server.listen(8000);
```

Rule of thumb: Default to restarting the node process, unless you are absolutely sure there are no leaking resources.

EXPLICITLY ADDING EVENT EMITTERS

09_domains_add.js:

```
var EventEmitter = require('events').EventEmitter;
var domain = require('domain');

var server = require('http').createServer();
server.on('request', function(req, res) {
  var d = domain.create();
  d.add(req);
  d.add(res);

  d.on('error', function(err) {
    res.writeHead(500);
    res.end(err.message);
  });

  res.emit('error', new Error('halp!'));
});
server.listen(8000);
```

INTERCEPTING CALLBACKS

Intercepting errors in callbacks

10_domains_intercept.js:

```
var domain = require('domain');
var fs = require('fs');

var server = require('http').createServer();
server.on('request', function(req, res) {

  var d = domain.create();

  d.on('error', function(err) {
    res.writeHead(500);
    res.end(err.message);
  });

  fs.readFile('non-existant-file', d.intercept(function(contents) {
    res.end(contents);
  }));

});
server.listen(8000);
```

SUMMARY

- Use domains to handle errors
- When errors occur, seriously consider exiting the process
- add event emitters to domains: `domain.add(emitter)`
- remove event emitters from domains: `domain.remove(emitter)`
- intercept errors in callbacks: `domain.intercept(function(result) {})`
- catch errors thrown in a callback `domain.bind(function(err, result) {})`