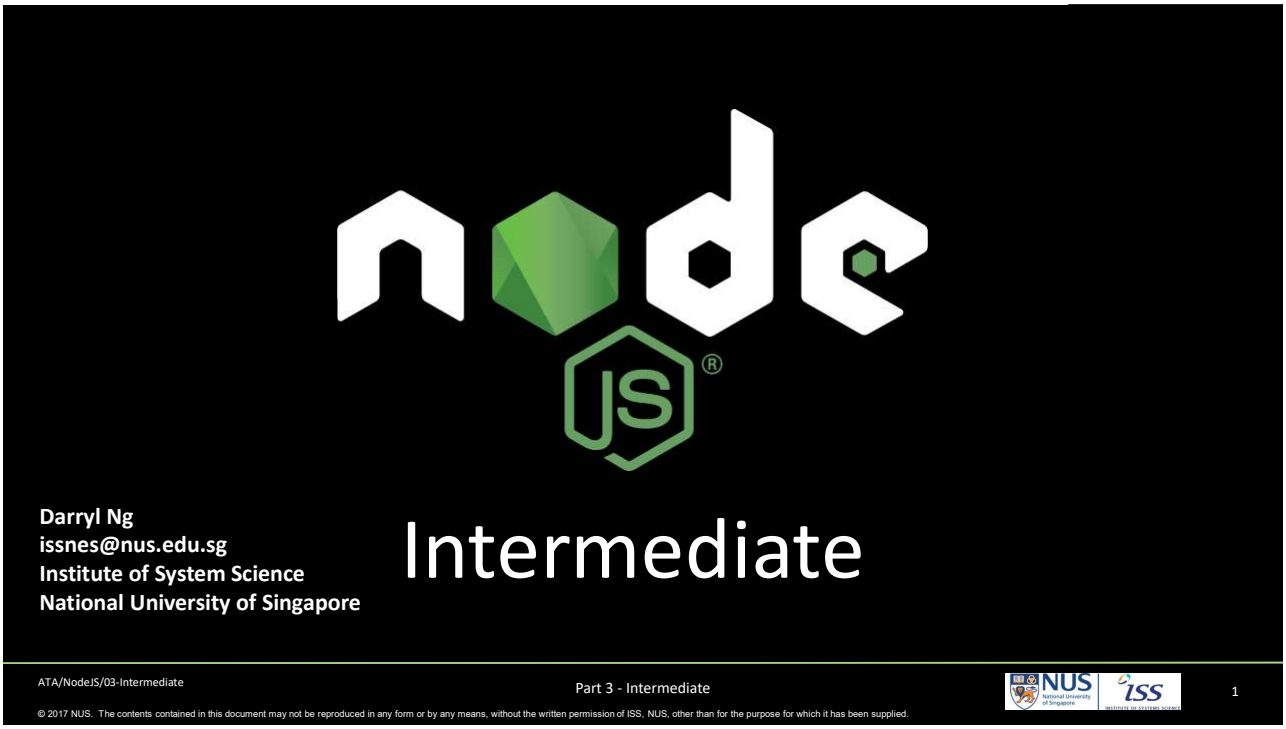
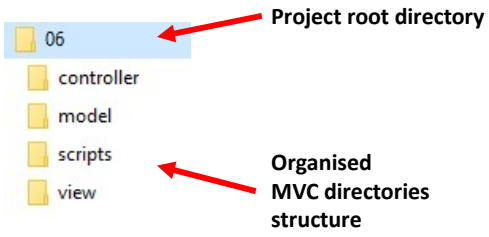


ExpressJS

- single package to handle HTML files
- wrapper class for Rendering, Routing, etc.
- **sendFile()**
  - send HTML files to browser
- Bootstrap
- jQuery
- MVC pattern and architecture
  - model
  - view
  - controller
  - scripts
  - etc.



Why ExpressJS

- **Feathers**: Build prototypes in minutes and production ready real-time apps in days.
- **ItemsAPI**: Search backend for web and mobile applications built on Express and Elasticsearch.
- **KeystoneJS**: Website and API Application Framework / CMS with an auto-generated React.js Admin UI.
- **Kraken**: Secure and scalable layer that extends Express by providing structure and convention.
- **LEAN-STACK**: The Pure JavaScript Stack.
- **LoopBack**: Highly-extensible, open-source Node.js framework for quickly creating dynamic end-to-end REST APIs.
- **MEAN**: Opinionated fullstack JavaScript framework that simplifies and accelerates web application development.
- **Sails**: MVC framework for Node.js for building practical, production-ready apps.
- **Bottr**: Framework that simplifies building chatbot applications.
- **Hydra-Express**: Hydra-Express is a light-weight library which facilitates building Node.js Microservices using ExpressJS.
- **Blueprint**: Highly-configurable MVC framework for composing production-ready services from reusable components

Package.json

```
{
  "name": "06",
  "version": "1.0.0",
  "description": "simple HTML website using nodeJS",
  "main": "htmlExample01.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "html",
    "nodeJS",
    "expressJS",
    "MVC"
  ],
  "author": "Darryl Ng",
  "license": "ISC"
}
```

Express Router

- Router
  - handlers
  - Validations
- Basic routing
- Basic middleware routing
- Accessing parameter in routing
- Handling 404 errors
- Sample routes for application

Express Router – 1
Basic Routing

```
var express = require('express');
var app = express();
var path = require('path');

//Creating Router() object
var router = express.Router();
. . .
// Provide all routes here
router.get("/", function(req, res){
        res.json({"message" : "Hello World"});
});

router.get('/home', function(req, res) {
        res.sendFile(path.join(__dirname + '/view/index.html'));
});
. . .
//Consume the routing
app.use("/", router);
```

Express Router – 2
Middleware Routing

- executed before your routes get invoked
  - log every request before its invoked
  - validate request before proceeding
  - pre-processing before processing routing request

```
// Router middleware, mentioned it before defining routes.
router.use("/", function(req,res,next) {
        console.log("/" + req.method);
        next();
});
```


Express Router – 3
Routing Parameters

- **Route :** http://example.com/api/:name/
- **Route with data :** http://example.com/api/darryl/

```
router.use("/user/:id", function(req,res,next){
        if(req.params.id == 0) {
        res.json({"message" : "You must pass ID other than 0"});
        }
        else next();
});


router.get("/user/:id", function(req, res){
        res.json({"message" : "Hello " + req.params.id});
});
```

Express Router – 3
Handling Routing Exceptions

- Handling genric error

```
app.use(function(err,req,res,next) {
  console.log(err.stack);
  res.status(500).send({"Error" : err.stack});
});
```

- Handling 404 error
- Last middleware inclusion

```
app.use("*", function(req, res){
    res.status(404).send('404');
});
```


Express Static Files

- serve images, CSS files, and JavaScript files
- pass the name of the directory that contains the static assets to the express.static middleware function
- use multiple static assets directories

```
app.use(express.static('public'))

app.use('/static', express.static('public'))

app.use(express.static(path.join(__dirname, 'public')))
```

Cookie Management - 1

- small pieces of data sent from a website and are stored in user's web browser
- **`cookie-parser`** package module
- Import and use the cookie parser

```
var cookieParser = require('cookie-parser');

var app = express();
app.use(cookieParser());
```


Cookie Management - 2

- parses Cookie header and populate **req.cookies** with an object keyed by the cookie names
- Set cookie using response object

```
app.get('/cookie',function(req, res){
  res.cookie(cookie_name , 'cookie_value').send('Cookie is set');
});
```

- Retrieve cookie using request object

```
app.get('/', function(req, res) {
  console.log("Cookies : ", req.cookies);
});
```

Cookie Management - 3

- set expiry of cookie
- expire time holds in milliseconds
  - expire property
  - maxAge property

```
res.cookie(name , 'value', {expire : new Date() + 9999});
res.cookie(name, 'value', {maxAge : 9999});
```

- delete cookie
  - clearCookie method

```
app.get('/clearcookie', function(req,res){
  clearCookie('cookie_name'); res.send('Cookie deleted');
});
```


Session Management - 1

- HTTP driven applications
- Stateless
- way to store information about the user across requests and pages
- **`node-session`** package module
- Session initialization, accessing session objects, clear session objects

Session Management – 2
Session Initialization

```
var NodeSession = require('node-session');
var express = require('express');
var parseurl = require('parseurl');

var nodeSession = new NodeSession({secret: 'Q3UBzdH9GEfiRCTKbi5MTPyChpzXLsTD'});

function session(req, res, next){
    nodeSession.startSession(req, res, next);
}

var app = express();

app.use(session);
```


Session Management – 3
Accessing session objects

- Store an item in a session

```
req.session.put('key', 'value');
```

- Retrieving an item from a session

```
var value = req.session.get('key');
```

- Retrieving all session data

```
var data = req.session.all();
```

Session Management – 4
Clear session objects

- **Removing an item from a session**

```
req.session.forget('key');
```

- **Removing all items from session**

```
req.session.flush();
```


Connecting to database - 1
Intro

- Open source database – MySQL
- Package – **mysql**
- Driver – **node-mysql**

Connecting to database – 2
Understanding database query process

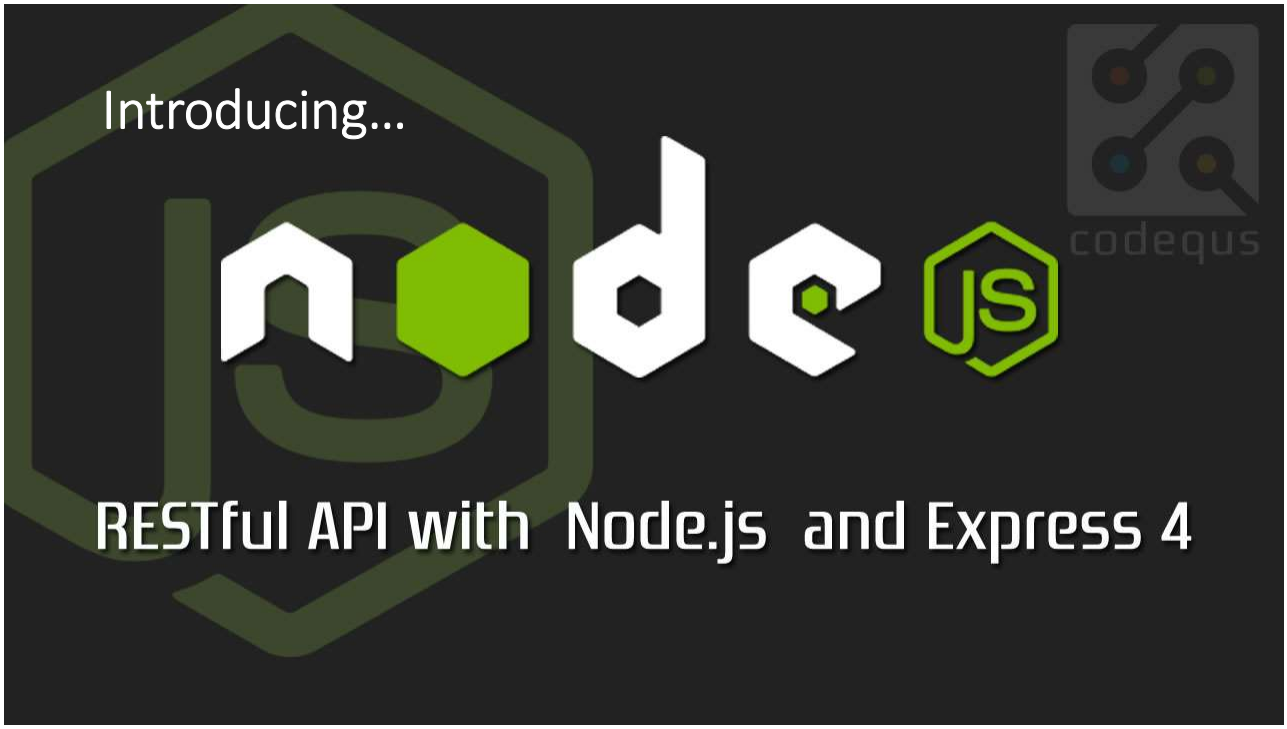
```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : '<MySQL server Machine IP or hostname>',
  user     : '<MySQL username>',
  password : '<MySQL password>',
  database : '<MySQL database name>'
});

connection.connect();

connection.query('<SQL query string>', function(err, rows, fields) {
  if (!err)
    console.log('The solution is: ', rows);
  else
    console.log('Error while performing Query:', err);
});

connection.end();
```


Connecting to database – 3
express it out (1)

```
var express    = require('express');
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : 'darryl1975',
  database : 'test'
});

var app = express();

connection.connect(function(err){
  if(!err) {
    console.log("Database is connected ... nn");
  } else {
    console.log("Error connecting database ... nn");
  }
});
```

Connecting to database – 4 express it out (2)

```
app.get("/", function(req, res){
  connection.query('SELECT * from member', function(err, rows, fields) {
    connection.end();
    if (!err) {
      console.log('The solution is: ', rows);
          res.json({"message" : + rows });
    }
    else
      console.log('Error while performing Query.');
  });
});

app.listen(1337);
```

Express 4 - 1

- express-generator
  - quick start project generator
- bundled middleware removed

Express 4 - 2
Skeleton code

```
var express = require('express');
var app = express();
var router = express.Router();

router.route('/post/:id')
        .all(function (req, res, next) {
                // runs each time
                // fetch post by id from database
        })
        .get(function (req, res, next) {
                // render post
        })
        .put(function(req, res, next) {
                // update post
        })
        .post(function(req, res, next) {
                // create new post
        })
        .del(function(req, res, next) {
                // remove post
        });
```