Cheat sheet: Node.js and Express.js

The following are objects and methods that you will need to know for MIS3502. In many cases the notes and examples have been simplified. For comprehensive documentation, please see: https://expressjs.com/en/4x/api.html and https://nodejs.org/en/docs/

Object and/or method	Variable Name (by convention)	Notes
The express object.	express	Top-level object. When the express() method is called it returns an express application object. You need this application object to make use of all other Express.js features.
The body-parser object	bodyParser	The body-parser object contains a variety of methods that return functions. (You read that right – it is a *method* that returns a *function*!) The purpose of these functions is to provide instructions about how to encode the data coming into the function. So, if the incoming data is going to be URL encoded, you should use bodyParser.urlencoded(). If the incoming data is expected to be JSON, you could use bodyParser.json(). In our class, the incoming data will always be URL encoded, and the outbound data of the API will always be JSON. Remember: URL Encoded in, JSON out.
bodyParser.urlencoded({extended:false})	Not Applicable	The URL encoded method of the bodyParser object returns a function with instructions on how to manage incoming URL encoded data. It must be provided with a parameter of either {extended:false} or {extended:true}. In our class, we will always use {extended:false}. The "extended" option has to do with the expected complexity of the incoming URL encoded data. Will it be multi-dimensional (extended) or not? The instructions include calls to the next() function so that code execution does not terminate prematurely.

Object and/or method	Variable Name (by convention)	Notes
The express app object	app	The app object is the most important object provided by the Express framework. It has a number of useful methods: • app.use • app.get • app.post • app.delete • app.put • app.listen

The use method of the app object	app.use()	The Express application will invoke the callback function without regard to the path or method (POST or GET) of the incoming request.
		
		Basic syntax:
		
		`app.use(function(req,res,next){`
		`      //code goes here`
		`});`

The GET method of the app object	app.get()	Trap and manage an HTTP GET event.
		
		The get() method takes a path as an argument, and also a callback function. The callback function provides a request and a response object.
		
		The path can be '/' which would mean no path.
		
		The path can be '/xyz would trap an HTTP POST targeted at xyz. The value 'xyz' is virtual. It does not correspond to a folder on the servier's file system.
		
		Basic syntax:
		
		`app.get('/xyz',function(req,res){`
		`      //code goes here`
		`});`

The POST method of the app object	app.post()	Trap and manage an HTTP POST event. Similar to app.get.
The DELETE method of the app object	app.delete()	Trap and manage an HTTP DELETE event. Similar to app.get.

Object and/or method	Variable Name (by convention)	Notes
The PUT method of the app object	app.put()	Trap and manage an HTTP PUT event. Similar to app.get.

The listen method of the app object	app.listen()	This method defines the port that the express app object will listen on. The callback function executes when the app has been started. The method returns an object that summarizes the properties of the server.
		
		`//here XXXX is the port number`
		`app.listen(XXXX,function(){`
		`});`

The express request object	req	All of the app callback functions provide a request object. The variable "req" is short for "request". It represents the data sent to the API endpoint. The Express framework (which includes body-parser) improves and simplifies the request object provided by Node.js alone.
		
		The request object will have query and body child objects, which in turn have properties that correspond to the data sent via GET and POST.
		
		For example:
		`req.query.x      // refers to a query`
		`                 // string parameter x`
		
		`req.body.y       // refers to a form tag`
		`                 // with the name of y`

The express response object	res	All of the app callback functions provide a response object. The variable "res" is short for "response". It represents the data sent from the API endpoint. The Express framework (which includes body-parser) improves and simplifies the response object provided by Node.js alone. The response object will have the following methods: • header() • writeHead() • write() • end()
The header method of the response object	res.header()	Sets the response's HTTP header field to value. Multiple header key-value pairs can be written. This method is really just an alias to res.set().

Object and/or method	Variable Name (by convention)	Notes
The writeHead method of the response object	res.writeHead()	Write the HTTP status code (e.g. 200 = success) and any accompanying HTTP response headers. This method must only be called once on a message and it must be called before response.end() is called.
The write method of the response object	res.write()	Write a stream of text to the HTTP response.
The end method of the response object	res.end()	End the HTTP response.
The express next object	next()	The app callback functions may also provide a next object. When the next() method is called the next matching express app event will be evaluated.

GOMY CODE ©