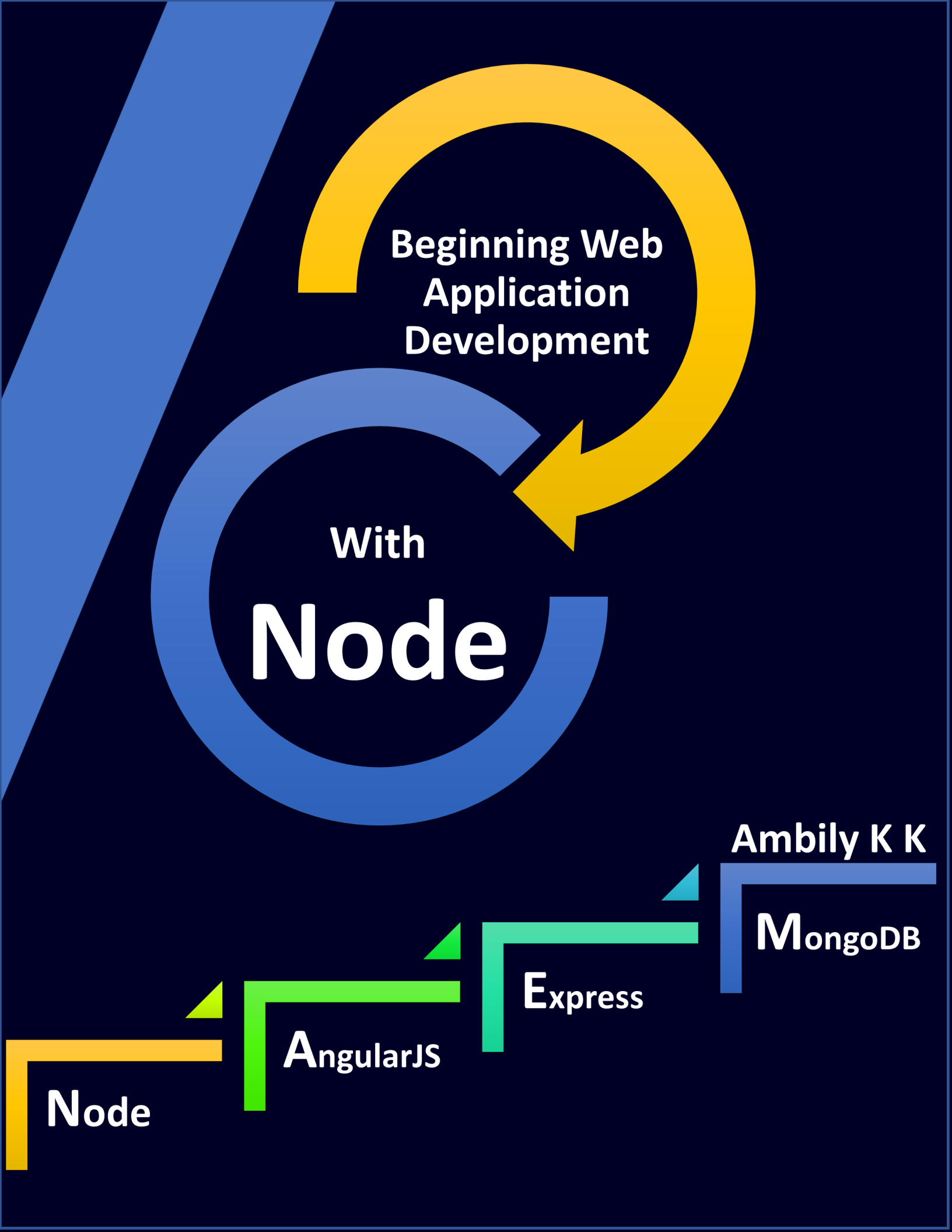

Beginning Web Application Development
With
Node
Ambily K K
MongoDB
Express
AngularJS
Node

Beginning Web Application Development with Node

Ambily K K

This book is for sale at http://leanpub.com/webappwithnode

This version was published on 2015-12-30

Contents

Exports Patterns

Let us discuss some of the export patterns used in node. Common export patterns observed are:

- Exports a Namespace
- Exports a Function
- Exports a Constructor

Exports a Namespace

Module.exports is used to define the end point for accessing the module functionality. In exports a namespace, the module returns object consists of properties and functions. Module invoking the dependent module can access the properties from the returned object and invoke the associated functions.

For example, consider the following dependency module defining the circle properties and functions. Circle module define a property to hold the value of pi and two functions for calculating the area and circumference of the circle.

circle.js

```
module.exports={
pi: 3.14,
area: function (r){
return this.pi * r * r;
},
circumference: function (r){
return 2 * this.pi * r;
}
};
```

app.js

```
var circle = require("./circle");
console.log(circle);

console.log(circle.pi);
console.log(circle.area(5));
console.log(circle.circumference(5));
```

Main module receives an object of the circle module along with the specified properties and functions.

Result

Exercises

f) Define a module using exports a namespace to find the area of a square. Invoke the module from main module and display the area of the square?

Exports a Function

Exports a function instead of a namespace. This is similar to the factory pattern observed in .NET. In this pattern, assign a function directly to exports.

For example, let us modify the previous example of circle module to exports a function.

Here, instead of defining the exports as an object, we will assign a function to the exports.

circle.js

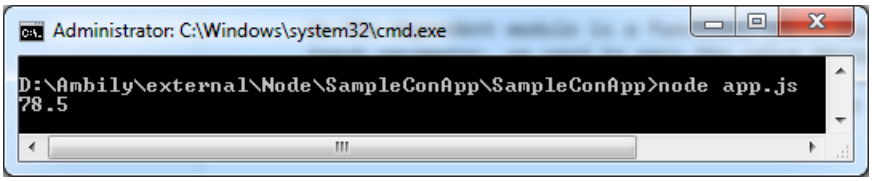
```
module.exports=function(radius){
var pi=3.14;

function findArea(){
console.log(pi * radius * radius);
}
return {area: findArea};
};
```

app.js

```
var circle = require("./circle")(5);
circle.area();
```

As the dependent module is a function, which expects an input parameter, we need to pass the value through the require statement. Passing value through require is a different pattern from whatever samples we have discussed as of now.

Result



Exercises

g) Rewrite the exercise f) using the exports a function pattern.

Exports a Constructor

Exports a constructor will return a class constructor, which can be extended using proto types. Prototype will be used to define the functions associated with the class. Main module needs to create and instance using the **new** keyword. We can create multiple instances at a time.

Let us rewrite our circle module with the exports a constructor pattern. Define the JavaScript class circle with class level properties. Extend the class with functions using prototype. At the end assign the class to the exports object.

circle.js

```
1  function circle(){
2  this.pi = 3.14;
3  }
4
5  circle.prototype.area =
6  function area(radius) {
7  console.log(" Area :  " +this.pi * radius * radius);
8  }
9
10 module.exports = circle;
```

app.js

```
var Circle = require("./circle");

var circle1 = new Circle();
var circle2 = new Circle();

circle1.area(10);
circle1.area(15);
```

Require will return the class object, which can be used for creating multiple instances. Invoke the methods using the instances created using the new keyword.

Class object names follow the naming convention to capitalize each word in the name. Objects will use the lowercase for the first word.

If you don't need instancing, then use the exports a namespace pattern. If you need instancing, then use the exports a constructor pattern.