Chapter 02. Node 간단하게 살펴보기

* 전역 객체
* Global Object는 함수 안과 밖에서 모두 사용할 수 있도록 범위를 제한하지 않는 객체로 코드 어느 부분에서도 사용할 수 있다. E.g. console, process, exports
* JSON
* JSON 포맷은 javascript에서 객체를 만들 때 사용하는 형식과 동일함. { } 를 이용해 객체를 생성하며, key와 value로 구성된 각 속성은 콤마로 구분한다.  
  e.g. {“name” : “저스트뮤직”, “age” : 20}
* javascript에서의 함수와 메소드
* 공통적으로 객체지향 언어에서 함수(Function)를 공통으로 사용하는데, javascript의 경우 객체(Object) 내에 정의된 함수를 메소드(Method)라고 별도로 지칭한다. 다만, 함수와 메소드를 엄격히 구분하지는 않는다.

[NodeExample1\ch02\_test1.js]

* Javascript의 ‘객체’ 생성
* 객체는 { } 중괄호를 통해 생성하며, 콤마( , )로 구분되는 ‘속성’이 포함된다.
* 속성 이름(name)과 속성 값(value)는 콜론( : )으로 구분한다.
* var Person = {name:”아이즈원”, age:20}

[NodeExample1\ch02\_test1.js]

* Process 객체 간단히 살펴보기
* argv 🡪 프로세스를 실행할 때 전달되는 파라미터 정보
* env 🡪 환경 변수 정보
* exit() 🡪 프로세스를 끝내는 변수

[NodeExample1\ch02\_test2.js]

* 위 파일에서 process.argv를 출력하면 node.exe와 ch02\_test2.js 파일의 위치 출력
* 세 번째 파라미터 환경변수를 보여주고, forEach로 각각 표현 가능(파일 참고)
* process.env를 통해 OS 환경변수에 접근 가능
* Node에서 모듈 사용하기
* Exports 전역 객체를 사용해 모듈 로드
* var module1 = require(‘module1’);  
  module1.함수명( );
* [module1.js]  
  exports.함수명 = 함수 정의;
* Exports와 module.exports 차이
* Exports : 속성을 추가 할 수 있어서 여러 개의 변수나 함수를 각각의 속성으로 추가 가능  
  e.g.) exports.add = function(a,b){ return a+b; }  
   exports.multiply = function(a,b) { return a\*b; }
* Module.exports : 하나의 변수, 함수, 객체를 직접 할당. 주로 객체를 할당하며, 할당한 객체 안에 넣어 둔 변수나 함수를 사용할 수도 있다.

e.g.) var calc = { };  
 calc.add = function(a,b) { return a+b; }  
 calc.multiply = function(a,b) { return a\*b; }

module.exports = calc;

[NodeExample1\ch02\_test4.js , calc.js, ,calc2.js, ch02\_test5.js]