02장 노드 간단하게 살펴보기

02-1 첫 번째 노드 프로젝트 만들기

02-2 콘솔에 로그 뿌리기

|  |  |
| --- | --- |
| 자바스크립트의 전역 객체 자료형 | |
| **전역 객체 이름** | **설명** |
| **console** | 콘솔 창에 결과를 보여주는 객체 |
| **process** | 프로세스의 실행에 대한 정보를 다루는 객체 |
| **exports** | 모듈을 다루는 객체 |

|  |  |
| --- | --- |
| console 객체의 주요 메소드 | |
| **메소드 이름** | **설명** |
| **dir ( object )** | 자바스크립트 객체의 속성들을 출력합니다 |
| **time( id )** | 실행 시간을 측정하기 위한 시작 시간을 기록합니다 |
| **timeEnd( id )** | 실행 시간을 측정하기 위한 끝 시간을 기록합니다 |

|  |  |
| --- | --- |
| 실행한 파일 정보를 알려주는 전역 변수 | |
| **변수 이름** | **설명** |
| **\_\_filename** | 실행한 파일의 이름을 출력합니다. 파일의 전체 패스가 출력됩니다  파일의 전체 패스가 출력됩니다 |
| **\_\_dirname** | 실행한 파일이 들어 있는 폴더를 출력합니다.  풀더의 전체 패스가 출력됩니다. |

02-3 프로세스 객체 간단하게 살펴보기

|  |  |
| --- | --- |
| process 객체의 주요 속성과 메소드 | |
| **속성 / 메소드 이름** | **설명** |
| **argv** | 프로세스를 실행할 때 전달되는 파라미터(매개변수) 정보 |
| **env** | 환경 변수 정보 |
| **exit()** | 프로세스를 끝내는 메소드 |

배열.forEach ( function ( item, index ) { //함수에 대한 정의}) ;

02-4 노드에서 모듈 사용하기

exports 전역 객체를 사용하여 나누어진 모듈 파일을 불러올수 있다~

exports vs module.exports

npm 패키지를 사용하면 다른 사람이 만들어 올려놓은 패키지를 다운로드하여 설치 가능

npm(Node Package Manager) 노드의 패키지를 사용할 수 있도록 설치 및 삭제 등을 지원하는 프로그램

npm init -> name 속성에만 node -> 나머지는 enter = json package 설치 완료

02-5 간단한 내장 모듈 사용하기

내장모듈 : 미리 포함되어 있는 모듈 / 바로 사용 가능

외장모듈 : 개발자가 직접 만들어 올린 모듈 / npm으로 설치 해야함

내장모듈의 대표적인 2가지 os모듈, path모듈

|  |  |
| --- | --- |
| 시스템 정보를 알려주는 os모듈의 주요 메소드 ex)var os = require(‘os’) / os.hostname() | |
| **메소드 이름** | **설명** |
| **hostname()** | 운영체제의 호스트 이름을 알려줍니다 |
| **totalmem()** | 시스템의 전체 메모리 용량을 알려줍니다 |
| **freemem()** | 시스템에서 사용 가능한 메모리 용량을 알려줍니다 |
| **cpus()** | CPU정보를 알려줍니다 |
| **networkIntefaces()** | 네트워크 인터페이스 정보를 담은 배열 객체를 반환합니다. |

|  |  |
| --- | --- |
| 파일 패스를 다루는 path 모듈 var path = require(‘path’) / path.join() | |
| **메소드 이름** | **설명** |
| **join()** | 여러 개의 이름들을 모두 합쳐 하나의 파일 패스로 만들어 줍니다.  파일 패스를 완성할 때 구분자 등을 알아서 조정합니다 |
| **dirname()** | 파일 패스에서 디렉터리 이름을 반환합니다 |
| **basename()** | 파일 패스에서 파일의 확장자를 제외한 이름을 반환합니다 |
| **extname()** | 파일 패스에서 파일의 확장자를 반환합니다. |

03장 노드의 자바스크립트와 친해지기

03-1 자바스크립트의 객체와 함수 이해하기

C언어와 같은 타입 기반의 언어(Type based language)와는 달리 자바스크립트는 타입을 명시하지 않고 사용(ex var)

|  |  |
| --- | --- |
| 자바스크립트의 자료형 | |
| **자료형** | **설명** |
| **Boolean** | [기본 자료형]true와 false의 두가지 값을 가지는 자료형 |
| **Number** | [기본 자료형] 64비트 형식의 IEEE 754값이며 정수나 부동소수 값을 가지는 자료형  몇 가지 상징적인 값을 가질수 있음 : Nan(숫자가 아님), +무한대(Number.MAX\_VALUE로 확인),  -무한대(Number.MIN\_VALUE로 확인) |
| **String** | [기본 자료형] 문자열 값을 가지는 자료형 |
| undefined | 값을 할당하지 않은 변수의 값 |
| null | 존재하지 않은 값을 가리키는 값 |
| **Object** | 객체를 값으로 가지는 자료형  객체는 속성들을 담고 있는 가방(Collection)으로 볼 수 있으며, 대표적인 객체로 Array나 Date를 들 수 있음 |

타입을 확인할 수 있도록 typeof 연산자를 제공 / 문자열을 숫자로 변환할 수 있또록 parseInt(), parseFloat() 함수도 제공

var add = function(a,b) { }; 익명함수

03-2 배열 이해하기

forEach 사용법

ex)

배열명.forEach ( function ( item, index ) {

console.log ( ‘배열 요소 #’ + index + ‘ : %s ‘, item.name);

});

|  |  |
| --- | --- |
| 배열 객체의 대표적인 속성과 메소드 | |
| **속성 / 메소드 이름** | **설명** |
| **push ( object )** | 배열이 끝에 요소를 추가합니다 |
| **pop ()** | 배열의 끝에 있는 요소를 삭제합니다 |
| **unshift()** | 배열의 앞에 요소를 추가합니다 |
| **shift()** | 배열의 앞에 있는 요소를 삭제합니다 |
| **splice ( index, removeCount, [object] )** | 여러 개의 객체를 요소로 추가하거나 삭제합니다 |
| **slice ( index, copyCount)** | 여러 개의 요소를 잘라내어 새로운 배열 객체로 만듭니다.  이함수의 경우 첫번쨰가 시작 인덱스고 2번쨰가 아마 미만인거같다  즉 1,3, 이면 인덱스 1이상 3미만의 배열을 자른다는 의미 |

03-3 콜백 함수 이해하기

자바스크립트의 변수에는 숫자나 문자여 과 같은 데이터, 그리고 중괄호를 이용해 만든 객체뿐 아니라 함수도 할당할 수 있다.

동기식 vs 비동기식

비동기식 처리를 함

콜백함수에 대한 조금더 깊은 이해가 필요하다. 기회가 있을때마다 복습하는 것이 좋을 듯 하다

callback 함수는 해당 함수의 연산이 다 끝나고 종료되기 직전 call되어서 실행됨 즉

기본적인 형태 : 기존 함수 실행 -> 콜백함수 실행 -> 콜백함수 종료 -> 기존함수 종료

클로져 : 함수 안에서 새로운 함수를 만들어 반환하는 경우에는 예외적으로 변수 접근을 허용한다 이를 클로져라고 한다.

03-4 프로토타입 객체 만들기

함수에 여러가지 기능과 속성이 추가되면서 객체 지향(Object oriented)언어에서 객체의 원형(Prototype)인 클래스를 만들고~

Person.walk = function(){ } 🡨-> Person.prototype.walk = function() { }

04장 노드의 기본 기능 알아보기

04-1 주소 문자열과 요청 파라미터 다루기

url 모듈을 사용하면 일분 주소 문자열을 URL 객체로 만들거나 또는 URL 객체에서 일반 문자열로 변환하는 일이 쉬워집니다.

|  |  |
| --- | --- |
| URL 모듈의 주요 메소드 | |
| **메소드 이름** | **설명** |
| **parse()** | 주소 문자열을 파싱하여 URL 객체를 만들어 줍니다. |
| **formant()** | URL 객체를 주소 문자열로 변환합니다. |

query 속성은 요청 파라미터 정보를 가지고 있는데 이 query 속성에 들어 있는 문자열을 다시 각각의 요청 파라미터로 분리해야 한다.

요청 파라미터는 &기호로 구분되는데 querystring 모듈을 사용하면 요청파라미터를 쉽게 분리할 수 있습니다.

|  |  |
| --- | --- |
| querystring 모듈의 주요 메소드 | |
| **메소드 이름** | **설명** |
| **parse()** | 주소 문자열을 파싱하여 URL 객체를 만들어 줍니다. |
| **stringify()** | URL 객체를 주소 문자열로 변환합니다. |

04-2 이벤트 이해하기

노드는 대부분 이벤트를 기반으로 하는 비동기 방식으로 처리합니다. 그리고 비동기 방식으로 처리하기 위해 서로 이벤트를 전달합니다.

EventEmitter를 사용해 이벤트를 주고받음

|  |  |
| --- | --- |
| 이벤트를 처리하는 EvenEmitter의 주요 메소드 | |
| **메소드 이름** | **설명** |
| **on ( event, listener )** | 지정한 이벤트의 리스너를 추가합니다 |
| **once ( event, listener )** | 지정한 이벤트의 리스너를 추가하지만 한 번 실행한 후에는 자동으로 리스너가 제거됩니다 |
| **removeListener ( event, listener )** | 지정한 이벤트에 대한 리스너를 제거합니다. |

on(), emit() 메소드에 대해서 좀더 잘 알아보도록 하자

04-3 파일 다루기

노드의 파일 시스템은 파일을 다루는 기능과 디렉터리를 다루는 기능으로 구성되어 있으며, 동기식 IO와 비동기식 IO 기능을 함께 제공합니다.

동기식 IO 와 비동기식IO를 구별하기 위해 동기식 IO메소드는 Synce라는 단어를 붙입니다.

|  |  |
| --- | --- |
| fs 모듈 – 파일 읽고 쓰기 | |
| **메소드 이름** | **설명** |
| **readFile ( filename, [encoding], [callback(err, data)]] )** | 비동기식 IO로 파일을 읽어 들입니다  여기서 callback 함수의 data는 읽어들인 파일의 내용을 의미 |
| **readFileSync ( filename, [encoding] )** | 동기식 IO로 파일을 읽어 들입니다 |
| **writeFile ( filename, data, encoding=’utf8’, [callback]** | 비동기식 IO로 파일을 씁니다 |
| **writeFileSynce ( filename, data, encoding = ‘utf8’ )** | 동기식 IO로 파일을 씁니다. |
| **fs.** |  |

대부분은 비동기 방식으로 파일을 읽고 쓰기 때문에 비동기 방식의 메소드를 주로 사용합니다,

파일을 직접 열고 닫으면서 읽거나 쓰고 싶다면 다음 메소드들을 사용할 수 있습니다.

|  |  |
| --- | --- |
| fs 모듈 - 파일을 직접 열고 닫으면서 일부분만 일거나 쓰기 위한 메소드 | |
| **메소드 이름** | **설명** |
| **open ( path, flags, [mode], [callback] )** | 파일을 엽니다 |
| **read ( fd, buffer, offset, length, position, [callback] )** | 지정한 파일의 파일 내용을 읽어 들입니다 |
| **write ( fd, buffer, offset, length, position, [callback] )** | 파일의 지정한 부분에 데이터를 씁니다 |
| **close ( fd, [callback] )** | 파일을 닫아 줍니다 |

함수 호출의 순서 open -> write -> close

|  |  |
| --- | --- |
| fs 모듈 - fs.open에서 flag에 주는 값 종류 | |
| **플래그** | **설명** |
| **‘r’** | 읽기에 사용되는 플래그입니다. 파일이 없으면 예외가 발생합니다 |
| **‘w’** | 쓰기에 사용되는 플래그 입니다. 파일이 없으면 만들어지고 파일이 있으면 이전 내용을 모두 삭제합니다 |
| **‘w+’** | 읽기와 쓰기에 모두 사용하는 플래그 입니다. 파일이 없으면 만들어지고 파일이 있으면 이전 내용을 모두 삭제합니다 |
| **‘a+’** | 읽기와 추가에 모두 사용되는 플래그입니다. 파일이 없으면 만들어지고 있으면 이전 내용에 새로운 내용을 추가합니다. |

파일을 읽거나 쓸 때는 데이터 단위가 아닌 스트림 단위로 처리할 수도 있습니다. 스트림은 데이터가 전달되는 통로와 같은 개념!

|  |  |
| --- | --- |
| fs 모듈 - 파일을 스트림 단위로 처리하는 대표적인 메소드 | |
| **메소드 이름** | **설명** |
| **createReadStream ( path, [option] )** | 파일을 쓰기 위한 스트림 객체를 만듭니다, path는 파일이름 , opetion에는 { flags : ‘주고싶은 프래그’} |
| **createWriteStream ( path, [option] )** | 파일을 쓰기 위한 스트림 객체를 만듭니다. |

http 모듈로 요청받은 파일 내용을 읽고 응답하기 부터 진행~

05장 간단한 웹 서버 만들기

05-1 간단한 웹 서버 만들기

|  |  |
| --- | --- |
| 서버 객체(server) 에서 사용할 수 있는 대표적인 메소드 | |
| **메소드 이름** | **설명** |
| **listen( port, [hostname], [backlog], [callback])** | 서버를 실행하여 대기시킵니다 |
| **on( 이벤트이름, [callback] )** | 이벤트를 처리할 때 사용되는 가장 기본적인 메소드 해당 이벤트를 실행한다 |
| **close([callback])** | 서버를 종료 시킵니다 |

클라이언트가 웹 서버에 요청할 때 발생하는 이벤트 처리하기

|  |  |
| --- | --- |
| 서버 객체(Server)의 주요 이벤트 | |
| **이벤트 이름** | **설명** |
| **connection** | 클라이언트가 접속하여 연결이 만들어질 때 발생하는 이벤트입니다 |
| **request** | 클라이언트가 요청할 때 발생하는 이벤트입니다. |
| **close** | 서버를 종료할 때 발생하는 이벤트입니다 |

|  |  |
| --- | --- |
| 응답객체(res)의 주요 메소드 | |
| **메소드 이름** | **설명** |
| **writeHead( statusCode, [statusMessage], [header] )** | 응답으로 보낼 헤더를 만듭니다 |
| **write( chunck, [encoding], [callback] )** | 응답 본문(body) 데이터를 만듭니다, 여러 번 호출될 수 있습니다 |
| **end ( [data], [encoding], [callback]** | 클라이언트로 응답을 전송합니다. 파라미터에 데이터가 들어 있다면 이 데이터를 포함시켜 응답을 전송합니다. 클라이언트의 요청이 있을 때 한번은 호출되어야 응답을 보내며, 콜백 함수가 지정되면 응답이 전송된 후 콜백 함수가 호출됩니다. |

클라이언트에서 요청이 있을 때 파일 읽어 응답하기

|  |  |
| --- | --- |
| Content-Type에 설정할 수 있는 대표적인 MIME Type | |
| **Content Type 값** | **설명** |
| **text/plain** | 일반 텍스트 문서 |
| **text/html** | HTML 문서 |
| **text/css** | CSS 문서 |
| **text/xml** | XML 문서 |
| **image/jpeg, image/png** | JPEG 파일, PNG 파일 |
| **video/mpeg, audio/mp3** | MPEG 비디오 파일, MP3 음악 파일 |
| **application/zip** | ZIP 압축파일 |

파일을 스트림으로 읽어 응답보내기

파일을 버퍼에 담아 두고 일부분만 읽어 응답 보내기]

|  |  |
| --- | --- |
| express() 메소드를 호출하여 만든 app 객체의 주요 메소드 | |
| 메소드 이름 | 설명 |
| **set(name, value)** | 서버 설정을 위한 속성을 지정한다 set() 메소드로 지정한 속성은 get() 메소드로 꺼내어 확인할 수 있다 |
| **get(name)** | 서버 설정을 위해 지정한 속성을 꺼내 온다 |
| **use( [path], function, [function] )** | 미들웨어 함수를 사용한다 |
| **get( [path], function )** | 특정 패스로 요청된 정보를 처리한다. |

|  |  |
| --- | --- |
| 서버 설정을 위해 미리 정해진 app 객체의 주요 속성 | |
| 속성이름 | 설명 |
| **env** | 서버 모드를 설정합니다 |
| **views** | 뷰들이 들어 있는 폴더 또는 폴더 배열을 설정합니다 |
| **view engine** | 기본값으로 사용할 뷰 엔진을 설정합니다. |

|  |  |
| --- | --- |
| 익스프레스에서 추가로 사용할 수 있는 응답 객체(res)의 메소드 | |
| 메소드 이름 | 설명 |
| **send([body])** | 클라이언트에 응답 데이터를 보냅니다. 전달할 수 있는 데이터는 HTML문자열, Buffer객체, JSON 객체, JSON배열 |
| **status(code)** | HTTP상태 코드를 반환합니다. 상태 코드는 end()나 send()와 같은 전송 메소드를추가로호출해야 전송할수 있습니다. |
| **sendStatus(statusCode)** | HTTP상태 코드를 반환합니다. 상태 코드는 상태 메시지와 함께 전송됩니다. |
| **redirect( [status], path )** | 웹 페이지 경로를 강제로 이동시킵니다. |
| **render (view, [local]. [callback] )** | 뷰 엔진을 사용해 문서를 만든 후 전송합니다. |

|  |  |
| --- | --- |
| 익스프레스에서 추가로 사용할 수 있는 요청 객체(req)의 속성들 | |
| 추가한 정보 | 설명 |
| **query** | 클라이언트에서 GET 방식으로 전송한 요청 파라미터를 확인합니다.  예) req,query,name |
| **body** | 클라이언트에서 POST방식으로 전송한 요청 파라미터를 확인합니다.  단 body-parser와 같은 외장 모듈을 사용해야 합니다  예)req.body.name |
| **header( name )** | 헤더를 확인합니다. |

|  |  |
| --- | --- |
| 라우터 미들웨어를 사용하면 app객체에 추가되는 메소드 | |
| **메소드 이름** | **설명** |
| **get(callback)** | GET 방식으로 특정 패스 요청이 발생했을 때 사용할 콜백 함수를 지정합니다 |
| **post(callback)** | POST 방식으로 특정 패스 요청이 발생했을 때 사용할 콜백 함수를 지정합니다 |
| **put(callback)** | PUT 방식으로 특정 패스 요청이 발생했을 때 사용할 콜백 함수를 지정합니다 |
| **delete(callback)** | DELETE 방식으로 특정 패스 요청이 발생했을 때 사용할 콜백 함수를 지정합니다 |
| **all(callback)** | 모든 요청 방식을 처리하며, 특정 패스 요청이 발생했을 때 사용할 콜백 함수를 지정합니다 |

5-5 쿠키와 세션 관리하기

쿠키 : 클라이언트 웹 브라우저에 저장되는 정보

세션 : 웹서버에 저장되는 정보

cookie-parser 미들웨어 : 쿠키를 설정하거나 확인 / use 메소드를 사용해 cookie-parser미들웨어를 사용하도록 만들면 요청 객체에 cookies 속성이 추가됨

express-session 모듈

06 데이터 베이스 사용하기

06-1 몽고디비 시작하기

|  |  |
| --- | --- |
| mongoose 모듈에서 사용할 수 있는 대표적인 메소드 | |
| **메소드 이름** | **설명** |
| **connect ( uri(s), [option], [callback] )** | mongoose를 사용해 데이터베이스를 연결합니다. 연결 후에는 moongoose coonection 객체를 사용해 연결 관련 이벤트를 처리할 수 있습니다 |
| **Schema()** | 스키마를 정의하는 생성자 입니다. |
| **model( name, [schema], [collection], [skiplnit])** | 모델을 정의합니다  [collection]이 지정되면 이 컬렉션을 사용하며, 지정하지 않으면 name으로 유추한 컬렉션을 사용합니다 |