Node js

리터럴 – 데이터를 표현하는 방식

Var a = 1 – 정수를 표현하는 방식

Var a = “1” – 문자를 표현하는 방식.

javascript에서 ‘내용’ + 변수 +’내용 \n’ 등의 표현은 번거롭기 때문에

template literal 을 사용한다

~칸의 ` 을 사용

`내용 ${변수}

내용‘ 으로 같은 결과를 얻는다. ${1+1}등도 가능

http – 서버와의 통신 방법, 프로토콜

opentutorials.org – 인터넷에 연결되어 있는 컴퓨터, host,domain

3000 – 포트

Main – 경로

?id=HTML&page=12 – 웹서버에 전달할 수 있는 정보 ?로 시작하며 값과 값은 &로 연결 값의 이름과 값은 =로 연결



Fs, http 같은건 모듈임 비슷한기능끼리 묶어둔 것.

Require(‘url’) - nodejs에게 url이라는 모듈을 사용할 것임을 말하는 것.

Fs = 파일 시스템

fs.readFile('/etc/passwd', (err, data) => {

if (err) throw err;

console.log(data);

});

최신기능?

fs.readFile('sample.txt', ‘utf8’ function(err, data) {

console.log(data);

});

구버전 같은 효과

이렇게 함으로서 다른 파일의 내용을 불러올 수 있게 된다.

Nodejs 파일은 수정될 때 마다 재시작 해야 하지만.

Nodejs 에서 불러온 파일이 수정된 경우 새로고침만 해도 바로 적용된다.

콘솔에서의 입력

Var args = process.argv;

args라는 배열에 콘솔에서 입력값을 문자로 받아온다.

콘솔에서 node name.js 123 이라고 입력하면

process.argv 가 ‘123’을 입력으로 받아

args[2]에 저장한다.

Response.writeHead(200) – 전송이 성공적이다.

Response.writeHead(404) – 파일을 찾을 수 없다.

Var \_url=request.url;

Url.parse(\_url,ture)는 url로부터 여러가지 정보를 가져온다.

Var pathname= Url.parse(\_url,ture).pathname;

Pathname==’/’라면 가장 최상위의 루트를 의미하며

Pathname은 / 이후의 쿼리 스트링은 포함하지 않는다.

따라서 쿼리스트링을 통해 페이지를 나누는 경우 pathname이 루트인지 아닌지에 따라 유효한 페이지인지 아닌지를 구분할 수 있다.

루트의 메인 페이지의 경우 /로만 구분되었기 때문에 하위 페이지와는 구분이 안되어있다.

따라서 queryData.id 가 undifined라면 메인페이지 임을 알 수 있으므로 하위페이지의 처리와 동일하게 조건문을 통해 구분하여 메인페이지를 구성할 수 있다.

Nodejs에서 파일 목록 불러오기

Var testFolder = ‘실행 위치기준에서의 해당 폴더 경로’;

Var fs = require(‘fs’);

Fs.readdir(testFolder,function(error,filelist){

Console.log(filelist);

})

라고 하면 해당 폴더의 파일들의 목록을 배열상태로 가져온다.

Node js상의 함수

Function 함수명(){

}

호출 함수명;

Math라는 객체에 정리되어있는 반올림 함수 round

호출 Math.round(1.6)

=2

함수 입출력

Function sum(first,second){//파라미터/parameter

Return first+second;

}

Sum(2,4); //아귀먼트/argument

출력 = 6

동기 – 비효율적

비동기 – 효율적, 병렬처리

패키지 매니저 – 여러가지 추가 소프트웨어

Pm2 – nodejs가 꺼지면 자동으로 다시 켜주고, 수정사항 발생시 자동으로 껏다 켜준다.

설치

Npm install pm2 -g

실행

Pm2 start main.js

Pm2로 실행중인 프로그램들 모니터링

Pm2 monit

Pm2로 실행중인 목록

Pm2 list

종료

Pm2 stop main.js

수정시 자동 껏다킴//웹페이지 자체는 새로고침 해야함

Pm2 start main.js –watch

로그/문제사항 발생시 확인

Pm2 log

Form은 그 안의 각각의 컨트롤에 사용자가 입력한 정보를 submit버튼을 눌렀을 때

액션 속성이 가리키는 서버로 쿼리스트링 형태로 데이터를 전송한다.

이는get 방식으로 적절하지 못하다

메소드 속성을 추가하여 post 방식으로 전송해야 서버만 쿼리스트링에 추가된다.

Input text 속성으로 placeholder=”내용”

하면 입력 전에는 내용이 표시되고 입력하면 내용이 사라지고 입력한 내용만 표기된다.

Post 방식의 데이터 받기

request.on('data', function (data) {

body += data;

});

데이터가 들어올 때 마다 fuction(data) 콜백함수를 호출함 data라는 인자를 통해 데이터를 줌

상단 추가 var qs = require('querystring');

request.on('end', function () {

var post = qs.parse(body);

var title = post.title;

var description = post.description;

});

모든 수신이 끝난경우 변수에 배열 형태로 저장한 뒤 이용할 수 있게 된다.

이후 파일을 생성하여 저장하려면

Fs.writeFile(`data/${title}`,description,’utf8’,function(err){

/\*에러 처리시의 내용을 넣어야 하는데 현재는 다루지 않는다\*/

response.writeHead(200);

response.end('success');

})

를 통해 title의 제목으로 description을 내용으로 하는 파일을 만들 수 있다.

현재는 파일을 생성하는데 성공하면 success만 뜨고 다시 주소를 수정해서 되돌아가야 한다.

이때 필요한 것이 redirection이다

Response.writeHead(302,{Location:`/?id=${title}`});

Response.end();

라고 하면 302는 리다이렉션을 의미하고 location으로 자동으로 이동된다.

현 실습 예제에서 update구현을 위해 어떤 내용을 업데이트 할 지에 대해 링크에 쿼리스트링을 붙여서 전송하였다.

<a href="/update?id=${title}">update</a>

사용자가 제목을 수정하게 될 경우 기존의 ${title}을 값으로 사용하면 수정된 이후에 data폴더에 존재하지 않을 수 있기 때문에 제목이 수정되더라도 보존되는 데이터가 필요하다.

따라서

<input type=”hidden” name=”id” value=”${title}”> 를 추가하여 미리 ${title}값을 꺼낸다.

이름의 바뀐 경우에 이름을 바꾸는 방법

Fs.rename(oldpath,newpath,callback(err))

이므로

현재 예제의 경우에는 에러 처리를 따로 안하고 있으므로

fs.rename(`data/${id}`,`data/${title}`,function(error){}); 로 하면 된다.

 fs.rename(`data/${id}`,`data/${title}`,function(error){

        //위의 create에서의 기능과 같이 이름을 바꾸고 내용을 바꾼다.

        fs.writeFile(`data/${title}`, description, 'utf8', function (err) {

          response.writeHead(302, { Location: `/?id=${title}` });

          response.end();

        })

      });

이와 같이 업데이트의 내용을 중간에 삽입함으로서 이름과 내용을 바꿀 수 있게 된다.

문서 삭제기능을 a href로 하게되면 정보가 get방식으로 전송되므로 반드시 form으로 감싸서 post방식으로 할 것

<form action = "/delete\_process" method="post" onsubmit="return del\_check();">

              <input type="hidden" name="id" value=${title}>

              <input type="submit" value="delete">

</form>

예제에서는 다음과같이 hidden으로 사라질 제목을 미리 알아낸 뒤 해당 절차로 넘어가는데, onsubmit의 내용은 내가 직접 추가한 함수로 확인을 한 뒤 true, false를 반환한다.

fs.unlink(`data/${id}`, function (error) {

        response.writeHead(302, { Location: `/` });

        response.end();

      })

삭제의 경우

Fs.unlink(path,callback)으로 이루어진다.

Onsubmit 에 커스텀 함수 넣으면 동작 안함 이유 불명 일단 confirm으로 대체함.

배열이 순서가 있는 정보를 정리하는 수단이라면

객체는 순서가 없는 정보들을 정리하는 수단.

var members = ['egoing', 'k8805', 'hoya'];

console.log(members[1]); // k8805

var roles = {

'programmer':'egoing',

'designer' : 'k8805',

'manager' : 'hoya'

}

console.log(roles.designer); //k8805

객체를 이용하여 저장한 예시

중괄호를 사용함에 유의

객체는 키와 값으로 저장된다

객체를 반복문에 사용할 때 .키도 가능하지만 [키] 도 가능하다

첫번째 var 값으로는 키를 받는다.



함수 또한 이러한 방식으로 객체에 넣고 사용할 수 있다.



객체 내에서 자기 자신을 참조 하는 경우 this를 사용한다.

모듈



서로 다른 js파일에서 모듈을 생성하고 외부사용을가능하게하고(왼쪽)

같은 디렉토리 내의 외부의 모듈을 가져와서 사용하는 방법이다(오른쪽)

Mudule.exports = {

V:’v’,

F:function(){

Console.log(this.v);

}

}

처럼 exports문을 객체 이름대신 넣어도 가능하다.