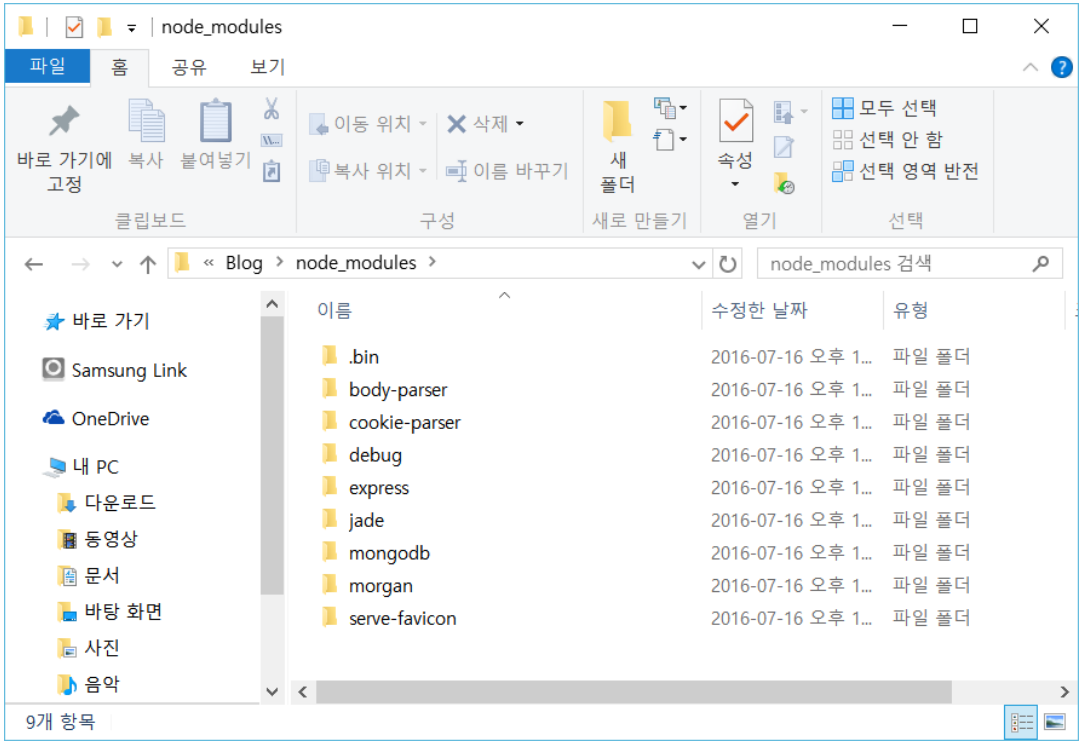
Node.js 기본 모듈

2016년도 2학기 2학년 방과후학교

Node.js 모듈 종류

- 기본 모듈 : Node.js와 함께 라이브러리 폴더에 설치돼서 별도의 설치가 필요 없다.
- 확장 모듈 : npm이라는 패키지 툴로 별도 설치 하고, 설치된 모듈은 node_modules 폴더에 설치된다.



Node.js 모듈 로딩과 객체 생성

```
1  // http 내장 모듈 로딩
2  var http = require('http');
3
4  // http.Server 객체 생성
5  var server = http.createServer(function (req, res) {
6      res.writeHead(200, {'Content-Type': 'text/html'});
7      res.end('<h1>Hello World!</h1>');
8  })
9
10 // http.Server의 listen 메소드 호출
11 server.listen(3000);
12
13 console.log('Server running at http://127.0.0.1:3000'
```

http.createServer([requestListener])

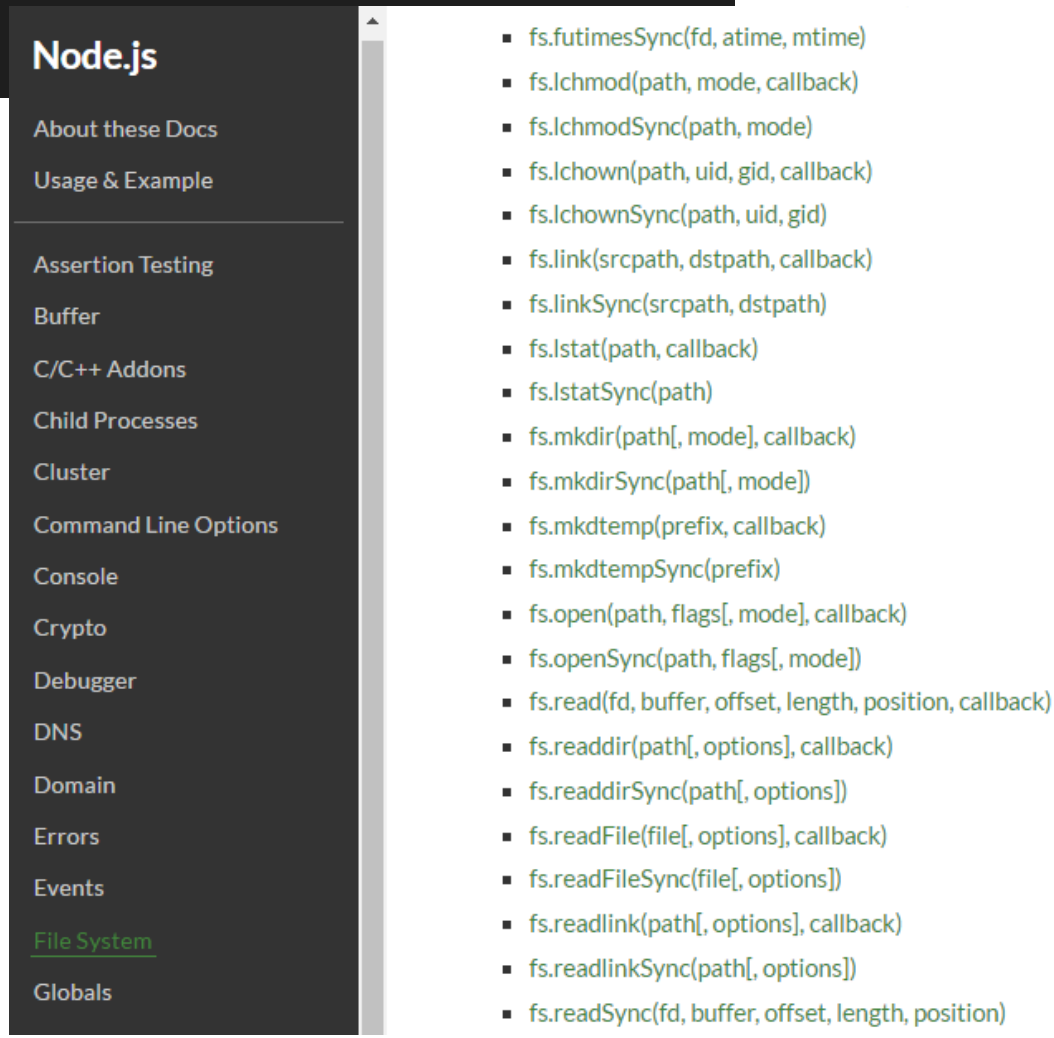
Returns a new instance of `http.Server`.

The `requestListener` is a function which is automatically added to the `'request'` event.

- Class: http.Server
  - Event: 'checkContinue'
  - Event: 'clientError'
  - Event: 'close'
  - Event: 'connect'
  - Event: 'connection'
  - Event: 'request'
  - Event: 'upgrade'
  - server.close([callback])
  - server.listen(handle[, callback])
  - server.listen(path[, callback])
  - server.listen(port[, hostname][, backlog][, callback])
  - server.maxHeadersCount
  - server.setTimeout(msecs, callback)
  - server.timeout

객체 생성 없이 모듈 사용

```
1  var fs = require('fs');
2  var content = fs.readFile('hello.txt', 'utf8', function (err, content) {
3      console.log(content);
4  });
5  console.log('Reading file...');
```



기본 모듈 - util

- var util = require('util');
- util.format('%d + %d = %d', 10, 20, 10+20);
- util.format('%s, %s', 'Hello', 'World');

Node.js 이벤트 처리

- EventEmitter 클래스 : 이벤트를 정의하고 다루는 기능을 정의한 클래스
- 이벤트를 발생하는 객체는 EventEmitter 클래스의 인스턴스이다.
- EventEmitter에는 이벤트를 발생시키고, 이벤트에 반응하는 이벤트 핸들러를 설정할 수 있는 기능을 제공한다.
- emitter.emit('이벤트명');
- emitter.on('이벤트명', 이벤트 리스너 함수);
- emitter.once('이벤트명', 이벤트 리스너 함수);

```
const EventEmitter = require('events');

class MyEmitter extends EventEmitter {}

const myEmitter = new MyEmitter();
myEmitter.on('event', () => {
  console.log('an event occurred!');
});
myEmitter.emit('event');
```

Node.js 이벤트 처리 예

- exit 이벤트는 Node.js 애플리케이션이 종료하면서 발생하는 이벤트다.
- process.on('exit', function( ) { console.log('매번 동작') };
- process.once('exit', function( ) { console.log('한번만 동작') };
- process.emit('exit'); // exit 이벤트 발생

기본 모듈 - path

- path 모듈은 파일의 경로를 다루는 기능을 제공하는 모듈이다.
- var path = require('path');
- 기존에 작성된 경로에서 경로를 구성하는 요소나 파일 이름, 확장자를 얻을 수 있고, 새로운 경로를 생성할 수 있다.
- path.basename( ) : 파일 이름
- path.dirname( ) : 파일이 포함된 경로
- path.extname( ) : 파일의 확장자

기본 모듈 - path

- path.sep // 경로 구분자
- path.parse('파일경로'); // 경로를 분석하여 객체형태로 제공
- path.join('..'); // 입력한 값들을 조합하여 경로를 생성
- path.format({ }); // 객체로 입력하여 경로를 생성

기본 모듈 - fs

- 파일 시스템을 다루기 위해서 fs 모듈을 사용한다.
- var fs = require('fs');
- 파일 생성, 읽기, 쓰기, 삭제
- 파일 접근 가능 여부, 속성
- 디렉토리 생성, 읽기, 삭제
- 파일 스트림 오픈
- fs 모듈은 동기식(이름에 Sync가 붙음)과 비동기식 API를 함께 제공한다.

기본 모듈 - fs

- 파일 접근 가능여부 확인하기
- fs.access(path [,mode], callback), fs.accessSync(path[,mode])
- 접근 mode
  - ✓ fs.F_OK : 파일 존재만 확인
  - ✓ fs.R_OK : 파일 읽기 가능 여부 확인
  - ✓ fs.W_OK : 파일 쓰기 가능 여부 확인
  - ✓ fs.X_OK : 파일 실행 가능 여부 확인

기본 모듈 - fs

- 파일 상태 얻기
- fs.stat(path, callback), fs.statSync(path)
- 결과(stats)
  - ✓ stats.isFile() : 파일 여부
  - ✓ stats.isDirectory() : 디렉토리 여부
  - ✓ stats.size : 파일 크기
  - ✓ stats.birthtime, stats.atime, stats.mtime : 생성/접근/수정일시

기본 모듈 - fs

- 파일에 write 하기 (같은 이름의 파일인 경우 덮어씀)
  - ✓ fs.write(fd, data[, position[, encoding]], callback)
  - ✓ fs.writeFile(filename, data[, options], callback)
  - ✓ fs.writeFileSync(filename, data[, options])
- 기존 파일에 추가하기 (파일이 없으면 생성)
  - ✓ fs.appendFile(file, data[, options], callback)
  - ✓ fs.appendFileSync(file, data[, options])

기본 모듈 - fs

- 파일 이름 변경/이동하기
  - ✓ fs.rename(oldPath, newPath, callback)
  - ✓ fs.renameSync(oldPath, newPath)
- 파일 삭제하기
  - ✓ fs.unlink(filename, callback)

기본 모듈 - fs

- 디렉토리 생성 (같은 이름의 디렉토리가 있으면 실패)
  - ✓ fs.mkdir(path [, mode], callback)
  - ✓ fs.mkdirSync(path, [, mode])
- 디렉토리 삭제 (디렉토리가 비어 있지 않으면 실패)
  - ✓ fs.rmdir(path, callback)
  - ✓ fs.rmdirSync(path)
- 디렉토리 내 목록 조회
  - ✓ fs.readdir(path, callback)
  - ✓ fs.readdirSync(path)

기본 모듈 - url

- URL 구성 요소
  - ✓ 프로토콜(Protocol)
  - ✓ 호스트(Host)
  - ✓ 포트번호(Port)
  - ✓ 경로(Path)
  - ✓ 쿼리(Query)
  - ✓ 프래그먼트(Fragment)

기본 모듈 - url

- var url = require('url');
- url.parse(urlStr[, parseQueryString][, slashesDenoteHost])
  - ✓ urlStr : URL 문자열
  - ✓ parseQueryString : 쿼리 문자열 파싱, 기본값 false
  - ✓ slashesDenoteHost : //로 시작하는 주소의 경우 호스트 인식 여부, 기본값 false

기본 모듈 - url

- url.format({ URL 정보 })
  - ✓ protocol : 프로토콜
  - ✓ host : 서버 호스트 주소
  - ✓ pathname : 경로
  - ✓ search : 쿼리 스트링
  - ✓ auth : 인증 정보

기본 모듈 - url

- URL에 허용되는 문자 : 알파벳, 숫자, -, _, ., ~
- 그 외의 문자는 %를 이용한 인코딩을 해야 함
- 한글이 입력된 URL
  - ✓ https://www.google.co.kr/?...newwindow=1&q=nodejs
  - ✓ https://www.google.co.kr/?...newwindow=1&q=%EB%85%B8%EB%93%9CJS

확장 모듈 - urlencode

- urlencode는 확장 모듈이므로 별도 모듈 설치가 필요함
- npm install urlencode
- var urlencode = require('urlencode');
  - ✓ console.log(urlencode('한글'));
  - ✓ console.log(urlencode.decode('%ED%95%9C%EA%B8%80'));

기본 모듈 - querystring

- querystring은 쿼리 문자열을 다루는 기본 모듈
- 쿼리 문자열 (이름=값 형태로 값을 표현)
  - ✓ URL에서 사용 (GET 방식) → url 모듈만으로도 가능
  - ✓ HTTP 메시지 바디로 정보를 전달 (POST 방식) → querystring 모듈 필요

기본 모듈 - querystring

- var querystring = require('querystring');
- 쿼리 문자열 분석하기
  - ✓ querystring.parse(str[, sep][, eq][, options])
  - ✓ 쿼리 구분자(sep)의 기본값 : '&'
  - ✓ 이름/값 구분자(eq)의 기본값 : '='

기본 모듈 - querystring

- 쿼리 문자열 만들기
  - ✓ querystring.stringify(obj [, sep][, eq][, options])
  - ✓ sep : 쿼리 구분자
  - ✓ eq : 이름/값 구분자