T아카데미 온라인 강의

Node.js 프로그래밍

01. Node.JS의 개요

CONTENTS

학습안내

학습 목표

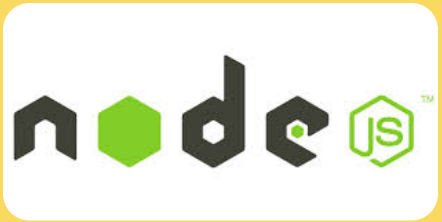
1. Node.js의 특징을 이해할 수 있습니다.
2. Node.js 프로그래밍 방법을 이해할 수 있습니다.
3. Node.js 개발 환경을 준비할 수 있습니다.
4. API 문서를 보고 코드 작성을 할 수 있습니다.

1. Node.js 소개

1. Node.js 소개

- Node.js

  - 2009년 Ryan Dahl
  - 자바 스크립트 언어
  - 크롬 V8 엔진



1. Node.js 소개

- Node.js의 특징

  - 싱글 쓰레드
  - 비동기 I/O
  - 이벤트 기반(event driven)

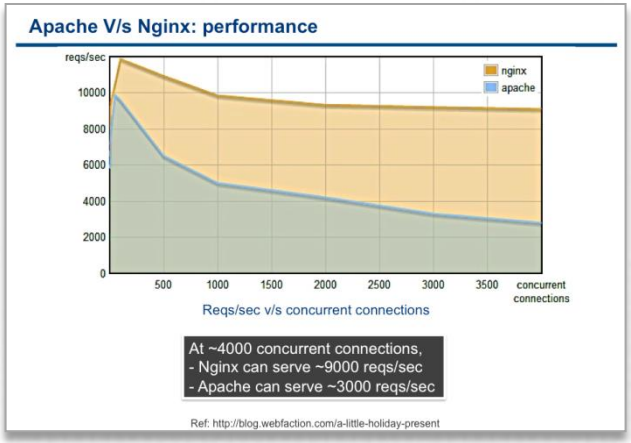
  네트워크 애플리케이션

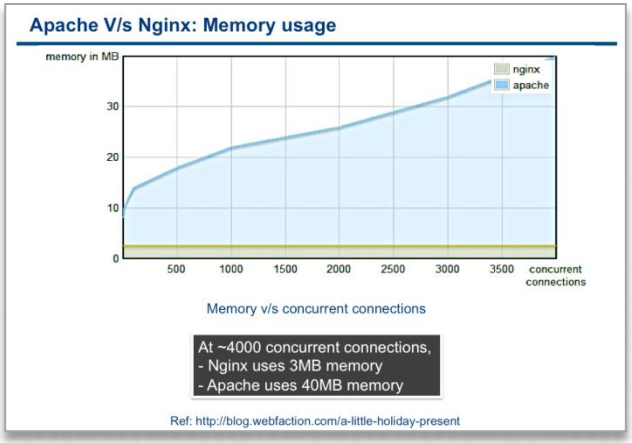
1. Node.js 소개

- 비동기 I/O
  - 시간이 걸리는 I/O
    - 하드 디스크 접근
    - 데이터베이스 서버
    - 네트워크를 이용해서 다른 서비스 접근
  - I/O 동작이 끝날 때까지 대기 : 동기식
  - I/O 동작이 끝날 때까지 대기하지 않음 : 비동기식

1. Node.js 소개

- 비동기 I/O의 장점
  - 아파치 vs Nginx





1. Node.js 소개

- Node.js의 장점

  - 싱글 쓰레드로 작성
  - 비동기 I/O
  - 간단한 구조의 경량 프레임워크와 풍부한 라이브러리
  - 서버와 클라이언트에서 사용하는 언어가 같다.

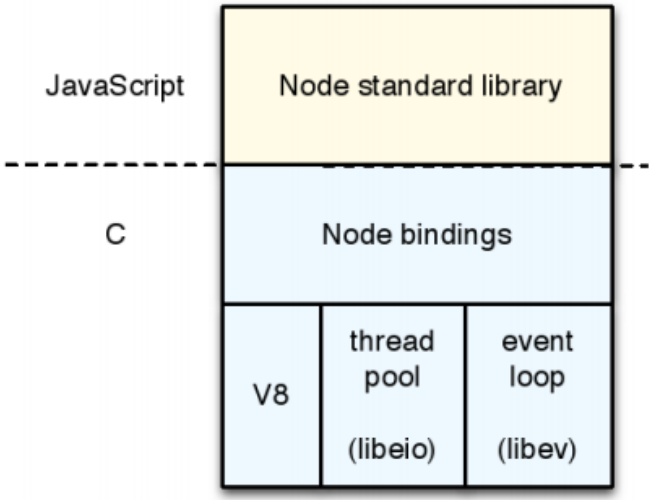
1. Node.js 소개

- Node.js 권장 분야

  - 실시간 소셜 네트워크 서비스
  - 데이터 중심의 서비스
  - IoT 기기 연동

1. Node.js 소개

- 아키텍처
  - 상위 레벨 – JavaScript
  - 로우 레벨 – C
    - 바인딩
    - v8 엔진
    - libev : Event
    - libeio : I/O

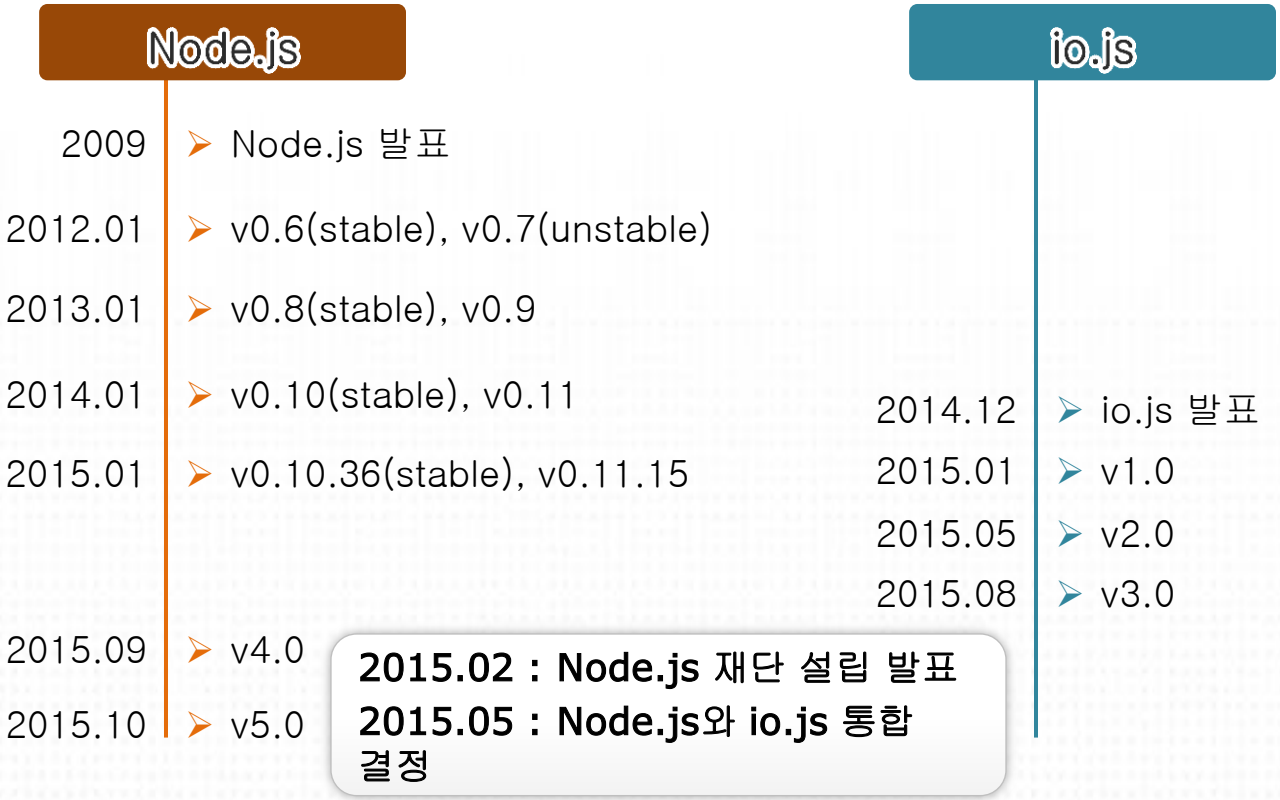


1. Node.js 소개

- Node.js 홈페이지(nodejs.org)

1. Node.js 소개

- Node.js의 간단한 역사



1. Node.js 소개

Node.js 재단

Node.js 재단

- Node.js 플랫폼과 관련된 모듈 개발 지원하는 협업 오픈 소스 프로젝트
- open gevernance model
- 기술 결정 위원회(Technical Streering Committee)

주요 멤버

- IBM, intel, Joyent, Microsoft, PayPal, redhat

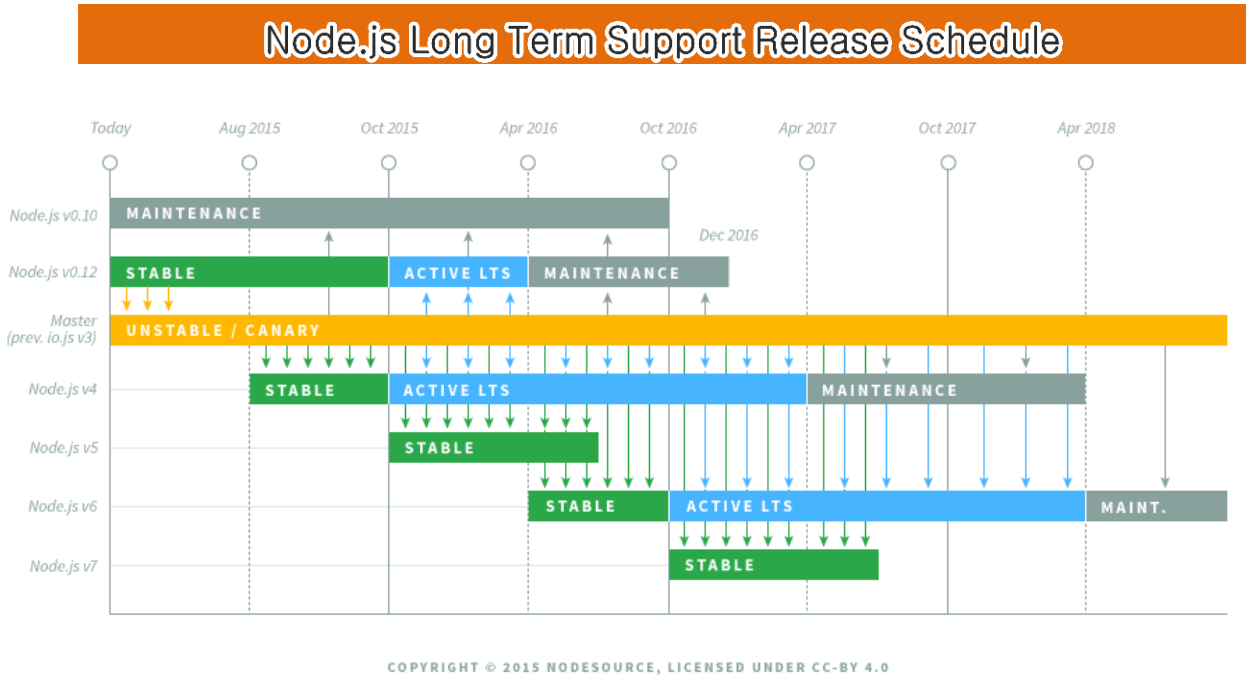
1. Node.js 소개

- 버전 구성과 지원

  - Node.js 버전을 두 단계로 진행
  - 기존 : 짝수버전(Stable) 홀수 버전(Unstable)
  - 4.x 이후 : Stable, LTS
  - LTS : 짝수 버전 Stable 6개월 이후 LTS로 전환
    - ✓LTS(Long Term Support)
    - ✓LTS : 호환성이 깨지는 변경 없음.
    - ✓LTS 18개월. 그후 Maintain 상태(12개월)
    - ✓매년 새로운 메이저 버전의 LTS 시작
  - 현재 4.2가 LTS 상태
    - ✓현재 4.2가 LTS 상태 (v4.2.0 -> v4.2.3)

1. Node.js 소개

- 릴리즈 계획



2. 프로그래밍 모델

2. 프로그래밍 모델

- 프로그래밍 모델

동기(Synchronous)

비동기(Asynchornous)

차이는 무엇일까?

2. 프로그래밍 모델

- 프로그래밍 모델

동기(Synchronous)	비동기(Asynchornous)
A 실행 – A 결과 – B 실행 – B 결과	A 실행 – B 실행 – (B 결과)– (A 결과)
실행이 끝나고 다음 실행	실행 결과가 끝날때까지 기다리지 않는다.

2. 프로그래밍 모델

- 동기식

파일 읽기

```
1  var fs = require('fs');
2  var content = fs.readFileSync("readme.txt", "utf8");
3  console.log(content);
4  console.log('Reading file...');
```

1

2

3

4

2. 프로그래밍 모델

- 비동기식

비동기식

비동기식

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

1
2
4
3

2. 프로그래밍 모델

- 동기/비동기 방식의 코드 차이점

동기식 함수 구현과 사용

동기식 함수 구현

```
function add(i, j) {
   return i + j;
}
```

동기식 함수 사용

```
var result = add(1, 2);
console.log('Result : ', result);
```

2. 프로그래밍 모델

- 동기/비동기 방식의 코드 차이점

비동기식 함수 구현과 사용

비동기식 함수 구현

```
function add(i, j, callback) {
   var result = i + j;
   callback(result);
}
```

비동기식 함수 사용

```
add(1, 2, function(result) {
   console.log('Result : ', result);
});
```

2. 프로그래밍 모델

- 비동기 방식의 API로 파일 읽는 코드 예

  - 콜백을 이용한 파일 읽기

```
fs.readFile('textfile.txt', 'utf8', function(err, text) {
    console.log('Read File Async', text);
});
```

2. 프로그래밍 모델

- 콜백 함수 형태

비동기 함수의 에러 처리

콜백 함수의 파라미터로

대부분 비동기 API

```
callbackFunc(arg1, arg2, function(error, result) {
  if ( error ) {
    // 에러 처리
    return;
  }
  // 정상 처리
}
```

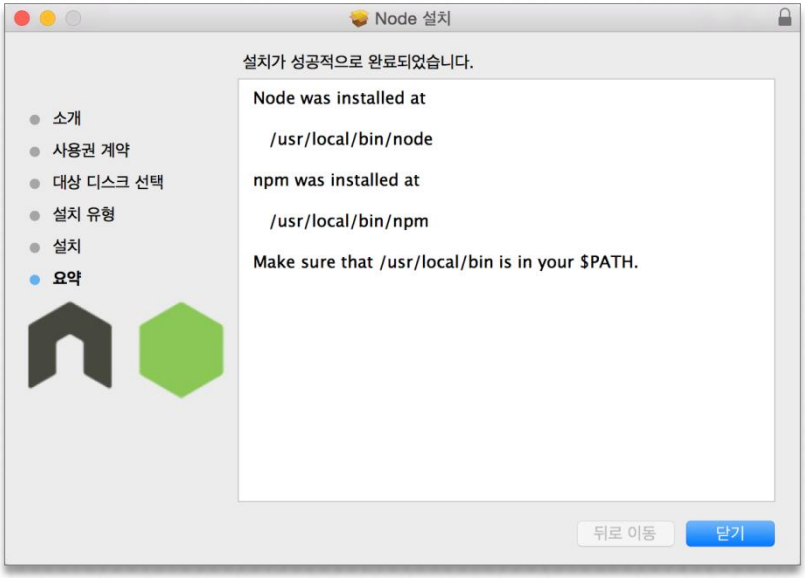
3. Node.js 개발환경

3. Node.js 개발환경

- 다운로드
  - 홈페이지(nodejs.org)
  - 플랫폼에 맞는 설치 파일 다운로드
  - 설치

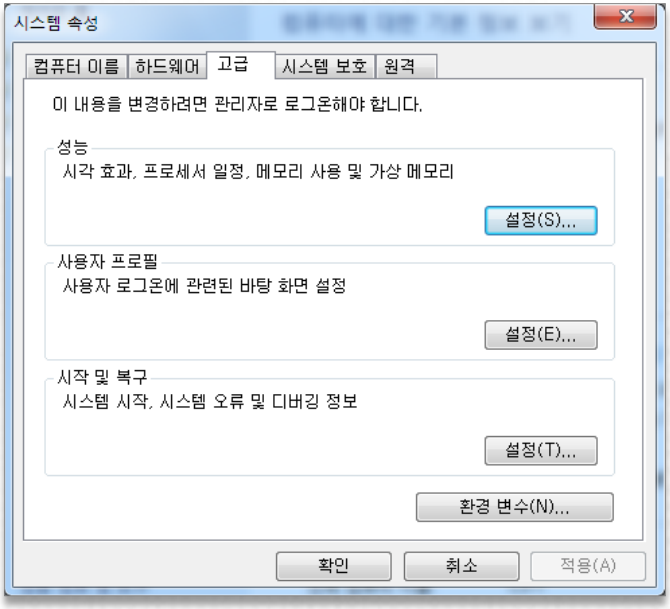
3. Node.js 개발환경

- 설치
  - 설치 완료
  - 환경 설정
    - 설치된 경로 위치



3. Node.js 개발환경

- 경로 설정
  - 자동으로 환경 설정 안되면 수동 설정
    - node 설치 폴더 위치 설정



3. Node.js 개발환경

- 설치 확인
  - 콘솔에서 node 명령 실행
  - node –v
    - v : 버전 확인 옵션

```
MacBook-Pro-2:~ wannabewize$ node -v
v5.1.1
```

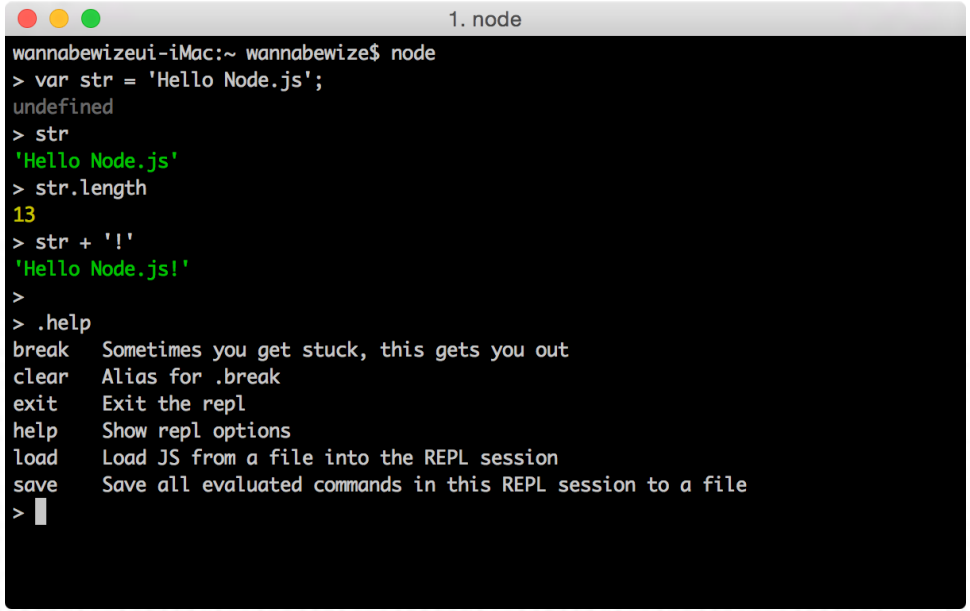
3. Node.js 개발환경

- Node.js 명령어
  - node 콘솔 명령
  - node [SOURCE.JS] [ARGS]
    - v : 버전
    - e, p : 스트립트 평가
    - c : 실행하지 않고 문법 체크
    - r : 모듈을 미리 로딩

```
MacBook-Pro-2:~ wannabewize$ node --help
Usage: node [options] [ -e script | script.js ] [arguments]
       node debug script.js [arguments]

Options:
  -v, --version         print Node.js version
  -e, --eval script     evaluate script
  -p, --print           evaluate script and print result
  -c, --check           syntax check script without executing
  -i, --interactive     always enter the REPL even if stdin
                        does not appear to be a terminal
  -r, --require         module to preload (option can be repeated)
  --no-deprecation      silence deprecation warnings
  --throw-deprecation   throw an exception anytime a deprecated function is used
  --trace-deprecation   show stack traces on deprecations
  --trace-sync-io       show stack trace when use of sync IO
                        is detected after the first tick
  --track-heap-objects  track heap object allocations for heap snapshots
  --v8-options          print v8 command line options
  --tls-cipher-list=val use an alternative default TLS cipher list
  --icu-data-dir=dir    set ICU data load path to dir
                        (overrides NODE_ICU_DATA)

Environment variables:
NODE_PATH               ':'-separated list of directories
                        prefixed to the module search path.
NODE_DISABLE_COLORS     set to 1 to disable colors in the REPL
NODE_ICU_DATA           data path for ICU (Intl object) data
NODE_REPL_HISTORY       path to the persistent REPL history file

Documentation can be found at https://nodejs.org/
```

3. Node.js 개발환경

- REPL
  - 콘솔 기반의 실행 환경

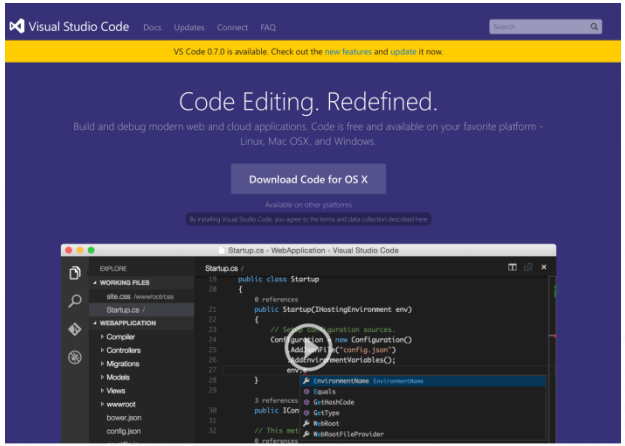


3. Node.js 개발환경

- 개발도구
  - IDE : Eclipse, WebStorm
  - Editor : Visual Studio Code, Sublime, …

3. Node.js 개발환경

- 편집기
  - 비주얼 스튜디오 코드 (무료!)
    - https://code.visualstudio.com
    - 설치
  - 개발툴 환경 설정
    - 사이트에서 Node 설정 보기
    - 콘솔에서 code



4. Hello World

4. Hello World

- 코드 작성 : helloWorld.js
  - console.log(‘Hello World!');

- 실행
  - node helloWorld.js

4. Hello World

- 서버 코드
  - helloWorld2.js

```
var http = require('http');
http.createServer( function(request, response) {
        response.writeHead( 200, {'Content-Type':
'text/html'});
        response.end('<h1>Hello World!</h1>');
}).listen(3000);
```

  - 웹 브라우저로 확인

127.0.0.1:3000

5. 도큐먼트

5. 도큐먼트

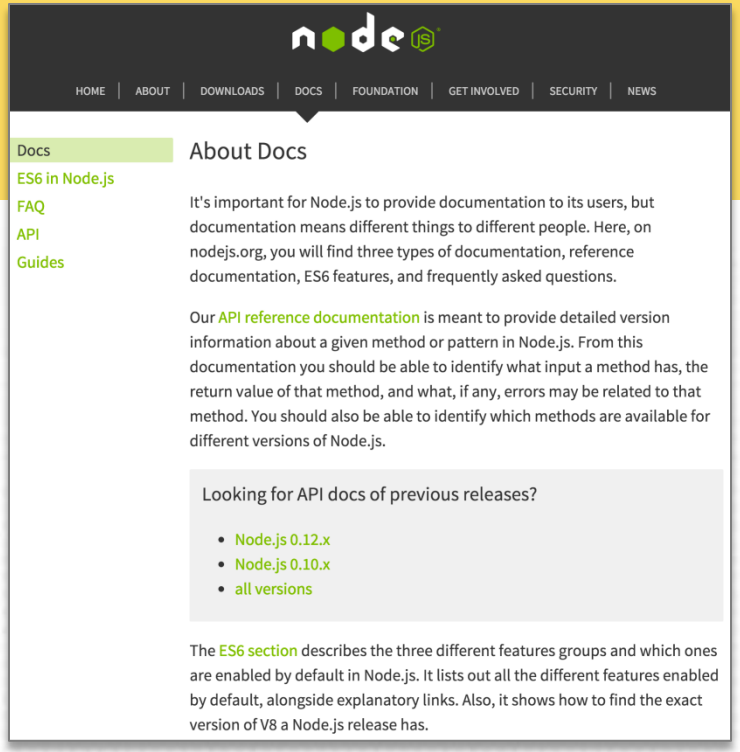
- 모듈

  - Node.js 간단한 구조
  - 필요한 모듈을 로딩
  - 모듈 : 다른 언어의 라이브러리에 해당

5. 도큐먼트

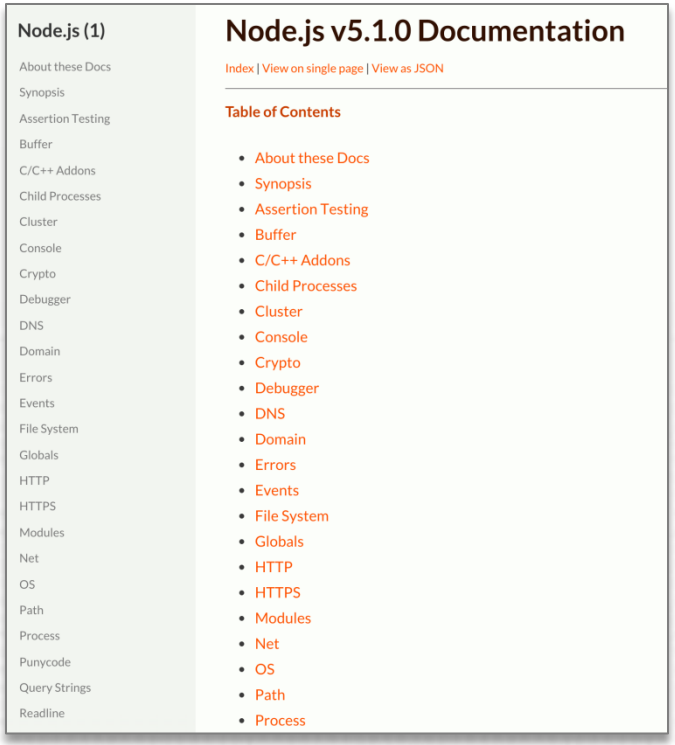
도큐먼트 보기

- Node.js 사이트
- DOCS
- API



5. 도큐먼트

- API 문서



API 문서 보기

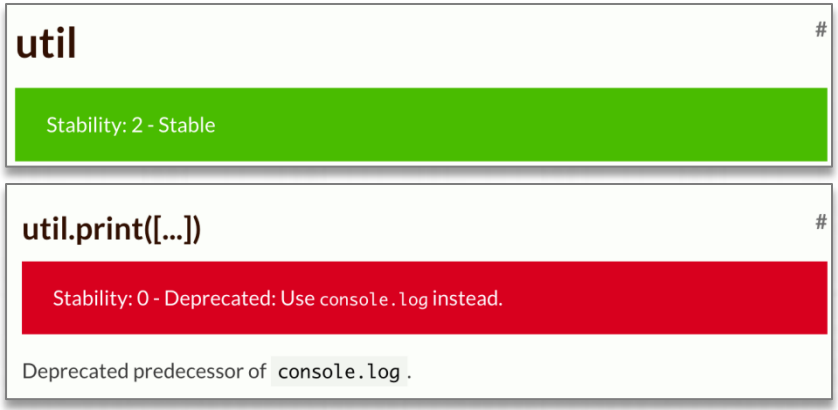
5. 도큐먼트

API 문서 보기

API 안정도

Stability

- 0 : Deprecated
- 1 : Experimental
- 2 : Stable
- 3 : Locked



5. 도큐먼트

모듈 사용하기

모듈 문서 보는 법을 알아보자!

모듈 Readline

- 클래스 : Interface
- 메소드
- 이벤트
- 모듈 함수

- Readline
  - Class: Interface
    - rl.close()
    - rl.pause()
    - rl.prompt([preserveCursor])
    - rl.question(query, callback)
    - rl.resume()
    - rl.setPrompt(prompt)
    - rl.write(data[, key])
  - Events
    - Event: 'close'
    - Event: 'line'
    - Event: 'pause'
    - Event: 'resume'
    - Event: 'SIGCONT'
    - Event: 'SIGINT'
    - Event: 'SIGTSTP'
  - Example: Tiny CLI
  - readline.clearLine(stream, dir)
  - readline.clearScreenDown(stream)
  - readline.createInterface(options)
  - readline.cursorTo(stream, x, y)
  - readline.moveCursor(stream, dx, dy)

5. 도큐먼트

모듈 로딩

모듈 로딩

- require(‘모듈 이름’)
- 절대 경로 혹은 상대 경로
  var readline = require('readline');

모듈 종류

- 기본 모듈 : 미리 컴파일된 상태로 라이브러리 디렉토리 – 설치 불 필요
- 확장 모듈 : npm으로 별도 설치

5. 도큐먼트

- 모듈 로딩 위치
  - 기본 모듈 로딩 위치
    - Node.js 라이브러리 디렉토리
  - 확장 모듈
    - 같은 폴더
    - node_modules 이름의 폴더
    - 상위 폴더의 node_modules

5. 도큐먼트

- 모듈 로딩 – 객체 생성
  - 클래스
    - Interface
  - 모듈 로딩과 객체 생성

```
var readline = require('readline');

var rl = readline.createInterface();
```

- Readline
  - readline.createInterface(options)
  - Class: Interface
    - rl.setPrompt(prompt)
    - rl.prompt([preserveCursor])
    - rl.question(query, callback)
    - rl.pause()
    - rl.resume()
    - rl.close()
    - rl.write(data[, key])

5. 도큐먼트

객체 생성

객체 생성 함수 옵션

```
var rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
```

readline.createInterface(options)

Creates a readline `Interface` instance. Accepts an "options" Object that takes the following values:

- `input` - the readable stream to listen to (Required).
- `output` - the writable stream to write readline data to (Optional).
- `completer` - an optional function that is used for Tab autocompletion. See below for an example of using this.
- `terminal` - pass `true` if the `input` and `output` streams should be treated like a TTY, and have ANSI/VT100 escape codes written to it. Defaults to checking `isTTY` on the `output` stream upon instantiation.

5. 도큐먼트

메소드 사용

모듈 로딩, 객체 생성 후 메소드 사용

```
var readline = require('readline');
var rl = readline.createInterface();
rl.setPrompt('>>');
```

- **Readline**
  - **readline.createInterface(options)**
  - **Class: Interface**
    - **rl.setPrompt(prompt)**
    - **rl.prompt([preserveCursor])**
    - **rl.question(query, callback)**
    - **rl.pause()**
    - **rl.resume()**
    - **rl.close()**
    - **rl.write(data[, key])**

5. 도큐먼트

- 이벤트
  - 이벤트 – 이벤트 핸들러
  - .on([이벤트 이름], [리스너 함수])

```
rl.on('line', function (cmd) {
  console.log('You just typed: '+cmd);
});
```

  - 리스너 함수의 파라미터
    - API 문서 참조

- Events
  - Event: 'line'
  - Event: 'pause'
  - Event: 'resume'
  - Event: 'close'
  - Event: 'SIGINT'
  - Event: 'SIGTSTP'
  - Event: 'SIGCONT'

Event: 'line'

```
function (line) {}
```

5. 도큐먼트

- 모듈 함수
  - 객체 생성 없이 모듈에 직접 사용

```
var readline = require('readline');

readline.cursorTo(process.stdout, 60, 30);
```

  - **readline.cursorTo(stream, x, y)**
  - **readline.moveCursor(stream, dx, dy)**
  - **readline.clearLine(stream, dir)**
  - **readline.clearScreenDown(stream)**

학습정리

학습정리

- 지금까지 'Node.JS의 개요'에 대해 살펴보았습니다.

Node.js 란

비동기 방식으로 자바 스크립트 언어를 이용해서 네트워크 애플리케이션 플랫폼 제작에 적합한 프레임워크

Node.js의 프로그래밍 모델

비동기 방식으로 작성하고, 콜백을 이용하는 방식으로 코드를 작성합니다.

Node.js 환경

Node.js를 다운로드해서 설치하고 개발 환경을 준비했습니다.

학습정리

- 지금까지 'Node.JS의 개요'에 대해 살펴보았습니다.

Hello World
Node.js 애플리케이션을 작성하고 실행해봤습니다.

도큐먼트 보기
API 문서를 보고 모듈을 로딩하고 객체를 생성해봤습니다.
그리고 함수 실행하는 방법을 알아봤습니다.