

서버시스템구축실습

Lecture #04 자바스크립트 기초

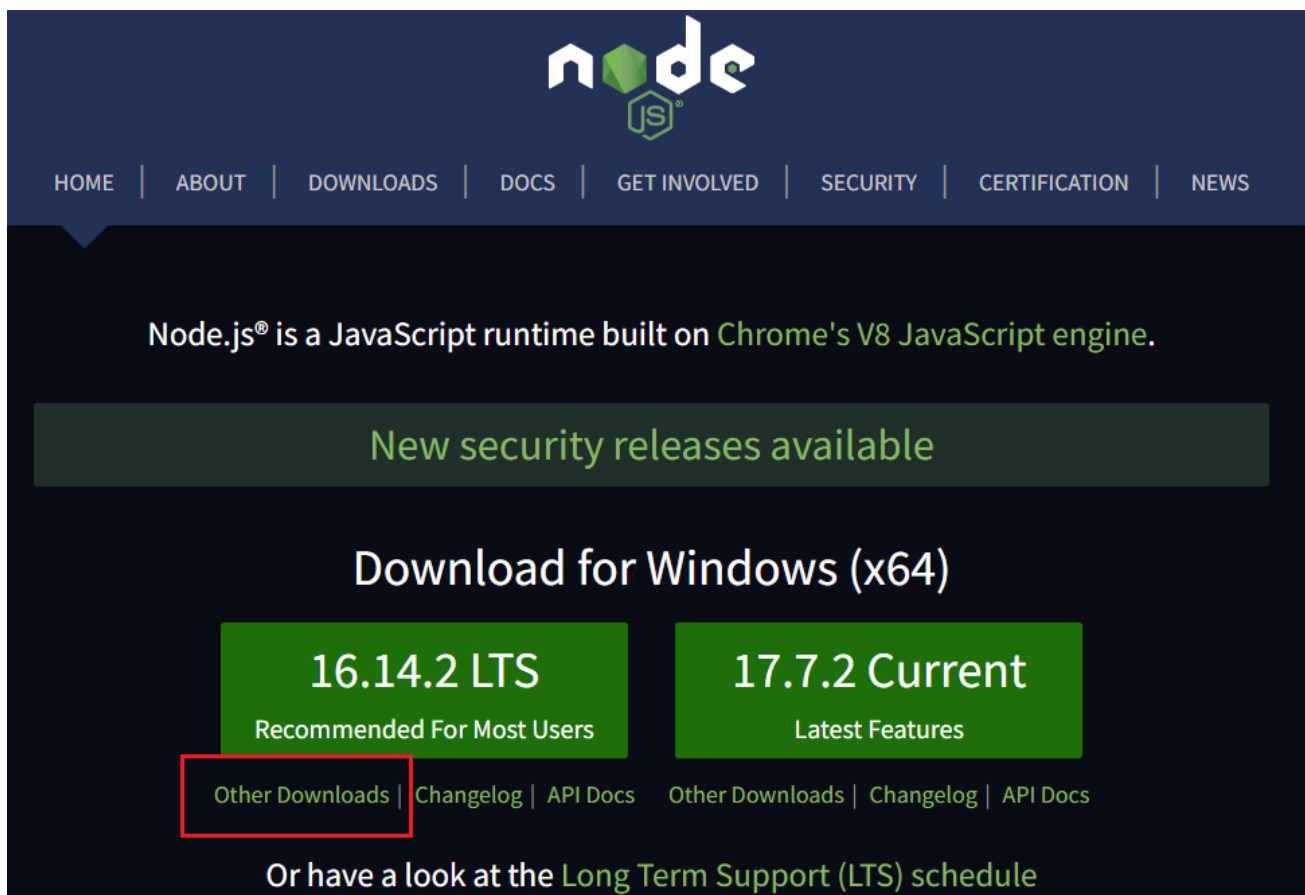
컴퓨터공학과
박지웅

무단 복제 및 배포를 금지합니다.

Node.js 설치

■ Node.js 런타임 설치

- <https://nodejs.org/en/>



The screenshot shows the Node.js website homepage. At the top is the Node.js logo and a navigation bar with links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. Below the navigation bar, a text block states: "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." A green banner below this reads "New security releases available". The main section is titled "Download for Windows (x64)" and contains two green buttons: "16.14.2 LTS Recommended For Most Users" and "17.7.2 Current Latest Features". Below these buttons, a row of links is shown: "Other Downloads | Changelog | API Docs" (under the LTS button) and "Other Downloads | Changelog | API Docs" (under the Current button). The "Other Downloads" link under the LTS button is highlighted with a red rectangle. At the bottom, a text block says: "Or have a look at the Long Term Support (LTS) schedule".

node JS

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

New security releases available

Download for Windows (x64)

16.14.2 LTS
Recommended For Most Users

17.7.2 Current
Latest Features

Other Downloads | Changelog | API Docs

Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule

Node.js 설치

■ Node.js 런타임 설치

- LTS 버전 다운로드 링크 복사

Downloads

Latest LTS Version: **16.14.2** (includes npm 8.5.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Recommended For Most Users

Current
Latest Features

Windows Installer

node-v16.14.2-x64.msi

macOS Installer

node-v16.14.2.pkg

Source Code

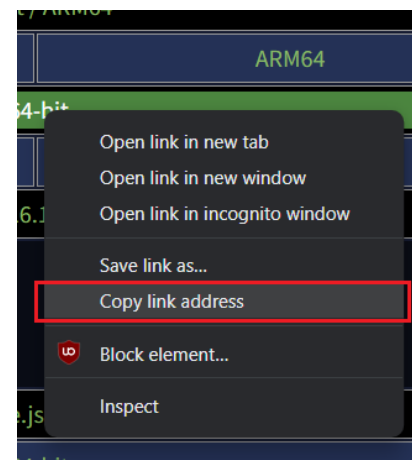
node-v16.14.2.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v16.14.2.tar.gz	

Additional Platforms

Docker Image	Official Node.js Docker Image
Linux on Power LE Systems	64-bit
Linux on System z	64-bit
AIX on Power Systems	64-bit

우클릭



Node.js 설치

■ Node.js 런타임 설치

- GCP VM 인스턴스에서 다운로드
 - wget <https://nodejs.org/dist/v16.14.2/node-v16.14.2-linux-x64.tar.xz>

```
jwpark12@ss:~$ wget https://nodejs.org/dist/v16.14.2/node-v16.14.2-linux-x64.tar.xz
--2022-03-21 05:20:19--  https://nodejs.org/dist/v16.14.2/node-v16.14.2-linux-x64.tar.xz
Resolving nodejs.org (nodejs.org)... 104.20.22.46, 104.20.23.46, 2606:4700:10::6814:172e, ...
Connecting to nodejs.org (nodejs.org)|104.20.22.46|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21941244 (21M) [application/x-xz]
Saving to: 'node-v16.14.2-linux-x64.tar.xz'

node-v16.14.2-lin 100%[=====>] 20.92M  139MB/s  in 0.2s

2022-03-21 05:20:19 (139 MB/s) - 'node-v16.14.2-linux-x64.tar.xz' saved
[21941244/21941244]

jwpark12@ss:~$
```

Node.js 설치

■ Node.js 런타임 설치

- 압축해제 및 설정
 - `tar xf node-v16.14.2-linux-x64.tar.xz`
 - `cd node-v16.14.2-linux-x64`
 - `sudo cp -Rvi ./{bin,include,lib,share} /usr/local/`

Node.js 설치

■ REPL (Read-Evaluate-Print-Loop)

- 자바스크립트를 작성하고 실시간으로 실행 및 결과 확인
 - REPL 환경에서는 Node.js의 모든 코어 모듈에 접근 가능
 - 종료를 위해서는 Ctrl+C 2번

```
jwpark12@ss:~$ node
Welcome to Node.js v16.14.2.
Type ".help" for more information.
> 3+3
6
> 3/0
Infinity
> console.log("hello universe!");
hello universe!
undefined
> let name="jwpark"
undefined
> console.log(name);
jwpark
```


Node.js 애플리케이션 실행

■ 자바스크립트 파일 실행

- 자바스크립트 파일 위치로 이동
- node 명령을 사용해 자바스크립트 파일 실행
 - 확장자는 생략 가능
 - node hello.js
 - node hello

```
jwpark12@ss:~/test$ node hello.js
Hello, Universe!
jwpark12@ss:~/test$
```


자바스크립트 - 데이터 타입

■ 숫자 데이터 타입 - Number

- 연산자(Operator)

- 단항 연산자

- `X++`, `++X`

- 이항 연산자

- `+`, `-`, `*`, `/` 등

- 삼항 연산자

- `var beverage = (age >= 21) ? "Beer" : "Juice"`

자바스크립트 - 데이터 타입

■ 숫자 데이터 타입 - Number

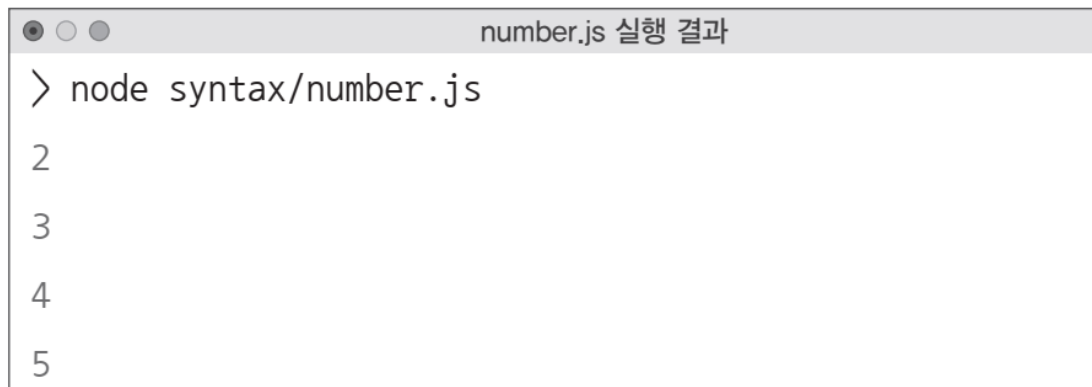
- 실습

- syntax 디렉토리 생성
 - cd test
 - mkdir syntax
 - cd syntax
- number.js 작성

예제 6-1 숫자 데이터의 표현 예 - 사칙연산

web2-nodejs/syntax/number.js

```
console.log(1 + 1);  
console.log(4 - 1);  
console.log(2 * 2);  
console.log(10 / 2);
```



```
number.js 실행 결과  
> node syntax/number.js  
2  
3  
4  
5
```

자바스크립트 - 데이터 타입

- 문자열 데이터 타입 - String
 - 문자열 표현
 - “ ” 또는 ‘ ’로 문자열 표현
 - 실습
 - string.js 파일 작성

예제 6-2 문자열 데이터의 덧셈 연산

web2-nodejs/syntax/string.js

```
console.log(1 + 1);  
console.log('1' + '1');
```

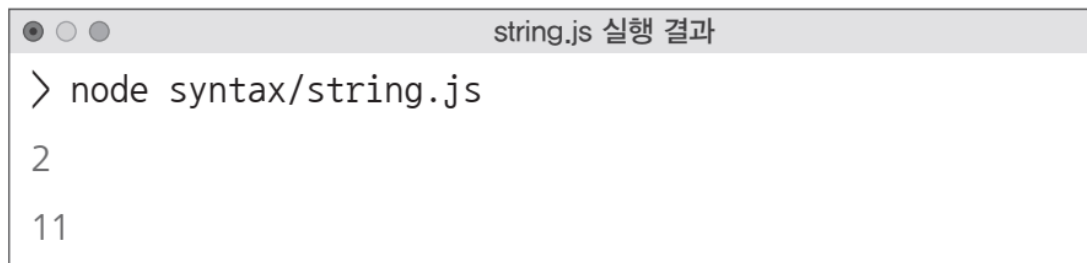
자바스크립트 - 데이터 타입

- 문자열 데이터 타입 - String
 - 문자열 표현
 - “ ” 또는 ‘ ’로 문자열 표현
 - 실습
 - string.js 파일 작성

예제 6-2 문자열 데이터의 덧셈 연산

web2-nodejs/syntax/string.js

```
console.log(1 + 1);  
console.log('1' + '1');
```



```
string.js 실행 결과  
> node syntax/string.js  
2  
11
```

자바스크립트 - 데이터 타입

■ 문자열 데이터 타입 - String

- 문자열 처리 예

- https://github.com/wikibook/nodejs/blob/master/source/01_web2-nodejs-fin/syntax/6-3_loremIpsum.txt

- Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

예제 6-3 의미 없는 문자열

web2-nodejs/syntax/loremIpsum.txt

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
```

자바스크립트 - 데이터 타입

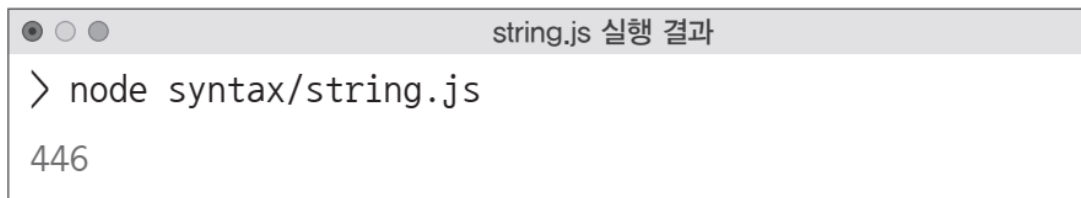
■ 문자열 데이터 타입 - String

- 문자열 처리 예
 - 문자열의 `length` 속성 활용

예제 6-4 문자열의 길이를 나타내는 `length` 속성

[web2-nodejs/syntax/string.js](#)

```
console.log('Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit  
in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat  
cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.'.length);
```



```
string.js 실행 결과  
> node syntax/string.js  
446
```

자바스크립트 - 변수

■ 변수의 형식

- 예제

- variable.js 파일 작성

예제 7-1 변수 선언 후 값 대입

web2-nodejs/syntax/variable.js

```
a = 1;      ← a 변수 선언 후 숫자 1 대입
console.log(a);
a = 2;      ← a 변수에 숫자 2 대입
console.log(a);
```

variable.js 실행 결과

```
> node syntax/variable.js
1
2
```

자바스크립트 - 변수

■ 변수의 형식

- 변수 vs. 상수
 - 변할 수 있는 수 vs. 변하지 않는 수

예제 7-2 상수의 값 변경

web2-nodejs/syntax/variable.js

... 생략 ...

1 = 2; ← 오류 발생!

```
variable.js 실행 결과
> node syntax/variable.js
ReferenceError: Invalid left-hand side in assignment
... 생략 ...
```


자바스크립트 - 변수

■ 변수의 이름

- 변수명에는 공백을 넣을 수 없음
- 변수명은 문자나 밑줄(_), 달러 기호(\$)로 시작해야 하고, 첫 글자 이후로는 문자, 숫자, 밑줄, 달러 기호만 사용할 수 있음
- 변수명은 대소문자를 구분함
- 자바스크립트 예약어를 변수명으로 쓸 수 없음

자바스크립트 - 변수

■ 주석

- // 를 사용하여 소스코드 주석 처리 가능
- 해석을 생략하고 실행에서 제외
- 자신이나 다른 사람이 코드를 해석할 때 도움이 되도록 메모

예제 7-3 잘못된 코드 앞에 주석 추가

web2-nodejs/syntax/variable.js

... 생략 ...

```
// 1 = 2;
```

자바스크립트 - 변수

■ 변수 선언 키워드

- var: 호이스팅 시 undefined로 저장
- const: 호이스팅 시 초기화되지 않은 상태로 저장
- let: 호이스팅 시 초기화되지 않은 상태로 저장

자바스크립트 - 변수

■ 호이스팅(Hoisting)

- 자바스크립트는 실행 되기 전에 함수 안에 필요한 변수 값들을 모두 모아서 유효 범위의 최상단에 선언
 - 자바스크립트 Parser가 함수 실행 전에 함수 내용 확인
 - 함수 안에 존재하는 변수/함수선언에 대한 정보를 기억
 - 유효 범위: 함수 블록 { } 안에서 유효
- 즉, 함수 내에서 아래쪽에 존재하는 내용 중 필요한 값들을 끌어올려 처리

자바스크립트 - 변수

■ 호이스팅의 대상

- var 변수 선언과 함수선언문에서만 호이스팅 발생
 - var 변수/함수의 **선언**만 위로 끌어 올려지며, **할당**은 끌어 올려지지 않음 (undefined로 자동 초기화)
 - let/const 변수 선언과 함수표현식에서는 호이스팅 발생하지 않음
 - 실제로 발생은 하지만 초기화는 변수 선언문을 만났을 때 수행

```
console.log("hello");  
var myname = "HEEE"; // var 변수  
let myname2 = "HEEE2"; // let 변수
```

```
/** --- JS Parser 내부의 호이스팅(Hoisting)의 결과 - 위와 동일 --- */  
var myname; // [Hoisting] "선언"  
console.log("hello");  
myname = "HEEE"; // "할당"  
let myname2 = "HEEE2"; // [Hoisting] 발생 X
```

자바스크립트 - 변수

- 호이스팅의 문제점
 - 잠재적 버그 발생

```
var n = 1;  
function test() {  
  console.log(n);  
  var n = 2;  
  console.log(n);  
}  
test();
```

자바스크립트 - 변수

- 호이스팅의 문제점
 - 잠재적 버그 발생

```
var n = 1;  
function test() {  
  console.log(n);  
  var n = 2;  
  console.log(n);  
}  
test();
```

```
undefined  
2
```

자바스크립트 - 변수

■ 구글의 자바스크립트 스타일 가이드

Use const and let

Declare all local variables with either const or let.

Use const by default, unless a variable needs to be reassigned.

The var keyword must not be used.

1. *const와 let을 이용해서 변수를 선언라.*
2. *값을 재할당하는 경우가 아니라면, const를 디폴트로 사용하라.*
3. *var는 절대로 사용하지 말라*

자바스크립트 - 변수

■ 변수 선언 키워드

- var

- 중복 선언 가능
- 재할당 가능
- 함수 레벨 스코프

- const

- 중복 선언 불가능
- 재할당 불가능
- 블록 레벨 스코프

- let

- 중복 선언 불가능
- 재할당 가능
- 블록 레벨 스코프

```
// 첫번째 변수 선언+초기화
var a = 10;
console.log(a); // 10
```

```
// 두번째 변수 선언+초기화
var a = 20;
console.log(a); // 20
```

```
// 세번째 변수 선언(초기화X)
var a;
console.log(a); // 20
```

```
var a = 10;
a = 20;
console.log(a); // 20
```

```
let b = 111;
b = 222;
console.log(b); // 222
```

```
if(true) {
  var a = 10;
  console.log(a); // 10
}

console.log(a); // 10
```

```
// let 중복 선언
let a = 10;
let a = 20; // SyntaxError: Identifier 'a' has already been declared

// const 중복 선언
const b = 10;
const b = 20; // SyntaxError: Identifier 'b' has already been declared
```

```
const c = 111;
c = 222; // TypeError: Assignment to constant variable.
```

```
if(true) {
  let a = 10;
  console.log(a); // 10
}

console.log(a); // ReferenceError: a is not defined
```

자바스크립트 - 변수

■ 변수의 활용

- 데이터에 이름을 붙이는 이름표 역할
 - 해당 데이터가 의미하는 바를 쉽게 파악 (코드 가독성)

예제 7-5 변수를 사용하지 않았을 때

web2-nodejs/syntax/variable2.js

```
console.log('Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit  
in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat  
cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.');
```

예제 7-6 변수를 사용했을 때

web2-nodejs/syntax/variable2.js

```
var letter = 'Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit  
in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat  
cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.';  
console.log(letter);
```

자바스크립트 - 변수

■ 변수의 활용

- 중복되는 코드 줄임

• 좋은 프로그래밍의 조건: 중복을 피하라!

- 특정 데이터가 반복해서 사용되고, 의미를 가져 한꺼번에 수정하는 등 일괄적으로 제어할 필요가 있는 경우

예제 7-7 변수를 사용하지 않았을 때의 편지글

web2-nodejs/syntax/variable2.js

```
Dear egoing Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua. egoing Ut enim ad minim veniam, quis nostrud  
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in  
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint  
occaecat cupidatat non proident, sunt in culpa egoing qui officia deserunt mollit anim id est  
laborum. egoing
```

예제 7-8 변수를 사용했을 때의 편지글

web2-nodejs/syntax/variable2.js

```
var name = 'egoing';  
var letter = 'Dear ' + name + ' Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed  
do eiusmod tempor incididunt ut labore et dolore magna aliqua. ' + name + ' Ut enim ad minim  
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.  
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla  
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa ' + name + ' qui officia  
deserunt mollit anim id est laborum. ' + name;  
console.log(letter);
```

자바스크립트 - 템플릿 리터럴

■ 템플릿 문자열

- 자바스크립트에서 문자를 표현하는 방법 중 하나
- 예제
 - template.js 파일 작성

예제 8-1 template.js 파일 만들고 코드 붙여넣기

web2-nodejs/syntax/template.js

```
var name = 'egoing';  
var letter = 'Dear ' + name + '    ← 엔터 두 번으로 줄 바꿈
```

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut  
labore et dolore magna aliqua. ' + name + ' Ut enim ad minim veniam, quis nostrud exercitation  
ullamco laboris nisi ut aliquip ex ea  
in voluptate velit esse cillum dolore  
cupidatat non proident, sunt in culpa  
+ name;  
console.log(letter);
```

template.js 실행 결과

```
> node syntax/template.js  
... 생략 ...  
SyntaxError: Invalid or unexpected token  
    at Module._compile (internal/modules/cjs/loader.js:721:23)  
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:787:10)  
    at Module.load (internal/modules/cjs/loader.js:653:32)  
... 생략 ...
```

자바스크립트 - 템플릿 리터럴

■ 템플릿 문자열

- 역슬래시(\)를 입력하면 에러 해결 가능

예제 8-2 줄 바꿈하려는 곳에 역슬래시 추가

web2-nodejs/syntax/template.js

```
var name = 'egoing';  
var letter = 'Dear ' + name + ' \  
\  
Lorem ipsum dolor sit amet, ... 생략 ... deserunt mollit anim id est laborum. ' + name;  
console.log(letter);
```

template.js 실행 결과

```
> node syntax/template.js  
Dear egoingLorem ipsum dolor sit amet, ... 생략 ... deserunt mollit anim id est laborum. egoing
```

자바스크립트 - 템플릿 리터럴

■ 템플릿 문자열

- 줄바꿈을 위해서는 특수기호인 `\n` 을 사용

예제 8-3 줄 바꿈 하려는 곳에 `\n` 추가

web2-nodejs/syntax/template.js

```
var name = 'egoing';  
var letter = 'Dear ' + name + '\n\nLorem ipsum dolor sit amet, ... 생략 ... deserunt mollit anim  
id est laborum. ' + name;  
console.log(letter);
```

● ○ ● template.js 실행 결과

```
> node syntax/template.js  
Dear egoing  
  
Lorem ipsum dolor sit amet, ... 생략 ... deserunt mollit anim id est laborum. egoing
```

자바스크립트 - 템플릿 리터럴

■ 템플릿 문자열

- 템플릿 리터럴 사용

- 문자열 그 자체가 값을 나타냄
- 백틱(`)을 사용하여 문자열을 템플릿 리터럴로 표시

~	!	@	#	\$
1	2	3	4	
Tab	"	<	>	
,	,	.		
Caps Lock	A	O	E	
↑				

그림 8-2 키보드에서 역따옴표(`)의 위치

예제 8-4 템플릿 리터럴 적용

web2-nodejs/syntax/template.js

```
var name = 'egoing';
var letter = `Dear ${name}` ← 엔터 두 번으로 줄 바꿈
```

```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
labore et dolore magna aliqua. ${name} Ut enim ad minim veniam, ... 생략 ... deserunt mollit
anim id est laborum. ${name}`;
console.log(letter);

```

```

template.js 실행 결과
> node syntax/template.js
Dear egoing

Lorem ipsum dolor sit amet, ... 생략 ... deserunt mollit anim id est laborum. egoing

```

자바스크립트 - Boolean

■ Boolean 데이터 타입

- 단 두개의 데이터만 가능: true(참), false(거짓)

예제 14-1 boolean.js 파일 생성

web2-nodejs/syntax/boolean.js

```
console.log(true);  
console.log(false);
```

boolean.js 실행 결과

```
> node syntax/boolean.js  
true  
false
```


자바스크립트 - Boolean

■ Boolean 데이터 타입

- 'true', 'false'는 변수 이름으로 사용할 수 없음

예제 14-2 변수 이름으로 사용할 수 없는 true, false

web2-nodejs/syntax/boolean.js

```
console.log(true);  
console.log(false);  
a = 1;  
true = 1; ← 에러가 발생합니다
```

```
boolean.js 실행 결과  
> node syntax/boolean.js  
true = 1;  
^^^^  
  
ReferenceError: Invalid left-hand side in assignment  
    at Module._compile (internal/modules/cjs/loader.js:721:23)  
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:787:10)  
... 생략 ...
```

자바스크립트 - 비교 연산자

■ 비교 연산자

- 왼쪽 항과 오른쪽에 있는 항을 대상으로 값을 비교하는 이항 연산자
- 연산의 결과가 참이면 **true**, 거짓이면 **false**를 리턴
 - 즉, 비교 연산자를 사용하면 결과는 **Boolean** 데이터 타입

예제 15-2 크고 작음을 비교하는 연산자

web2-nodejs/syntax/comparison.js

```
console.log(1 == 1);    // true
console.log(1 == 2);    // false
console.log(1 > 2);     // false
console.log(1 < 2);     // true
```



```
comparison.js 실행 결과
> node syntax/comparison.js
true
false
false
true
```

자바스크립트 - 비교 연산자

■ 비교 연산자

- == 과 ===의 차이
 - 254 == '254' vs. 254 === '254'
 - true == 1 vs. true === 1

표 15-1 =, ==, === 연산자 비교

연산자	구분	의미
=	대입 연산자	오른쪽 값을 왼쪽 변수에 대입
==	비교 연산자	두 항의 값이 같은지 비교
===		두 항의 값과 데이터 타입이 같은지 비교

자바스크립트 - 제어문

■ 제어문의 필요성

- 프로그램을 실행할 때마다 다른 명령을 실행해야 하는 경우

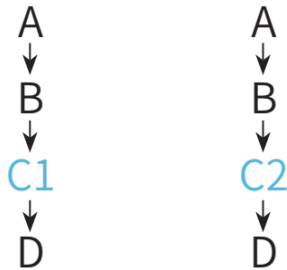


그림 16-2 실행할 때마다 다른 명령을 실행해야 하는 경우

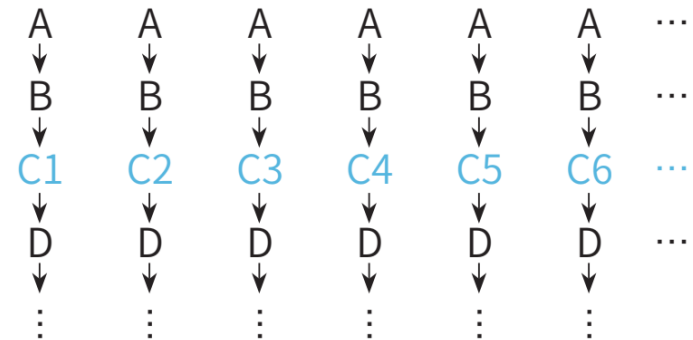


그림 16-3 중복되는 코드가 1억 줄이고, 프로그램이 1억 개라면?

자바스크립트 - 제어문

■ 제어문의 종류

- 조건문
 - 시간 순서에 따라 실행되는 실행의 흐름을 제어해서 다르게 동작하도록 할 수 있음
- 반복문
 - 조건에 따라 반복해서 실행

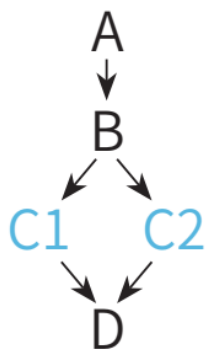


그림 16-4 조건문

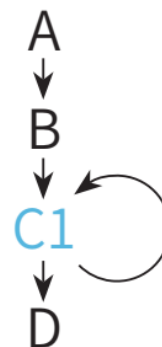


그림 16-5 반복문

자바스크립트 - 조건문

■ 조건문

- 조건에 따라 서로 다른 코드가 동작하도록 프로그램의 실행 흐름을 제어

예제 17-1 conditional.js 파일 생성

web2-nodejs/syntax/conditional.js

```
console.log('A');  
console.log('B');  
console.log('C1'); ← 분기에 따라 실행할 명령  
console.log('C2'); ←  
console.log('D');
```

예제 17-3 조건에 따라 실행하도록 if ~ else 조건문 추가

web2-nodejs/syntax/conditional.js

```
... 생략 ...  
if(true) {  
    console.log('C1'); ← 실행  
} else {  
    console.log('C2'); ← 실행하지 않음  
}  
... 생략 ...
```

자바스크립트 - 콘솔에서 입력받기

■ 프로그램

- 입력된 정보를 처리하여 결과를 출력



그림 18-1 프로그램의 입력과 출력

■ 매개변수(Parameter) vs. 인자(Argument)

- 프로그램에 필요한 값을 입력 받는 형식
- 형식에 맞게 실제 입력한 값

자바스크립트 - 콘솔에서 입력받기

■ 콘솔로부터 입력받기

- process 모듈의 argv라는 기능을 사용
 - argv = argument vector

예제 18-1 콘솔로부터 입력받기

web2-nodejs/syntax/conditional.js

```
var args = process.argv;  
console.log(args);  
console.log('A');  
console.log('B');  
... 생략 ...
```

- 프로그램을 실행할 때 넘어온 입력 값을 args 변수에 저장

자바스크립트 - 콘솔에서 입력받기

■ 콘솔로부터 입력받기

- args는 일반 변수가 아니라 배열 (Array)
 - Node.js 런타임이 위치한 경로
 - 실행 파일이 위치한 경로
 - 프로그램을 실행할 때 입력한 값들...

```
conditional.js 실행 결과(입력값이 'egoing k8805'일 때)
> node syntax/conditional.js egoing k8805
[ '/usr/local/bin/node',
  '/Users/username/Desktop/web2-nodejs/syntax/conditional.js',
  'egoing',
  'k8805' ]
... 생략 ...
```

자바스크립트 - 콘솔에서 입력받기

■ 콘솔로부터 입력받기

- 세 번째 입력 값만 출력
 - 대괄호와 숫자는 배열에서 몇 번째 데이터에 접근할 것인지를 나타냄
 - 0부터 시작

예제 18-2 콘솔로부터 입력한 값만 출력

web2-nodejs/syntax/conditional.js

```
var args = process.argv;  
console.log(args[2]);  
console.log('A');  
console.log('B');  
... 생략 ...
```

conditional.js 실행 결과

```
> node syntax/conditional.js egoing  
egoing  
... 생략 ...
```

자바스크립트 - 콘솔에서 입력받기

- 입력 값에 따라 다르게 동작하는 프로그램
 - 콘솔에서 입력한 값은 모두 문자열로 취급

예제 18-3 입력한 값에 따라 다르게 출력하기

web2-nodejs/syntax/conditional.js

```
var args = process.argv;
console.log(args[2]);
console.log('A');
console.log('B');
if(args[2] === '1') {
    console.log('C1');
} else {
    console.log('C2');
}
console.log('D');
```

conditional.js 실행 결과(입력값이 1일 때)

```
> node syntax/conditional.js 1
```

```
1
A
B
C1
D
```

conditional.js 실행 결과(입력값이 2일 때)

```
> node syntax/conditional.js 2
```

```
2
A
B
C2
D
```

자바스크립트 - 반복문

■ while 반복문

예제 20-2 특정 코드를 반복해서 실행

web2-nodejs/syntax/loop.js

```
console.log('A');  
console.log('B');  
console.log('C1');  
console.log('C2');  
console.log('C1');  
console.log('C2');  
console.log('C1');  
console.log('C2');  
console.log('D');
```

예제 20-4 반복문을 두 번만 반복하게 조건을 추가

web2-nodejs/syntax/loop.js

```
console.log('A');  
console.log('B');  
var i = 0;  
while(i < 2) {  
    console.log('C1');  
    console.log('C2');  
    i = i + 1;  
}  
console.log('D');
```

```
> node syntax/loop.js
```

A

B

C1

C2

C1

C2

D

자바스크립트 - 배열

■ 배열 만들기

- 배열 리터럴([])을 사용하여 만드는 방법

```
// 배열 생성 (초기 값 할당)  
var arr = ['zero', 'one', 'tow'];
```

- Array() 생성자 함수로 배열을 생성하는 방법

```
// 배열 생성 (초기 값 할당)  
var arr = new Array('zero', 'one', 'tow');
```

자바스크립트 - 배열

■ 배열에서 값 읽기

- 배열 전체 값 읽기

예제 21-2 배열 읽어오기

web2-nodejs/syntax/array.js

```
var arr = ['A', 'B', 'C', 'D'];
console.log(arr);
```

array.js 실행 결과

```
> node syntax/array.js
['A', 'B', 'C', 'D']
```

- 배열 안에 특정 값 가져오기 (인덱스 활용)

표 21-1 배열의 인덱스와 값

인덱스	0	1	2	3
값	'A'	'B'	'C'	'D'
읽어오기	arr[0]	arr[1]	arr[2]	arr[3]

자바스크립트 - 배열

■ 배열에서 값 갱신하기

예제 21-4 배열에서 값 갱신하기

web2-nodejs/syntax/array.js

```
var arr = ['A', 'B', 'C', 'D'];  
... 생략 ...  
arr[2] = 3;  
console.log(arr);
```

array.js 실행 결과

```
> node syntax/array.js  
... 생략 ...  
['A', 'B', 3, 'D']
```

자바스크립트 - 배열

■ 배열의 크기 구하기

예제 21-5 배열의 크기 구하기

web2-nodejs/syntax/array.js

```
var arr = ['A', 'B', 'C', 'D'];  
... 생략 ...  
console.log(arr.length);
```

array.js 실행 결과

```
> node syntax/array.js  
... 생략 ...  
4
```


자바스크립트 - 배열

■ 배열의 마지막에 값 추가하기

예제 21-6 배열의 끝에 값 추가하기

web2-nodejs/syntax/array.js

```
var arr = ['A', 'B', 'C', 'D'];  
... 생략 ...  
arr.push('E');  
console.log(arr);
```

array.js 실행 결과

```
> node syntax/array.js  
... 생략 ...  
['A', 'B', 3, 'D', 'E']
```

자바스크립트 - 배열과 반복문

■ 배열과 반복문의 활용

예제 22-2 배열을 반복문으로 순회하면서 배열의 값 출력

web2-nodejs/syntax/array-loop.js

```
var number = [1, 400, 12, 34, 5]; ← 배열을 생성합니다
var i = 0;                          ← 변수를 초기화합니다
while(i < 5) {                      ← 5번 반복합니다
    console.log(number[i]);          ← 배열의 첫 번째부터 다섯 번째 요소까지 차례로 조회합니다
    i = i + 1;                      ← i의 값을 1 증가시킵니다.
}
```

예제 22-5 배열의 크기에 따라 유연하게 반복 횟수를 결정

web2-nodejs/syntax/array-loop.js

```
var number = [1, 400, 12, 34, 5, 10000];
var i = 0;
while(i < number.length) {
    console.log(number[i]);
    i = i + 1;
}
```

자바스크립트 - 함수

- 함수의 기본 문법

```
function 함수이름 ( ) {  
    함수에서 실행할 코드;  
}
```

자바스크립트 - 함수

■ 함수의 활용

- 변수 → 데이터에 이름
- 함수 → 로직에 이름

예제 25-1 함수를 사용하지 않은 코드

web2-nodejs/syntax/function.js

```
console.log(1);
console.log(2);
console.log(3);
console.log('A');
console.log('Z');
console.log('B');
console.log(1);
console.log(2);
console.log(3);
console.log('F');
console.log('C');
console.log('P');
console.log('J');
console.log(1);
console.log(2);
console.log(3);
console.log('U');
console.log('A');
console.log('Z');
console.log('J');
console.log('I');
console.log(1);
console.log(2);
console.log(3);
```

예제 25-2 함수 정의

web2-nodejs/syntax/function.js

```
... 생략 ...
function f123() {
    console.log(1);
    console.log(2);
    console.log(3);
}
```

예제 25-3 기존 명령문을 함수를 실행하는 명령으로 대체

web2-nodejs/syntax/function.js

```
f123();
console.log('A');
console.log('Z');
console.log('B');
f123();
console.log('F');
console.log('C');
console.log('P');
console.log('J');
f123();
console.log('U');
console.log('A');
console.log('Z');
console.log('J');
console.log('I');
f123();

function f123() {
    console.log(1);
    console.log(2);
    console.log(3);
}
```

자바스크립트 - 함수

■ 함수의 입력

- 입력 값 두개를 받는 sum이라는 함수 정의

예제 25-5 기존 명령문을 함수를 실행하는 명령으로 대체

web2-nodejs/syntax/function2.js

```
function sum(first, second) {  
    console.log(first + second);  
}  
  
sum(2, 4);
```

function2.js 실행 결과

```
> node syntax/function2.js  
6
```

매개변수(Parameter) vs. 인자(Argument)

자바스크립트 - 함수

■ 함수의 출력

- 함수의 처리 결과를 반환
 - 함수를 호출한 쪽에서 결과값을 다양하게 활용 가능

예제 25-6 처리 결과를 함수를 호출한 쪽으로 반환하는 함수

web2-nodejs/syntax/function2.js

```
function sum(first, second) {  
    return first + second;  
}
```

```
console.log(sum(2, 4));
```

function2.js 실행 결과

```
> node syntax/function2.js
```

```
6
```

자바스크립트 - 함수

■ 함수의 출력

- return 문 이후에는 명령이 있어도 생략하고 함수를 종료

예제 25-7 return문 앞뒤로 출력문을 두고 테스트

web2-nodejs/syntax/function2.js

```
function sum(first, second) {  
    console.log('a');  
    return first + second;  
    console.log('b');  
}  
  
console.log(sum(2, 4));
```

function2.js 실행 결과

```
> node syntax/function2.js  
a  
6
```

THANK YOU

무단 복제 및 배포를 금지합니다.