

깃헙 4.3K 스타 웹 차트 오픈소스

billboard.js 성장기

SOSCON
SAMSUNG
OPEN SOURCE
CONFERENCE

NAVER 박재성

b||board.js?

- 2017/6 첫 공개된 D3.js 기반 SVG 웹 차트 라이브러리
- 2020 9월 기준, GitHub 4.3K star
- 선언적 인터페이스 - 총 226개의 옵션과 API를 제공
- 코드 베이스: 13,700 LOC
- 코어 메인테너: 1명

[참고] 14일 만에 GitHub 스타 천 개 받은 차트 오픈소스 개발기 (2017)

Examples

billboard.js

API Docs | GitHub

Theme: insight

Launch code editor JS TS

Chart

AreaChart

AreaRangeChart

BarChart

BubbleChart

BubbleDimensionChart

CombinationChart

DonutChart

GaugeChart

LineChart

LineChartWithRegions

MultipleXYLineChart

PieChart

RadarChart

ScatterPlot

SimpleXYLineChart

SplineChart

StackedAreaChart

StackedBarChart

StepChart

TimeseriesChart

Axis

AdditionalYAxis

AxisLabel

Area Chart

x	data1	data2
0	300	130
1	350	100
2	300	140
3	0	200
4	0	150
5	0	50

Sample code

```
<!-- Markup -->
<div id="areaChart"></div>
```

• Try it out by editing below code or click right sided buttons.
• for ESM imports usage, checkout [this link](#).

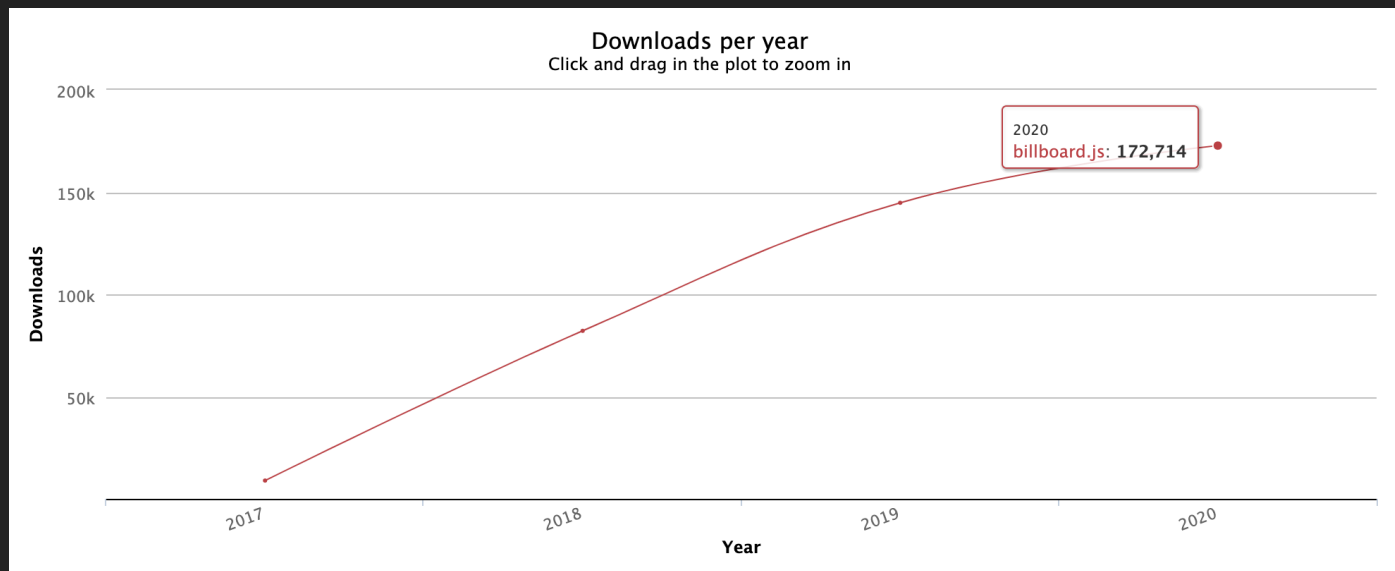
```
var chart = bb.generate({
  data: {
    columns: [
      ["data1", 300, 350, 300, 0, 0, 0],
      ["data2", 130, 100, 140, 200, 150, 50]
    ],
    types: {
      data1: "area",
      data2: "area-spline"
    }
  },
  bindto: "#areaChart"
});
```

JS TS Copy to Clipboard

[참고] <https://naver.github.io/billboard.js/demo/>

몇몇 지표들

- 20k npm downloads/month (누적 40만+)
8,941 (2017) → 82,147 → 144,749 → 20만+ 예상
- 200k CDN hits/month



공개 후, 지금까지 **40+** 컨트리뷰터들 참여

[참고] [npm downloads](#)

오픈소스 프로젝트의 지속 가능한 메인テナンス?

방법을 찾기 위한 긴 여정

- 1) 사용자 확보: 누가 사용할까?
- 2) 운영부담 덜어내기: 자동화
- 3) 지속적 신규 기능 개발: 릴리스

1) 사용자 확보

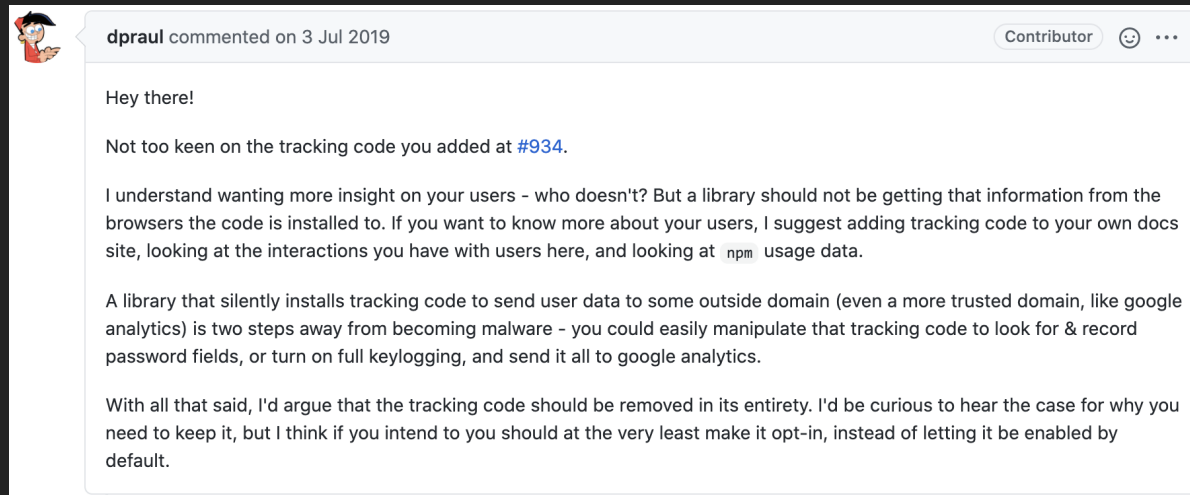
누가 사용할까?

누가 사용하는지 알고 싶다.

사용자가 드러나지 않고, 파악도 어려워 지표 수집 시도

- feat(stats): Intent to ship stats

그러나...



User tracking should be removed or disabled by default

[참고] Who's using billboard.js

2) 운영 부담 덜어내기 자동화

다양한 릴리스 채널

- nightly (커밋 기반 일간 릴리스)
- latest (stable)
- next (RC)

정기 릴리스는 매 3개월 주기로 진행

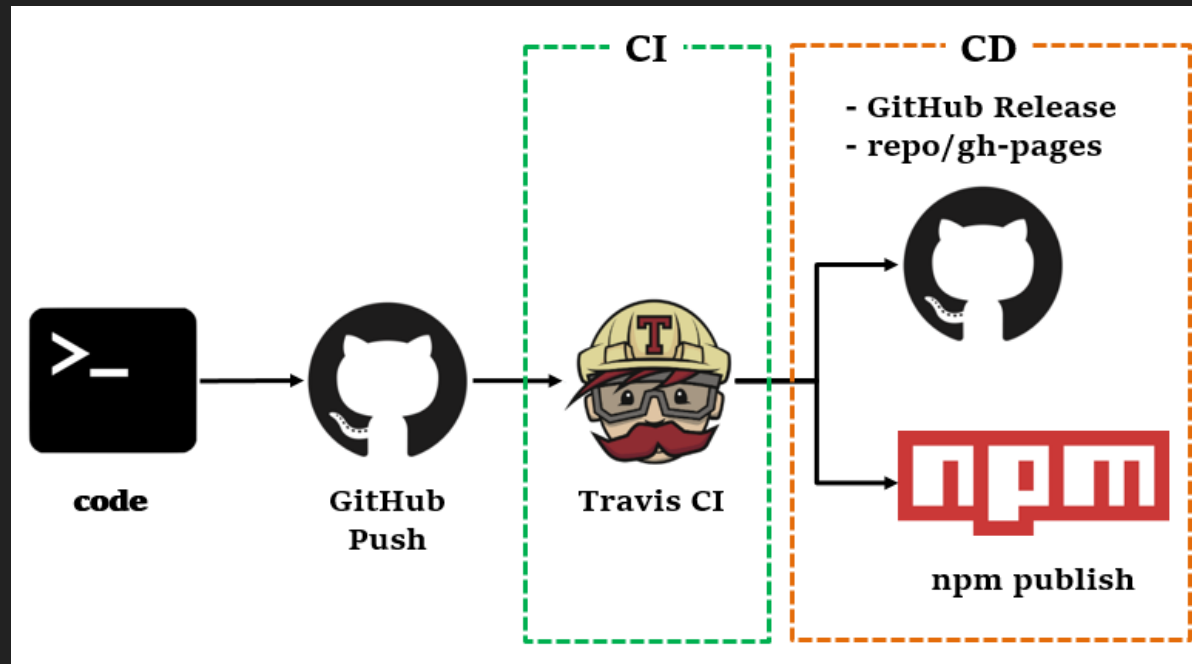
당연하지만, 채널이 많을수록
관리 부담도 증가한다.

한번의 정식 릴리스를 진행하려면...

1. master 브랜치 에서 x.x.x-rc 브랜치를 생성하고 이동
2. Package.json에서 차기 버전 정보를 변경
3. Regression 테스트 & linting 수행
4. 빌드 및 API doc을 생성
5. 직전 step에서 생성된 폴더와 변경사항을 commit
6. Changelog를 생성
7. Tagging 및 upstream에 push
8. GitHub "Draft a new release"를 통해 push된 tag를 새로운 릴리스로 등록
9. gh-pages에 빌드 파일과 API doc을 deploy
10. 패키징 후, npm publish 수행

릴리스 Workflow

CI와 CD 사이 또는 CI/CD 직전에 사람의 작업(개입)을 필요



그냥 커밋만 하면 알아서 릴리스 되었으면...

semantic-release

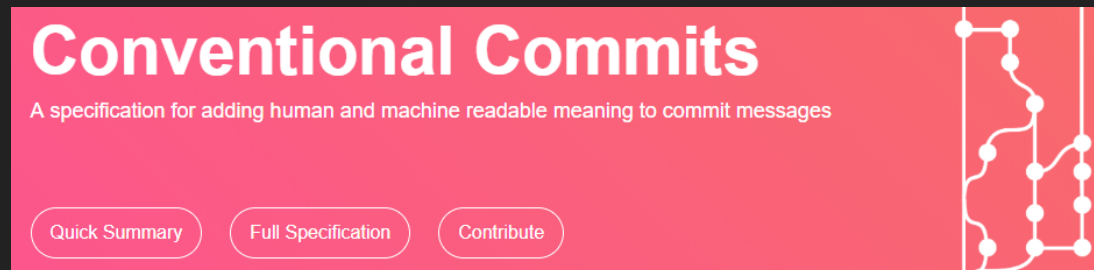
패키지 릴리스 워크플로 대부분을 자동화해 주는 도구



- 커밋 로그 분석을 통해 자동으로 차기 버저닝 설정
- 커밋 로그 기반의 릴리스 노트 자동생성
- 패키지 레지스트리(npm)에 자동 배포

[참고] <https://github.com/semantic-release>

Conventional Commits



```
<type>[optional scope]: <description>
```

```
[optional body]
```

```
[optional footer]
```

- 차기 버저닝을 위한 커밋 타입 분석에 사용
- Changelog/release note 생성에 사용

[참고] <https://www.conventionalcommits.org/>

릴리스 Trigger

커밋 로그의 type에 따라 릴리스 타입 결정되며,
해당 브랜치에 push 되면 릴리스 workflow가 자동으로 수행

```
fix(module): subject # Patch  
feat(module): subject # Minor  
  
perf(module): subject # Major/Breaking
```

```
BREAKING CHANGE: The option has been removed.
```

semantic-release는 커밋 로그에 기반해 릴리스 타입(버저닝)을 결정하므로 컨벤션에 따른 커밋 로그를 잘 작성하는게 관건


커밋 로그를 분석하고, 릴리스 타입을 설정

```
[6:53:08 AM] [semantic-release] [@semantic-release/commit-analyzer] > Analyzing commit: feat(radar): Intent to ship axis.text.position
```

- Make default axis text to be positioned not overlapping chart area
- Implement axis text position

Fix #998

```
[6:53:08 AM] [semantic-release] [@semantic-release/commit-analyzer] > The release type for the commit is minor
```



[참고] <https://travis-ci.org/naver/billboard.js/jobs/568700732>

Nightly

Nightly 빌드는 Cron Job을 통해 매일 1회 수행되도록 처리

The screenshot shows the 'Cron Jobs' configuration page for a job named 'nightly'. The job is configured to run daily. The 'INTERVAL' dropdown menu is open, showing options: Daily, Monthly, Weekly, and Daily (highlighted). The 'OPTIONS' section shows 'Always run'.

BRANCH	INTERVAL	OPTIONS
Select branch ▼	Daily ▲ Monthly Weekly Daily	Always run

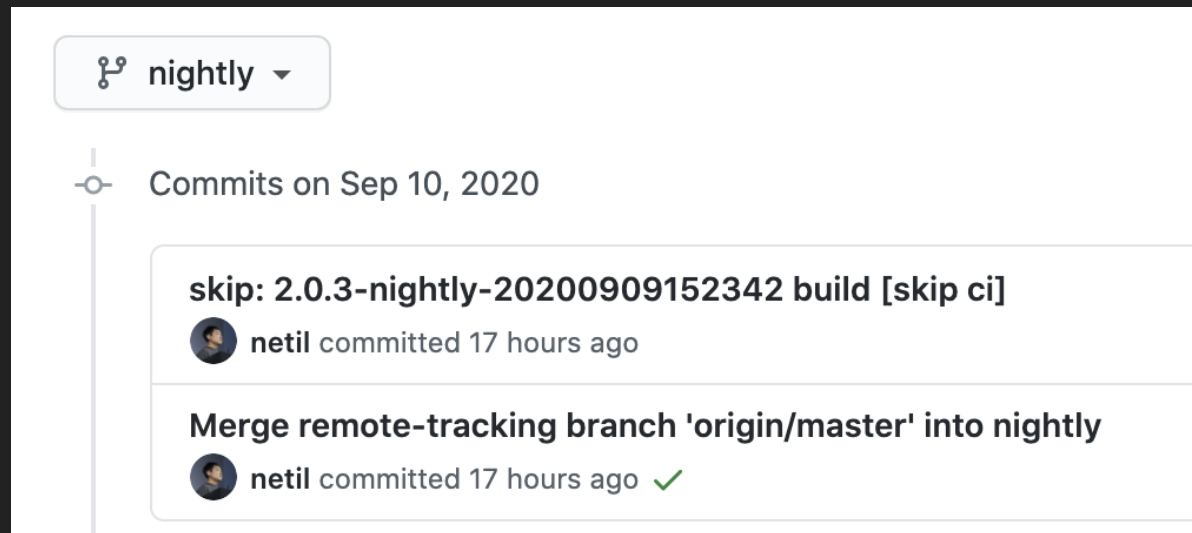
[참고] <https://docs.travis-ci.com/user/cron-jobs/>

Build Configuration

```
# .travis.yml
before_install:
  - npm install -g npm@latest
  - bash ./config/deploy-nightly.sh setup
...
before_script:
  - npm run lint
script:
  - npm run coverage
after_success:
  - bash ./config/deploy-nightly.sh build
```

[참고] <https://github.com/naver/billboard.js/blob/nightly/config/deploy-nightly.sh>

Nightly build commit



[참고] <https://github.com/naver/billboard.js/commits/nightly>

3) 지속적 신규 기능 개발

v2 릴리스

v2.0의 기술적 과제들

- 하위 호환성을 그대로 유지 시키기
- 파일 구조와 아키텍처 개선
- JS 프로젝트의 TS 전환
- 더 나은 성능: 실행속도 개선 & 빌드 크기 감소

테스트, 테스트 그리고 테스트

[참고] <https://github.com/naver/billboard.js/wiki/CHANGELOG-v2>

성능 개선 #1

불필요 노드 생성을 제어

- Look for possibility to decrease node generation
- Too bloated DOM with empty/hidden SVG elements

ex) Pie 유형 차트를 렌더링할 때,
사용되지 않는 축 관련 노드들도 생성되는 등

4 ~ 53% 감소

성능 개선 #2

모듈화를 통한 번들 크기 감소

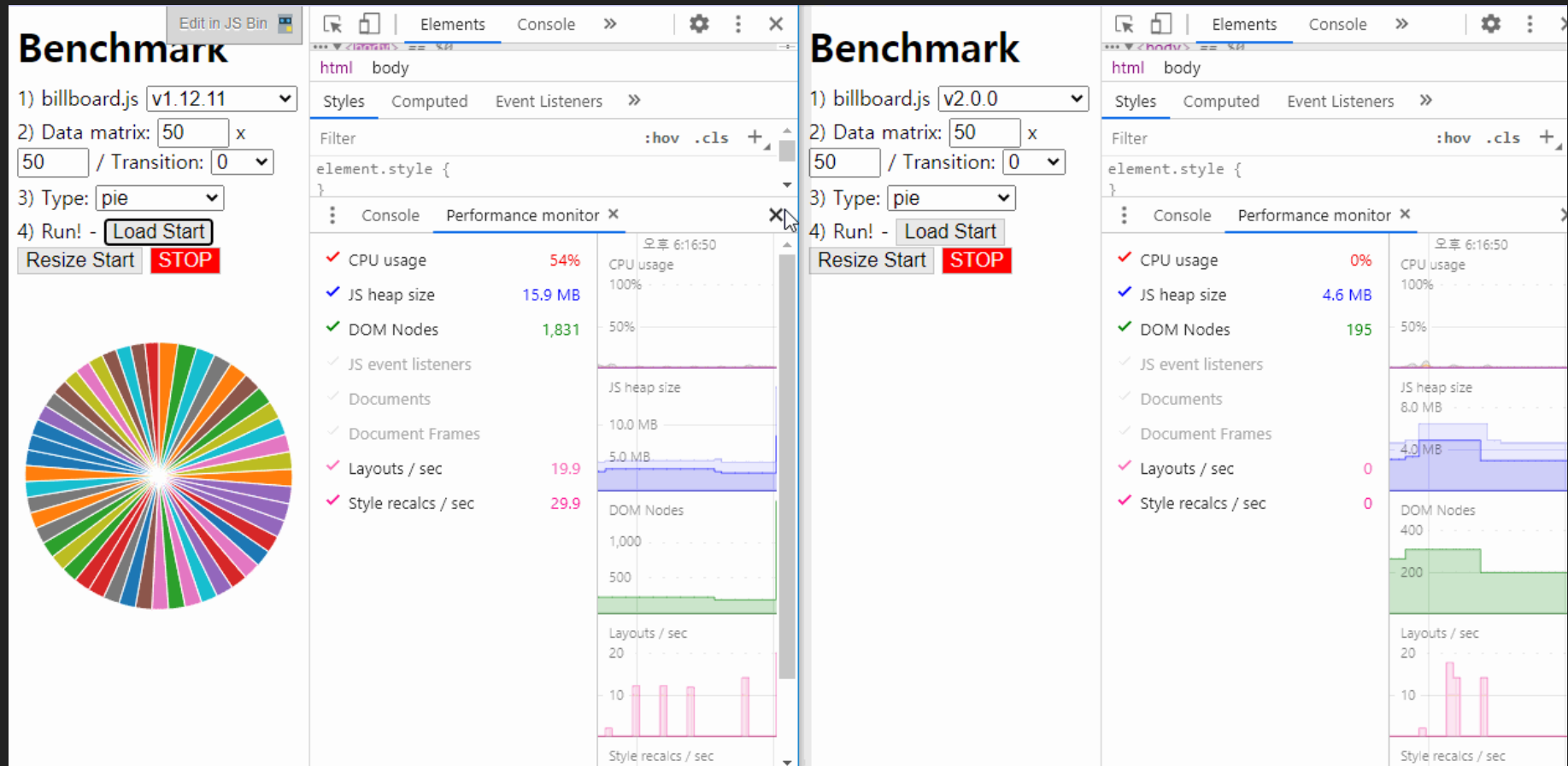
모듈화에 대한 고민 모든 차트 생성에 모든 타입의 유형을 사용하지 않기 때문

- Full modularization by its functionality

최신 번들러들은 Tree-shaking을 통해 사용되지 않는 코드들은 번들링에서 제외 기능을 제공한다.

10 ~ 43% 감소

Benchmark



[참고] <https://output.jsbin.com/pecafi>

모듈화 고민

기존 구조는 모든 모듈들이 prototype으로 확장되는 형태

```
Chart.prototype.resize = function(size) { ... }
```

완전한 모듈화를 위해서는 prototype 확장 형태를 변경해야 하나,
하위 호환성 문제와 테스트 등 너무 많은 시간 소요의 문제

- 기능적/유형별 코드를 그룹핑하고 분리
- ESM와 UMD 빌드에서 모두 사용할 수 있는 resolver

Resolver

```
import shapeArea from "../../ChartInternal/shape/area";

// extend 수행시 Chart.prototype로 확장
let area = (): string => (
  extend(ChartInternal.prototype, shapeArea),
  (area = () => TYPE.AREA)()
);

export {area, ... }
```

ESM

```
// shape module
export { area, ... } from "./config/resolver/shape";
```

UMD

```
import * as shape from "./config/resolver/shape";

// extends shape modules
Object.keys(shape).forEach(v => shape[v]());
```

하위 호환성 유지

- 기존 인터페이스를 해치지 않아야 한다.
- 해친다면, 마이그레이션 작업은 최소화 되어야 한다.

v1

```
bb.generate({  
  data: {  
    type: "line"  
  }  
});
```

v2

```
import {line, bar, pie} from "billboard.js";  
  
bb.generate({  
  data: {  
    type: line()  
  }  
});
```

빌드 제공

기존에는 webpack을 사용해 UMD 빌드를 제공 했으나,
ESM 빌드를 할수 없는 문제

2개의 번들러를 사용하는 전략 채택

- UMD:  webpack
- ESM:  Rollup.js

```
// package.json
scripts: {
  "build:esm": "rollup -c ./config/rollup/esm.js",
  "build:production": "cross-env NODE_ENV=production webpack --output-rep
}
```

[참고] [Rollup 설정](#) / [Webpack 설정](#)

모두의 Harmony

총 226개의 옵션, APIs 및 속성 들이 존재
차트 유형과 옵션들간 조합에 따라 다르게 동작될 수 있다.

모든 옵션들이 조화롭게 올바른 동작을 유지
하는 것은 매우 도전적인 일

[참고] <https://naver.github.io/billboard.js/release/latest/doc/>

테스트, 테스트 그리고 테스트

모든 케이스들에 대한 테스트 코드만이 유일한 방법

```
2007 SUMMARY:
2008 ✓ 1038 tests completed
2009 TOTAL: 1038 SUCCESS
2010
2011 ===== Coverage summary =====
2012 Statements   : 95.82% ( 4604/4805 )
2013 Branches    : 86.93% ( 4284/4928 )
2014 Functions   : 93.45% ( 871/932 )
2015 Lines       : 95.65% ( 3937/4116 )
2016 =====
```



[참고] <https://travis-ci.org/github/naver/billboard.js/jobs/725432665>
<https://coveralls.io/github/naver/billboard.js>

참여자 관점에 따라 다른 '오픈소스'

- 사용자의 입장에서
- 내가 필요한 것을 얻기 위한 컨트리뷰터의 입장
- 커미터의 입장
- 프로젝트 메인터너의 입장

메인터너의 입장에서

[SOSCON 2018] 오픈소스 개발: Behind the Scenes

Alex Ellis

OpenFaaS 쿠버네티스에 코드/함수 배포를 도와주는 도구를 개발



It can be extremely lonely as an open source maintainer because at the end of the day, there's only one person who's got their neck on the line, and that's me.

[참고] [Balancing open source sacrifice and success](#)

TJ Holowaychuk

Node.js 버전 관리자인 `n`, `Express.js`를 개발



In the end open-source doesn't pay the bills
so it's best to focus on other things if you can.

[참고] Has TJ Holowaychuk been as prolific in the Golang community as he was in the Node.js community?

오픈소스 참여를 통해 무엇을 얻을 수 있을까?

- 다른이의 코드를 통해 배울 수 있다.
- 다른 세계의 개발자와 협업을 경험할 수 있다.
- 더 나은 세상을 만들 수 있도록 다른 이들을 도울 수 있다.

오픈소스로 부터 혜택을 받지 않은 개발자는 아무도 없다.

지금까지 여러분의 프로젝트에서 '오픈소스'를
사용하지 않고 개발할 수 있었던 경험이 있으신가요?

오픈소스 참여의 의미

- 내가 받은 도움을 되돌려 주는 것
- 무엇을 기대하기 보다는 흥미있고 재미있다면 일단 시작
하지만, 너무 매몰되지 않게.

If you really enjoy the project(s) you're working on then go for it but don't neglect other areas of your life (or people). - TJ Holowaychuk

고맙습니다.
Thank You.
Gracias.