

Vue.js 소개

목차

- ▶ Vue.js 개발 환경 구성
- ▶ Vue.js 추세
- ▶ Vue.js 라이브러리 생태계
- ▶ Vue.js 향후 로드맵
- ▶ Vue.js 코딩 컨벤션과 스타일 가이드
- ▶ MVVM 패턴과 Reactivity
- ▶ 미니 라이브러리 구현

개발 환경 구성

개발 환경 구성

- ▶ Chrome
- ▶ Visual Studio Code
- ▶ Vue.js Dev Tools
- ▶ Node.js

VSCode 유용한 플러그인 목록

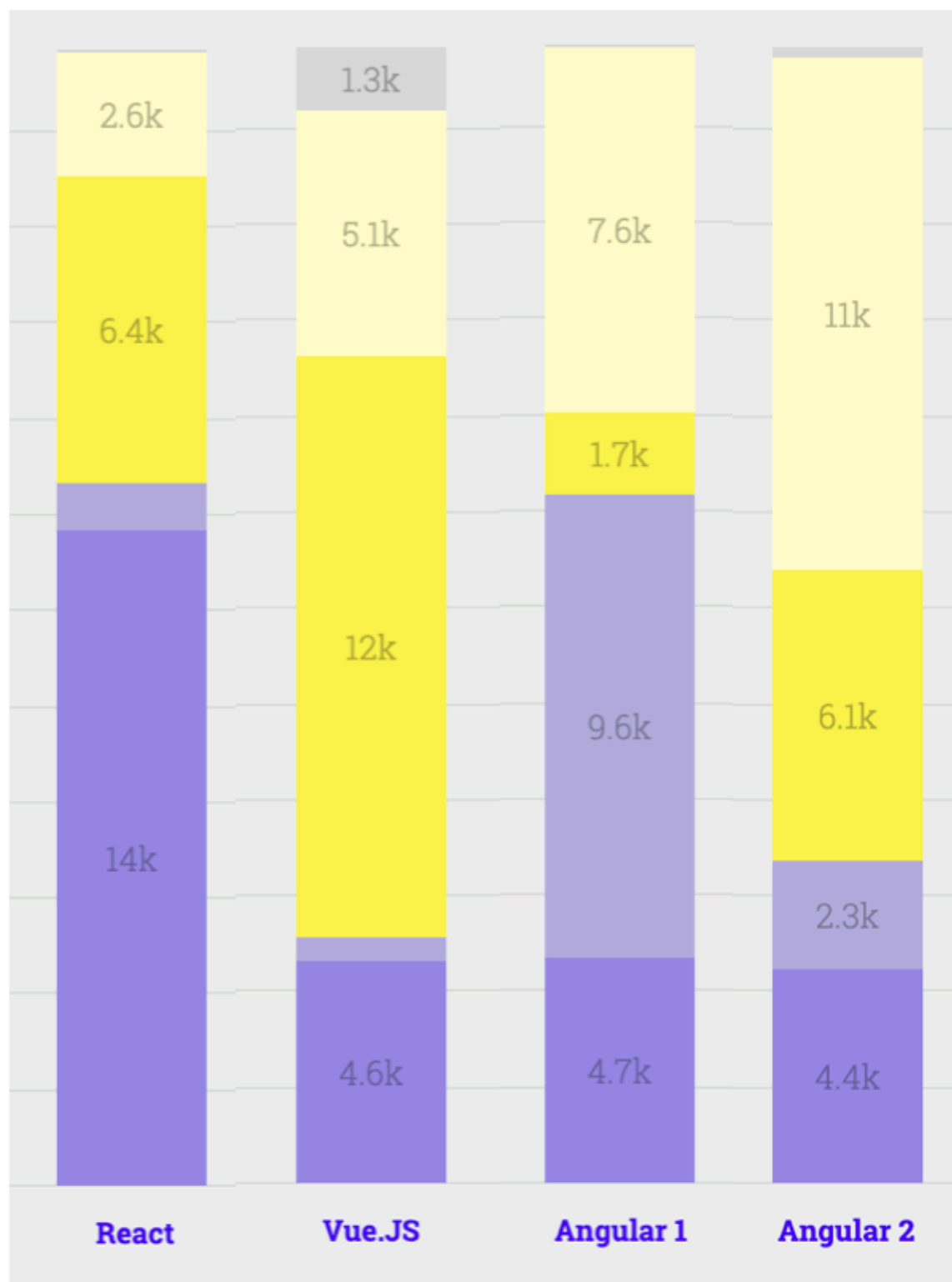
- ▶ 색 테마 : [Night Owl](#)
- ▶ 파일 아이콘 테마 : [Material Icon Theme](#)
- ▶ 뷰 확장 플러그인 : [Vetur](#)
- ▶ 뷰 코드 스니펫 : [Vue VSCode Snippets](#)
- ▶ 문법 검사 : ESLint, [ISLint](#)
- ▶ 실습 환경 보조 : [Live Server](#)
- ▶ 기타
 - ▶ [Prettier](#), [Project Manager](#), [Auto Close Tag](#), [GitLens](#), [Atom Keymap](#), [Jetbrains IDE Keymap](#) 등

Vue.js 추세

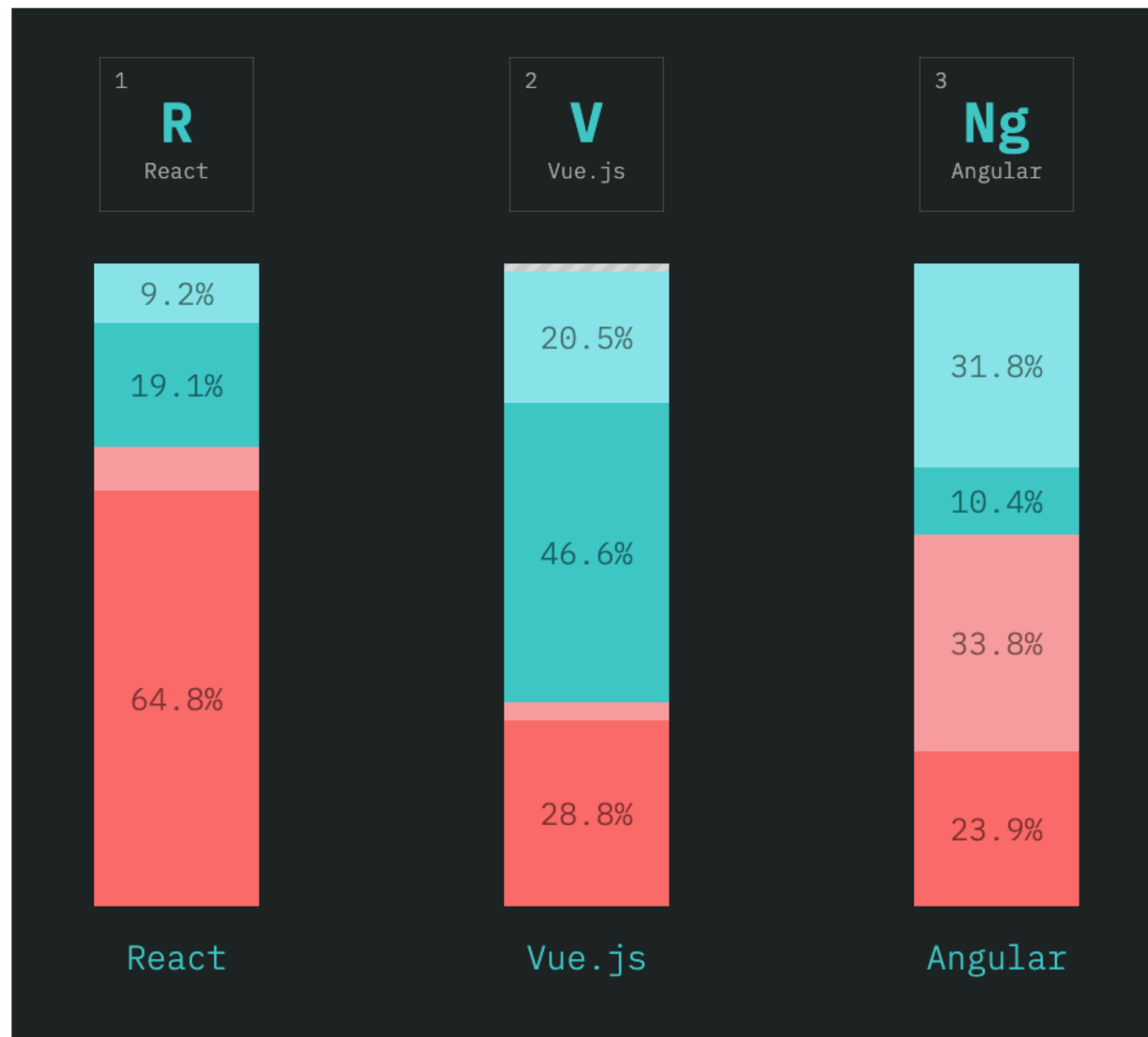
Vue.js 추세

- ▶ 미디어와 언론에서 얘기하는 Vue.js 추세
 - ▶ State of JS
 - ▶ State of Vue.js
- ▶ 실무자, 강사 관점에서 바라보는 Vue.js 추세

State of JS



2017



2018

State of Vue.js

THE MOST IMPORTANT REASON BEHIND ADDING VUE TO THE TECH STACK

Vue.js is pretty easy to start with

59%

PROBABILITY OF USING VUE IN UPCOMING PROJECT

THE BIGGEST ADVANTAGE

Percentages do not sum up to 100% due to the multiple choices.

2019

2017

Ease of integration

76%

81%

Great documentation

75%

45%

Performance

51%

56%

Progressiveness

50%

49%

Highly involved community

36%

29%

Other

5%

4%

45%

5%

18%

74%

Very high

실무자, 강사 관점에서 바라본 Vue.js 추세

- ▶ jQuery 레거시 코드에서 **점진적인 적용**

실무자, 강사 관점에서 바라본 Vue.js 추세

- ▶ jQuery 레거시 코드에서 **점진적인 적용**
- ▶ **디자이너, 퍼블리셔** 분들의 역량 강화 및 직무 전환

실무자, 강사 관점에서 바라본 Vue.js 추세

- ▶ jQuery 레거시 코드에서 **점진적인 적용**
- ▶ **디자이너, 퍼블리셔** 분들의 역량 강화 및 직무 전환
- ▶ **백엔드 개발자** 분들에게 좋은 선택지

실무자, 강사 관점에서 바라본 Vue.js 추세

- ▶ jQuery 레거시 코드에서 **점진적인 적용**
- ▶ **디자이너, 퍼블리셔** 분들의 역량 강화 및 직무 전환
- ▶ **백엔드 개발자** 분들에게 좋은 선택지
- ▶ 프론트엔드 개발 리소스가 부족한 프로젝트 - **많은 기업들의 현실적인 대안**

실무자, 강사 관점에서 바라본 Vue.js 추세

- ▶ jQuery 레거시 코드에서 **점진적인 적용**
- ▶ **디자이너, 퍼블리셔** 분들의 역량 강화 및 직무 전환
- ▶ **백엔드 개발자** 분들에게 좋은 선택지
- ▶ 프론트엔드 개발 리소스가 부족한 프로젝트 - **많은 기업들의 현실적인 대안**

“앞으로도 더 꾸준히 인지도가 상승할 것으로 추측”

Vue.js 라이브러리 생태계



The Progressive JavaScript Framework

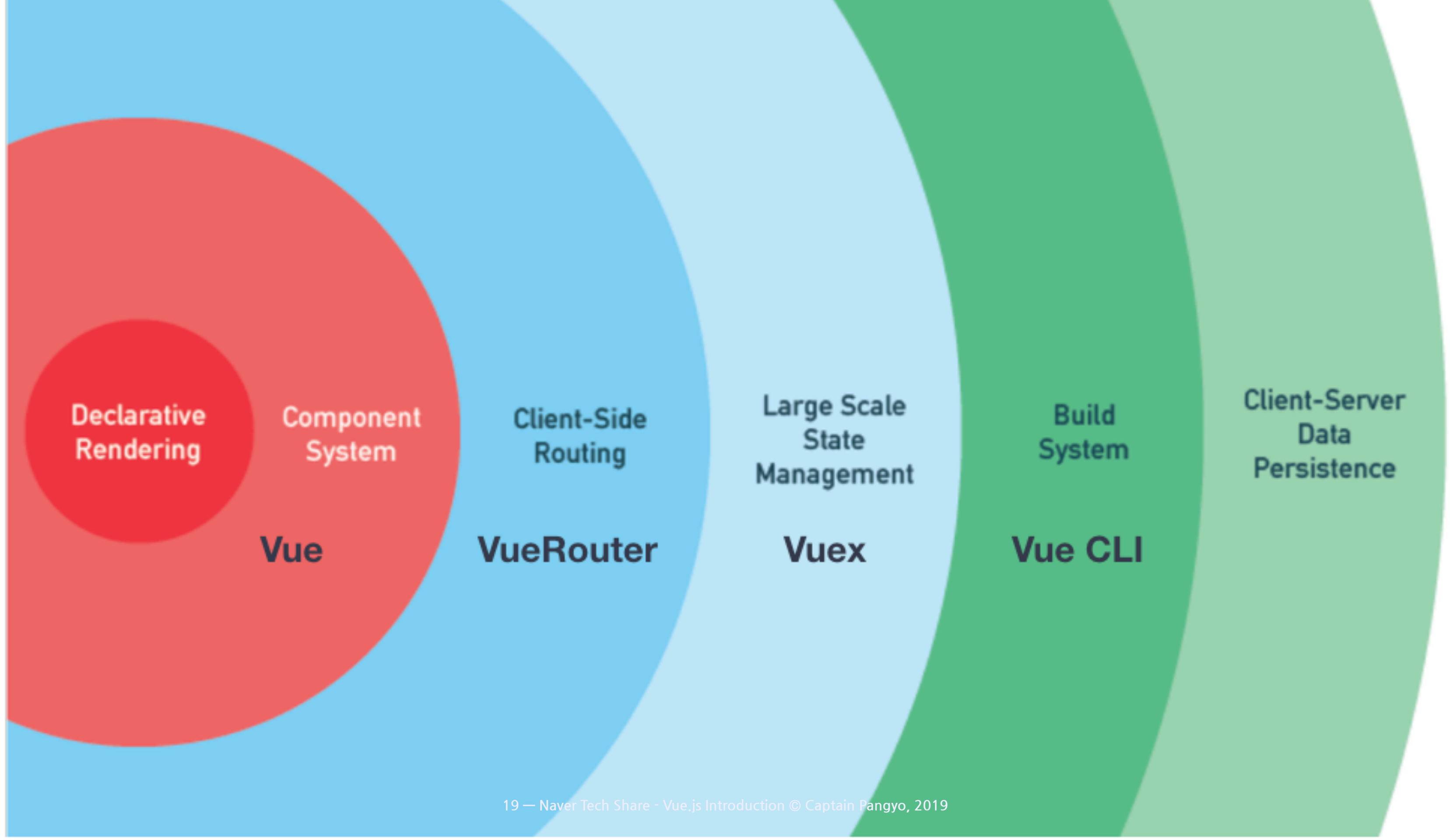


WHY VUE.JS?

GET STARTED



GITHUB



Vue.js 라이브러리 생태계

- ▶ Vue Core
- ▶ Vue Router
- ▶ Vue CLI
- ▶ Vuex
- ▶ Vue Devtools
- ▶ Nuxt
- ▶ Vue Test Utils

Vue.js 라이브러리 생태계

- ▶ Vue Core - 화면 렌더링 및 DOM 조작 API
- ▶ Vue Router - 라우팅
- ▶ Vue CLI - 프로젝트 설정 도구
- ▶ Vuex - 상태 관리
- ▶ Vue Devtools - 디버깅 도구
- ▶ Nuxt - 서버 사이드 렌더링
- ▶ Vue Test Utils - 테스트 도구

Vue.js 향후 로드맵

Vue.js 향후 로드맵

▶ Vue Core

- ▶ 올해 하반기 메이저 버전 3.x 업데이트 예정(Rewrite with ES6, TS)
- ▶ 2.x(IE 지원 O), 2.x-next(IE 지원 X) 2개 버전으로 유지보수
- ▶ 렌더링 추적, 불필요 렌더링 제거, 타입스크립트 지원 강화 등

▶ Vuex

- ▶ 메이저 버전 업데이트 예정(헬퍼 함수, actions 제거)

▶ 기타

- ▶ Experimental Hooks API, Exposed Reactivity API 지원 등

Vue.js 향후 로드맵

▶ 시사점

- ▶ RFC로 커뮤니티, 사용자와 소통하며 라이브러리 업데이트 방향 결정
- ▶ 기존 코드의 호환성과 문법 변화를 최대한 자제
- ▶ 더이상 개인 프로젝트가 아닌 팀, 오픈소스 커뮤니티 프로젝트로 발전

기존 코드의 호환성 보장과 지속 가능한 커뮤니티 성장

Vue.js 코딩 컨벤션과 스타일 가이드


Style Guide — Vue.js

https://vuejs.org/v2/style-guide/


Vue.js


LearnEcosystemTeamSupport VueTranslations


Special Sponsor


 Standard Library


Platinum Sponsors


 MODUS

 Share Code

 tooltwist

 Vue School

 VEHIKL

 NativeScript


[Become a Sponsor](#)

Style Guide

This is the official style guide for Vue-specific code. If you use Vue in a project, it's a great reference to avoid errors, bikeshedding, and anti-patterns. However, we don't believe that any style guide is ideal for all teams or projects, so mindful deviations are encouraged based on past experience, the surrounding tech stack, and personal values.

For the most part, we also avoid suggestions about JavaScript or HTML in general. We don't mind whether you use semicolons or trailing commas. We don't mind whether your HTML uses single-quotes or double-quotes for attribute values. Some exceptions will exist however, where we've found that a particular pattern is helpful in the context of Vue.

Soon, we'll also provide tips for enforcement. Sometimes you'll simply have to be disciplined, but wherever possible, we'll try to show you how to use ESLint and



Limited time offer: Get 10 free Adobe Stock images.

ads via Carbon

컴포넌트 이름은 최소 두 단어 이상

```
<!-- X -->
```

```
<todo></todo>
```

```
<!-- 0 -->
```

```
<todo-list></todo-list>
```

컴포넌트의 데이터 속성은 반드시 함수로 선언

// X

```
data: {  
  num: 10;  
}
```

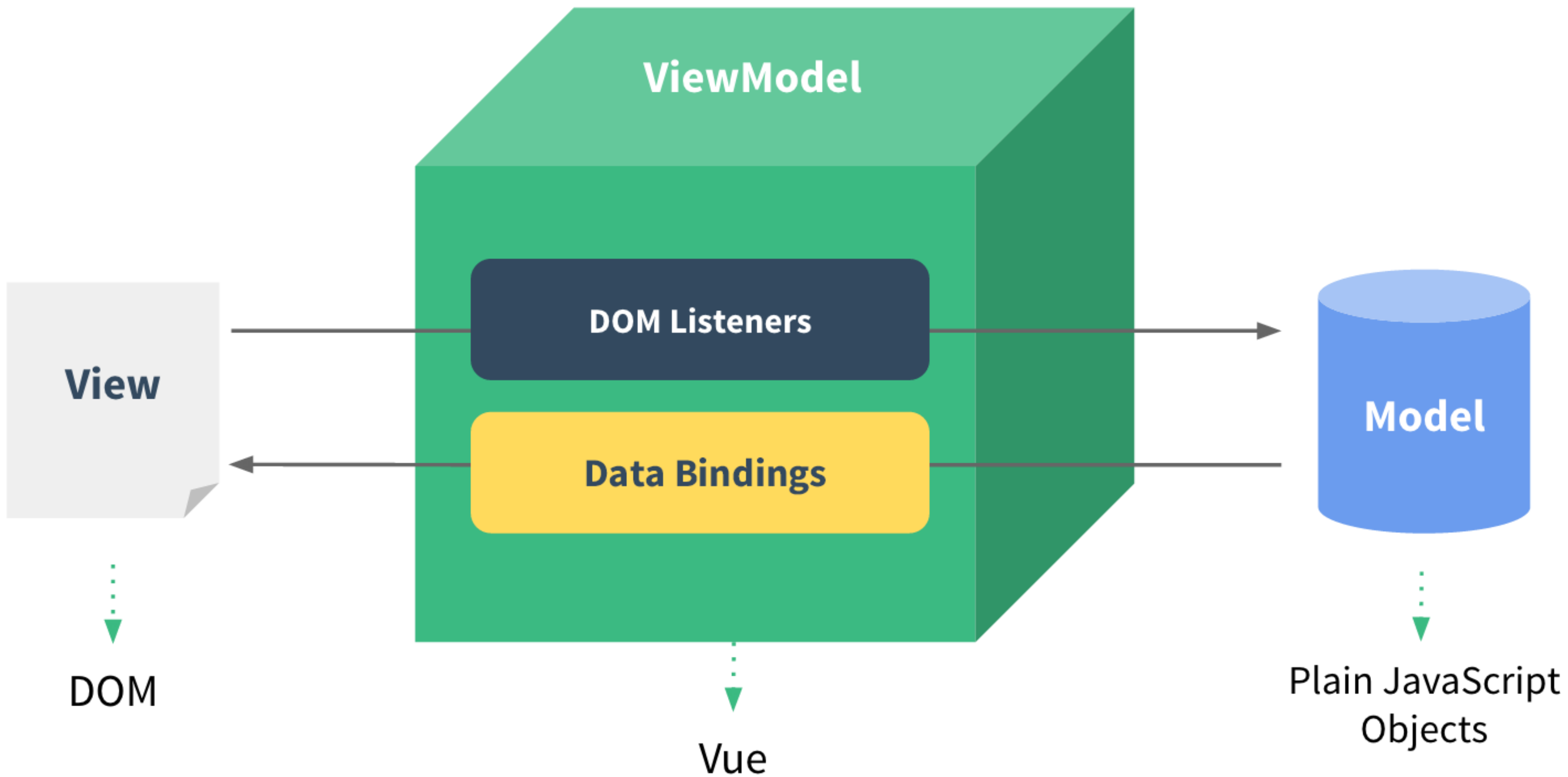
// O

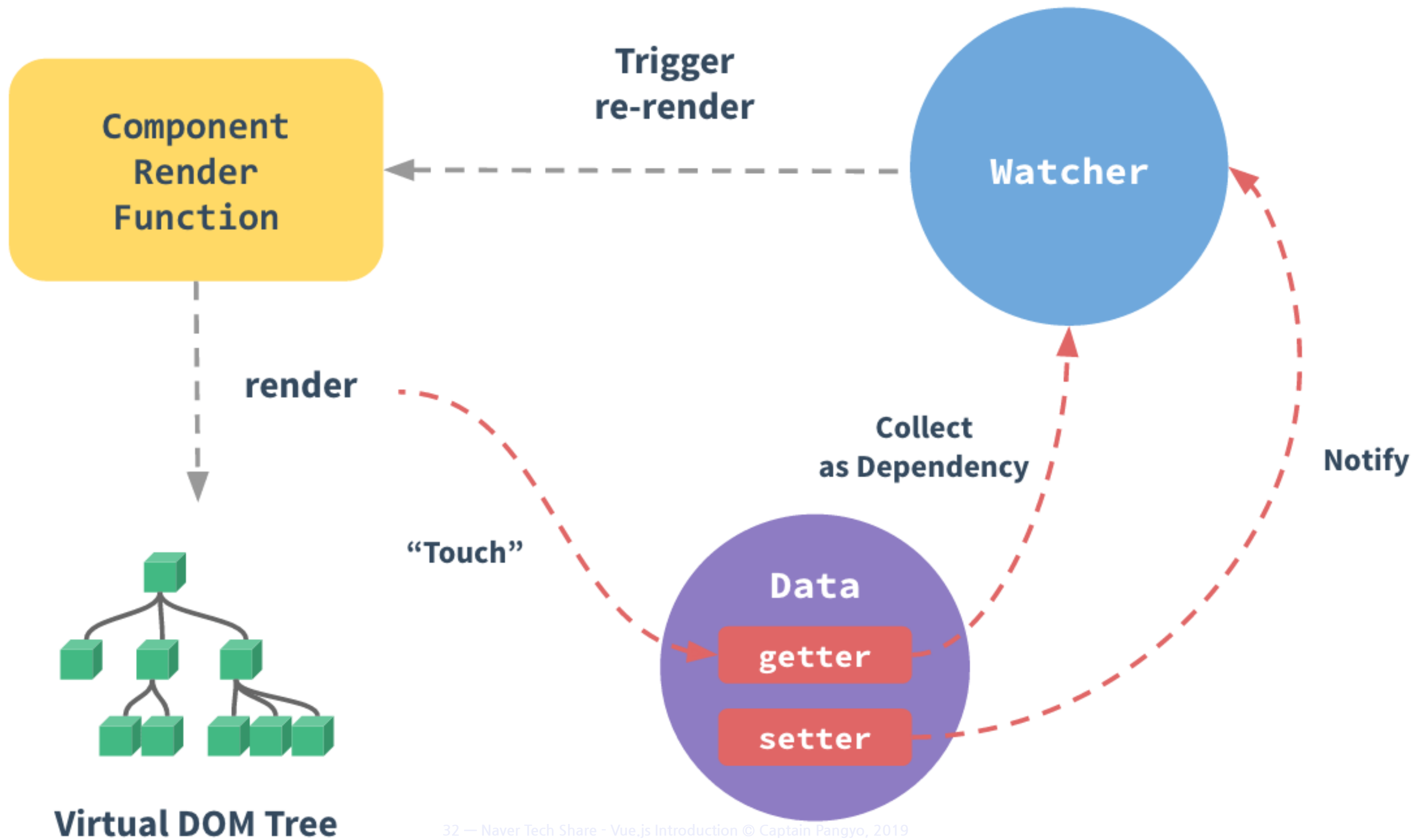
```
data: function() {  
  return { num: 10 }  
}
```

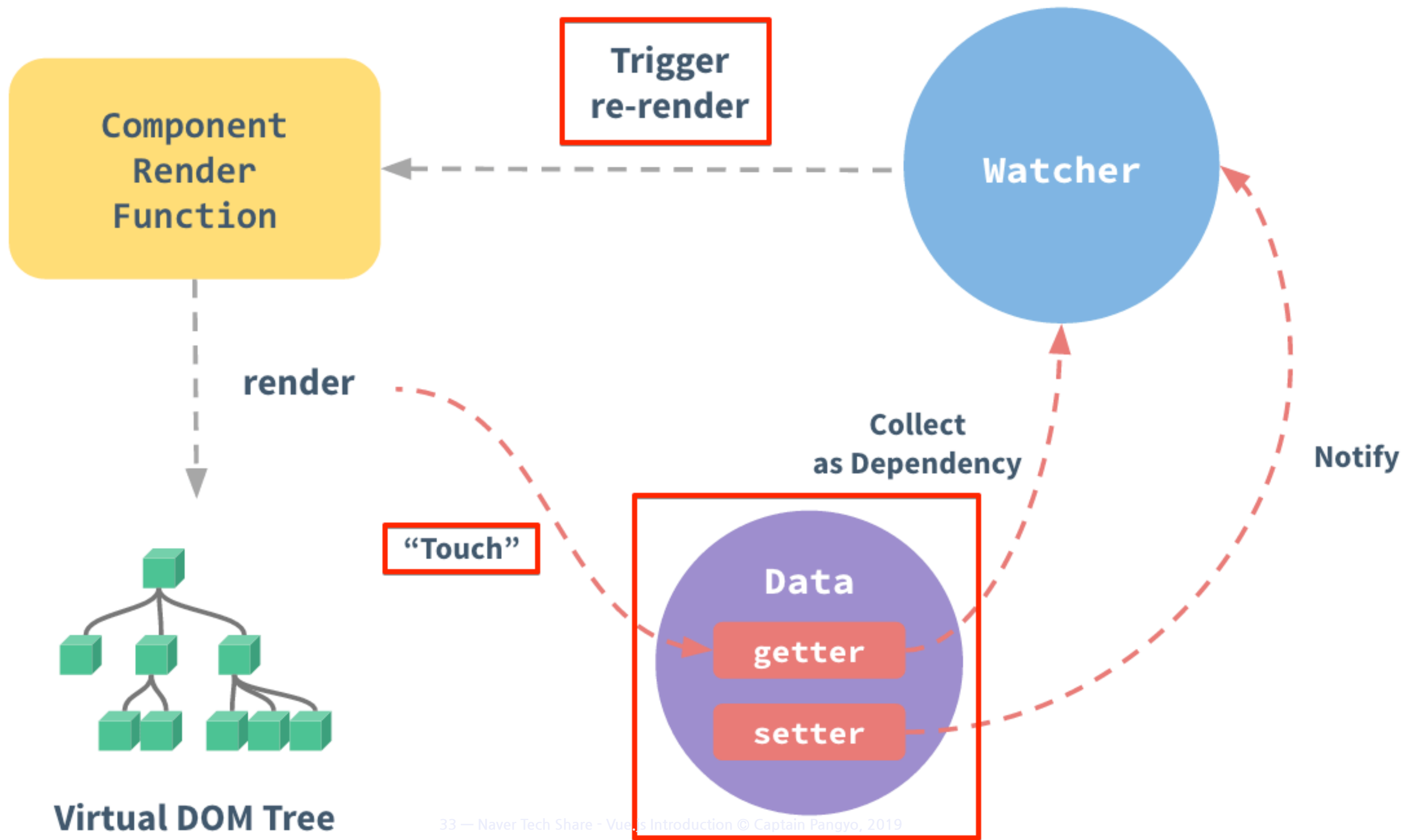
Vetur, Prettier, ESLint와 함께..!

<https://bit.ly/2UYTUQV>

MVVM 패턴과 Reactivity







Object.defineProperty()

```
var viewModel = {};
```

```
Object.defineProperty(viewModel, 'str', {  
  get: function() {  
    // 값이 접근되었을 때 실행할 로직  
  },  
  set: function() {  
    // 값이 할당되었을 때 실행할 로직  
  },  
});
```

Reactivity 구현 실습

Reactivity 구현 실습

아래 조건을 만족하는 Reactivity를 구현해보세요.

- ▶ 객체의 특정 속성을 접근했을 때 콘솔에 '접근' 출력
- ▶ 객체의 특정 속성에 값을 할당 했을 때 할당된 값으로 DOM을 갱신

끝