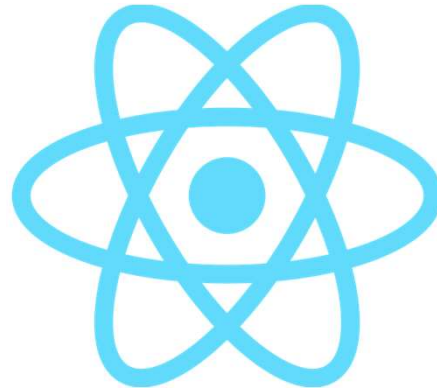


Introduction to React



Bok, Jong Soon
javaexpert@nate.com
<https://github.com/swacademy/React>

현대 Front-end 개발과 React

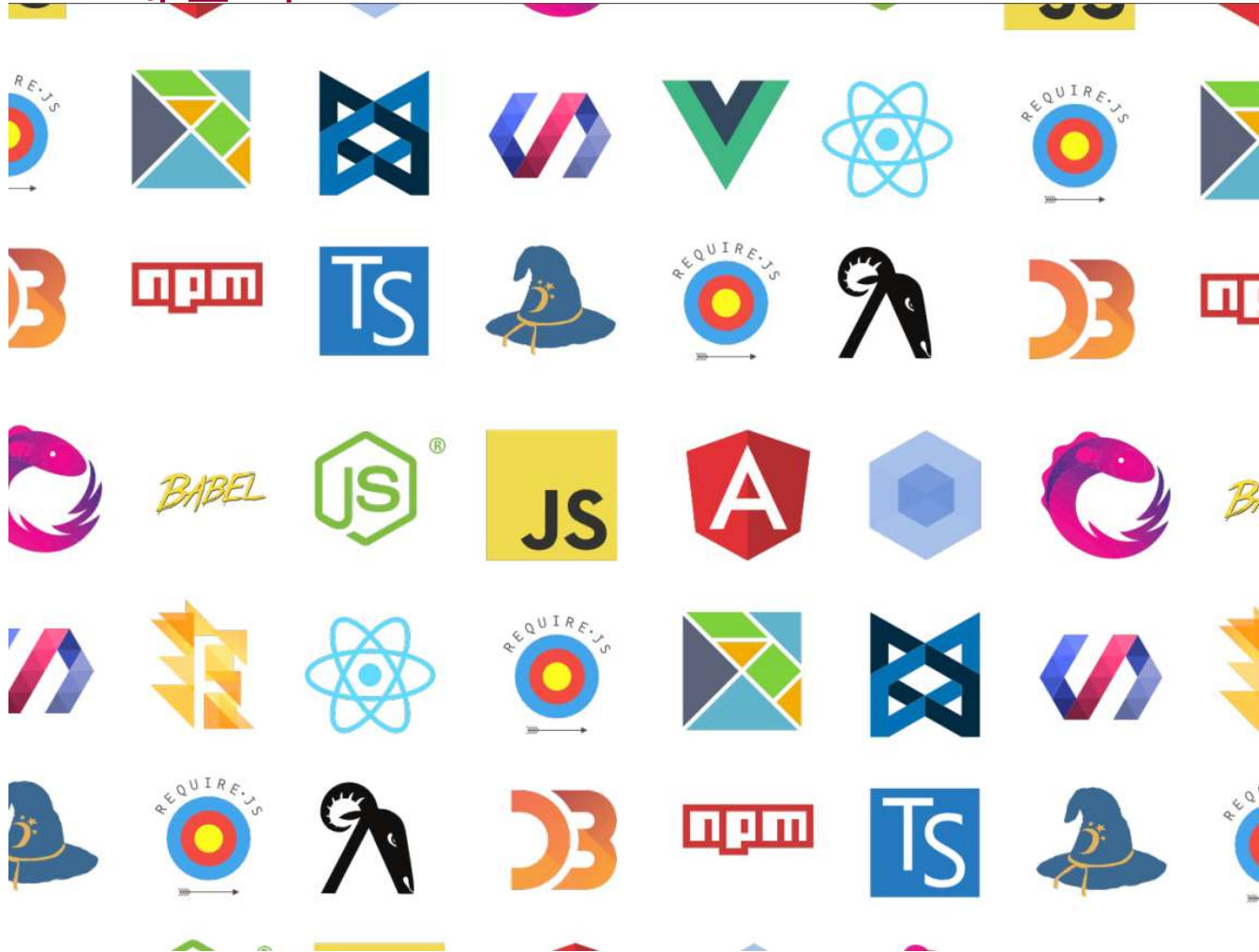
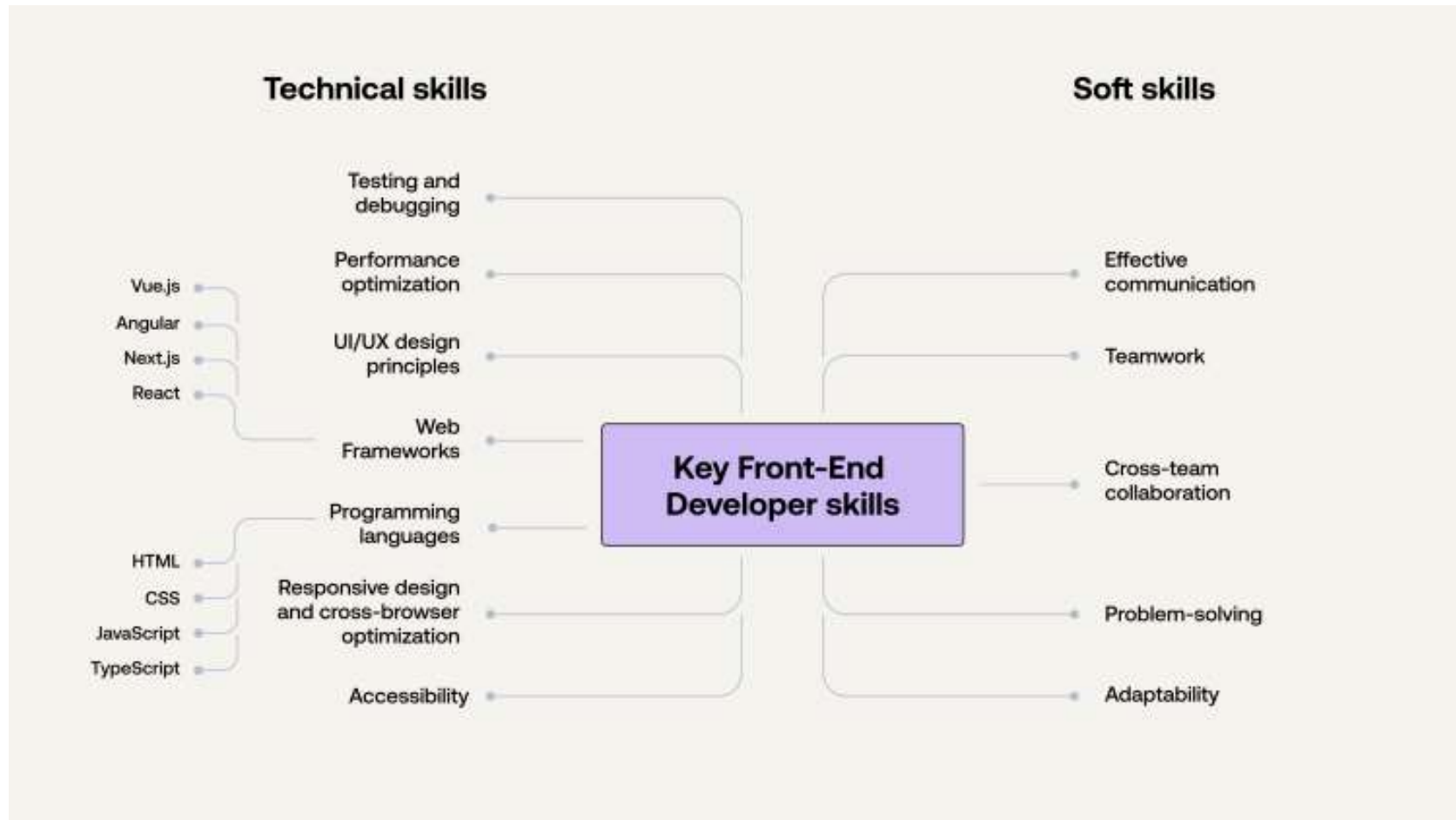
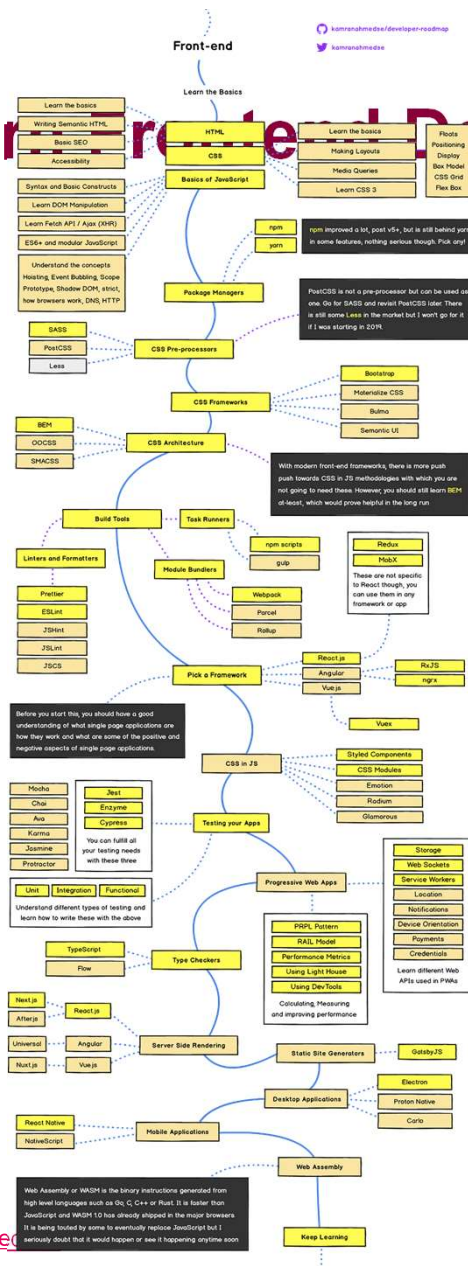


Image from : 장기호, Vue.js를 활용한 웹애플리케이션 개발 입문 강의자료, 2019년 9월 25일 Tacademy 토크ON세미나, 6th slide

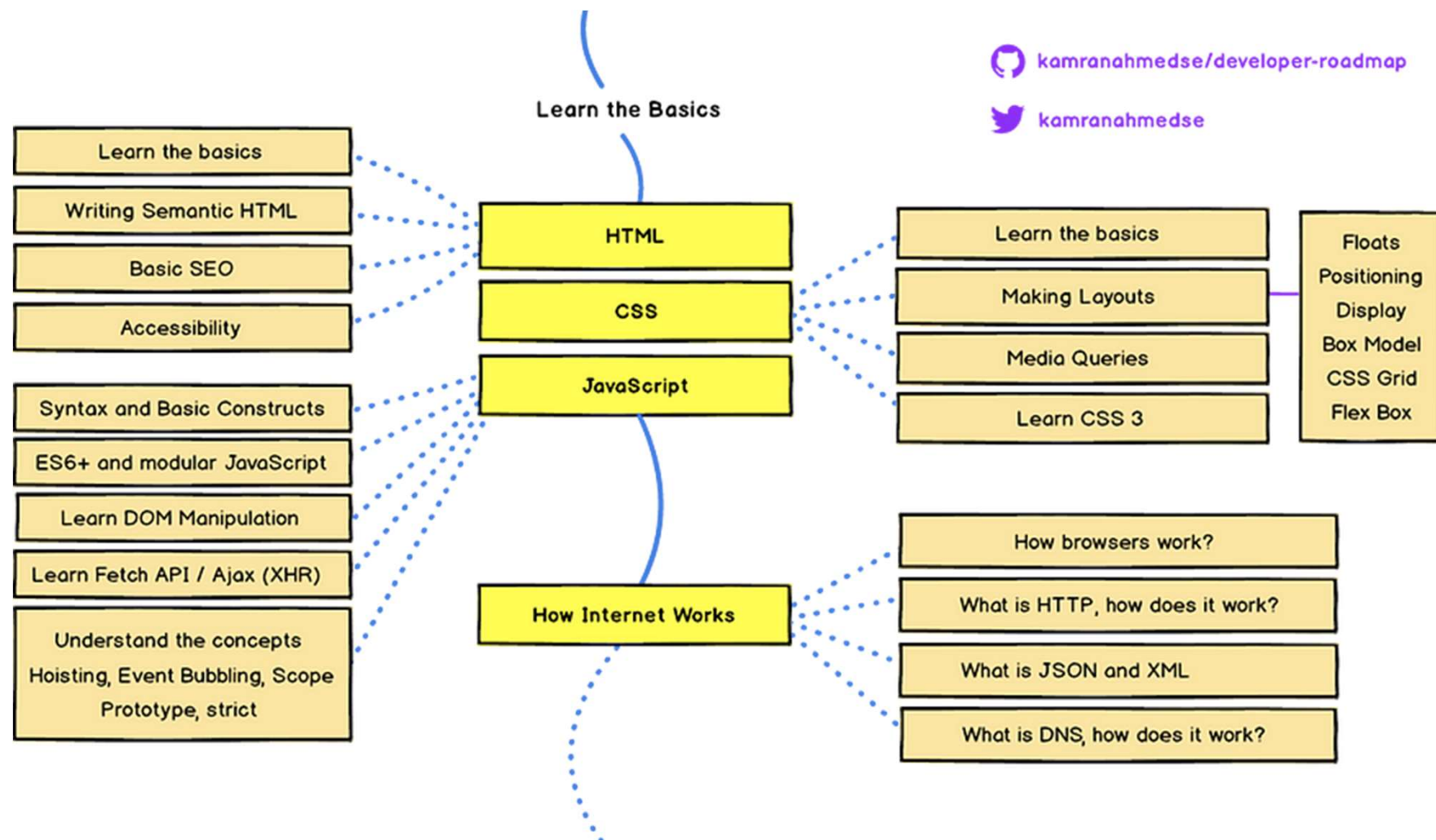
How to Become a Front-End Developer in 2024: The Ultimate Guide



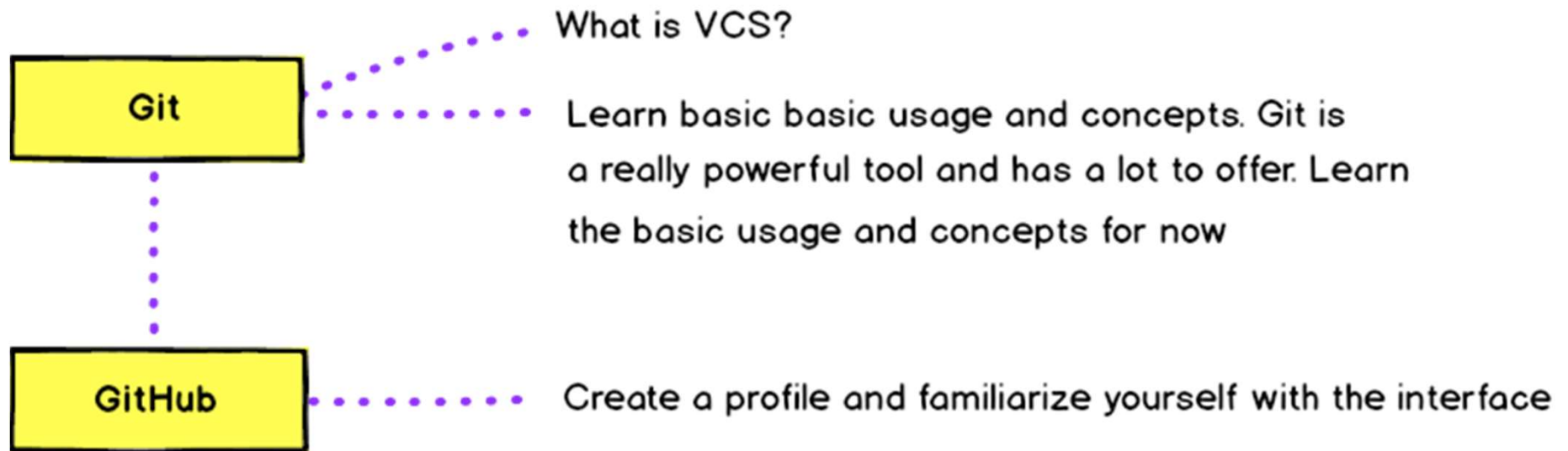
Learn to become a modern frontend Developer in 2022



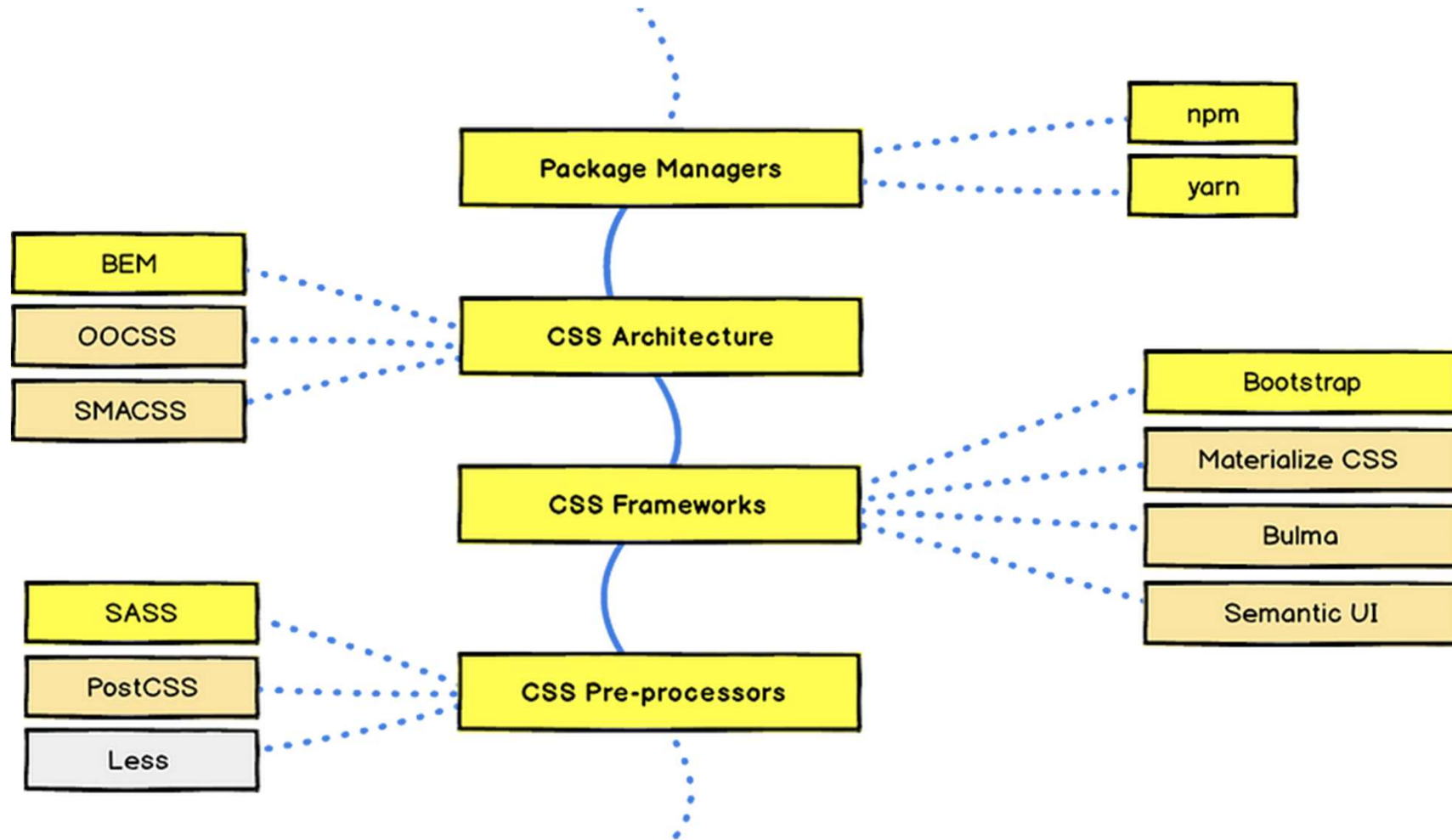
Let's focus on...



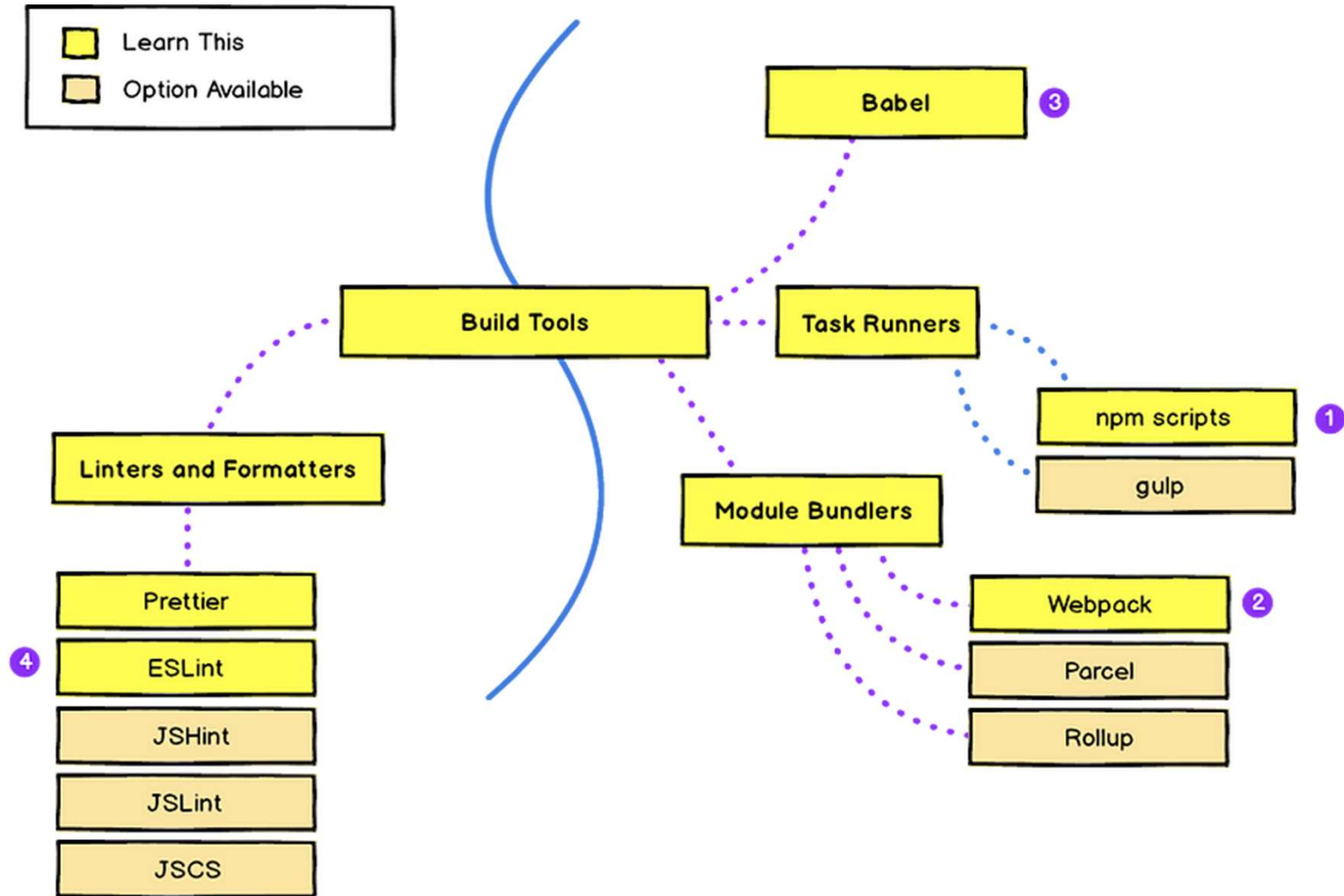
Let's focus on...(Cont.)



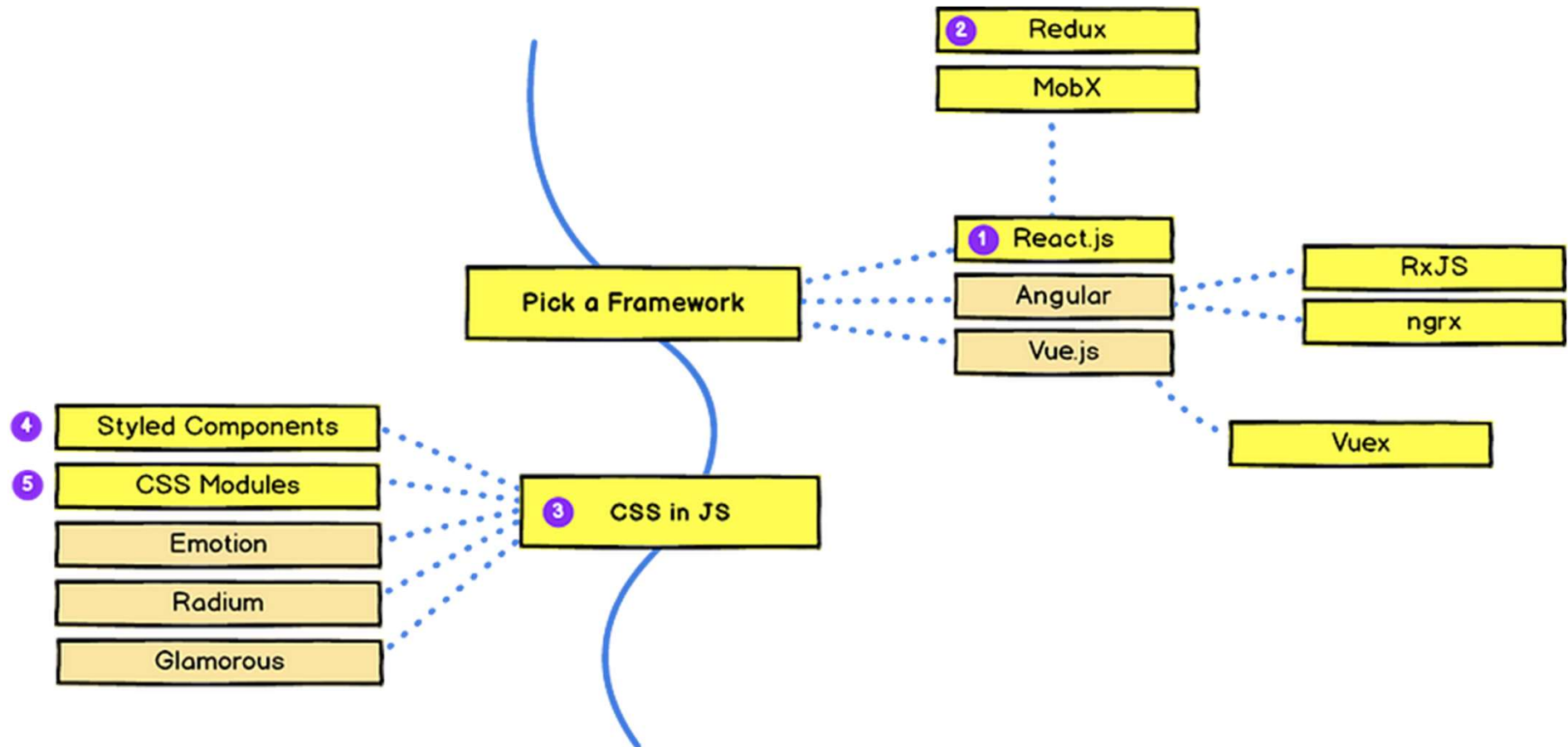
Let's focus on...(Cont.)



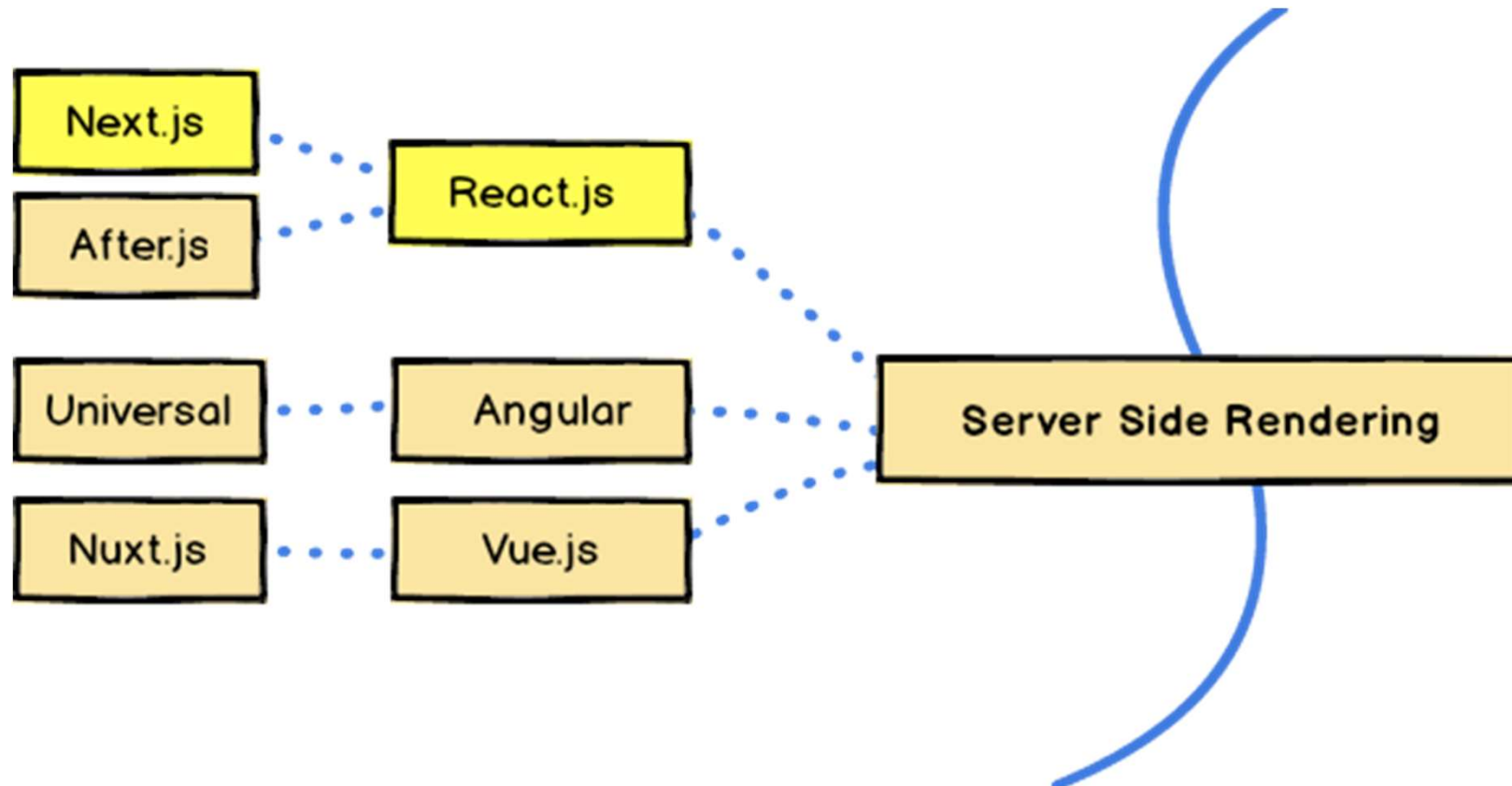
Let's focus on...(Cont.)



Let's focus on...(Cont.)



...Let's focus on...(Cont.)



SPA

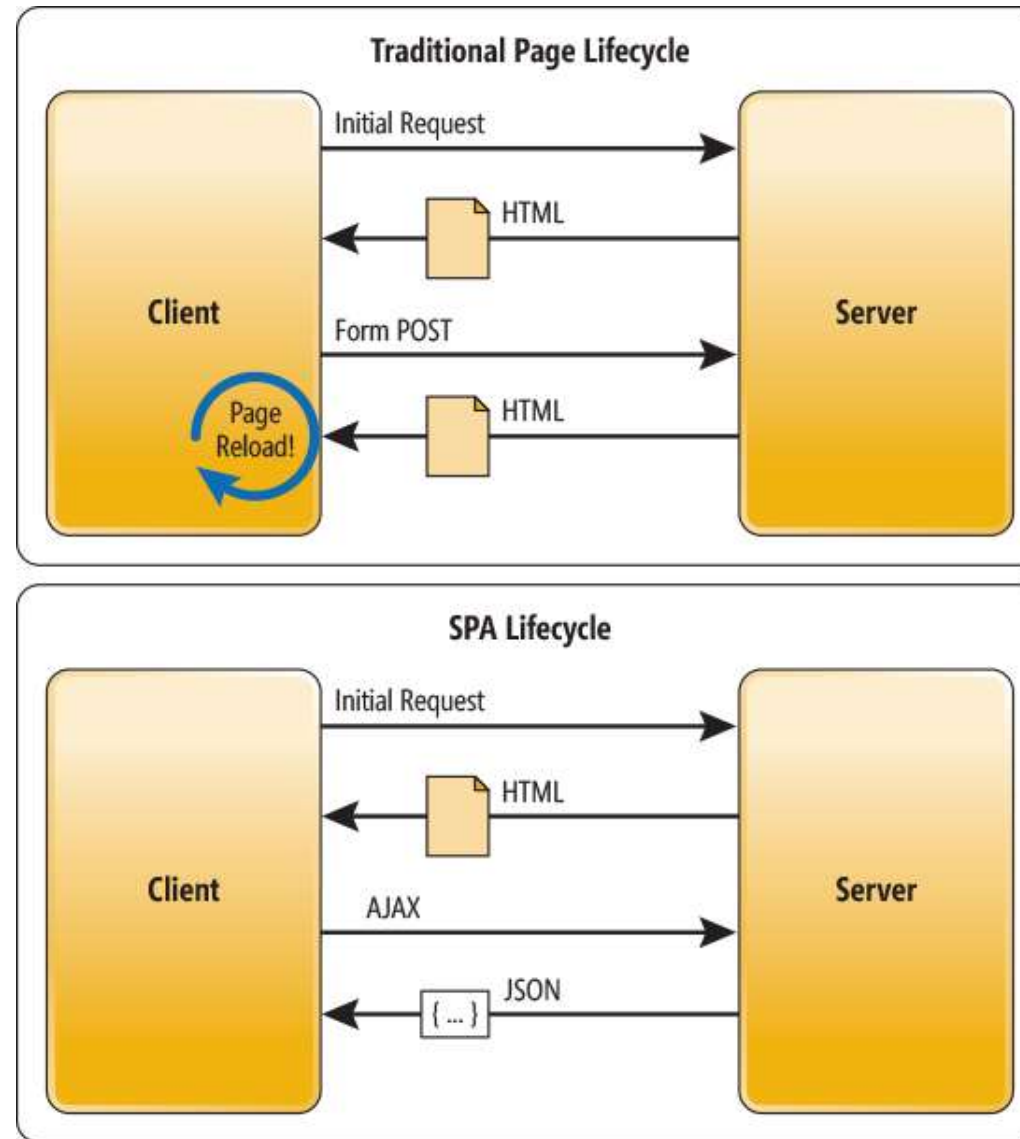
- Single Page Application
- Is a web application or web site that **interacts** with the user by **dynamically rewriting** the current page rather than loading entire new pages from a server.
- Ref <https://poiemaweb.com/js-spa>
- Ref <https://velog.io/@josworks27/SPA-%EA%B0%9C%EB%85%90>



Demo : SPA



SPA

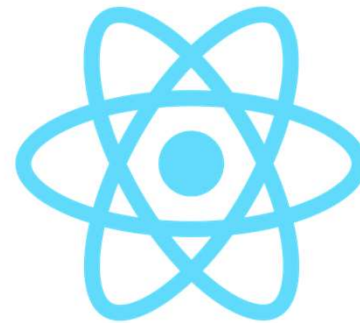


from : <https://linked2ev.github.io/devlog/2018/08/01/WEB-What-is-SPA/>

SPA (Cont.)

■ JavaScript frameworks

- [AngularJS](#)
- [Ember.js](#)
- [ExtJS](#)
- [Knockout.js](#)
- [Meteor.js](#)
- [React](#)
- [Vue.js](#)



SPA (Cont.)

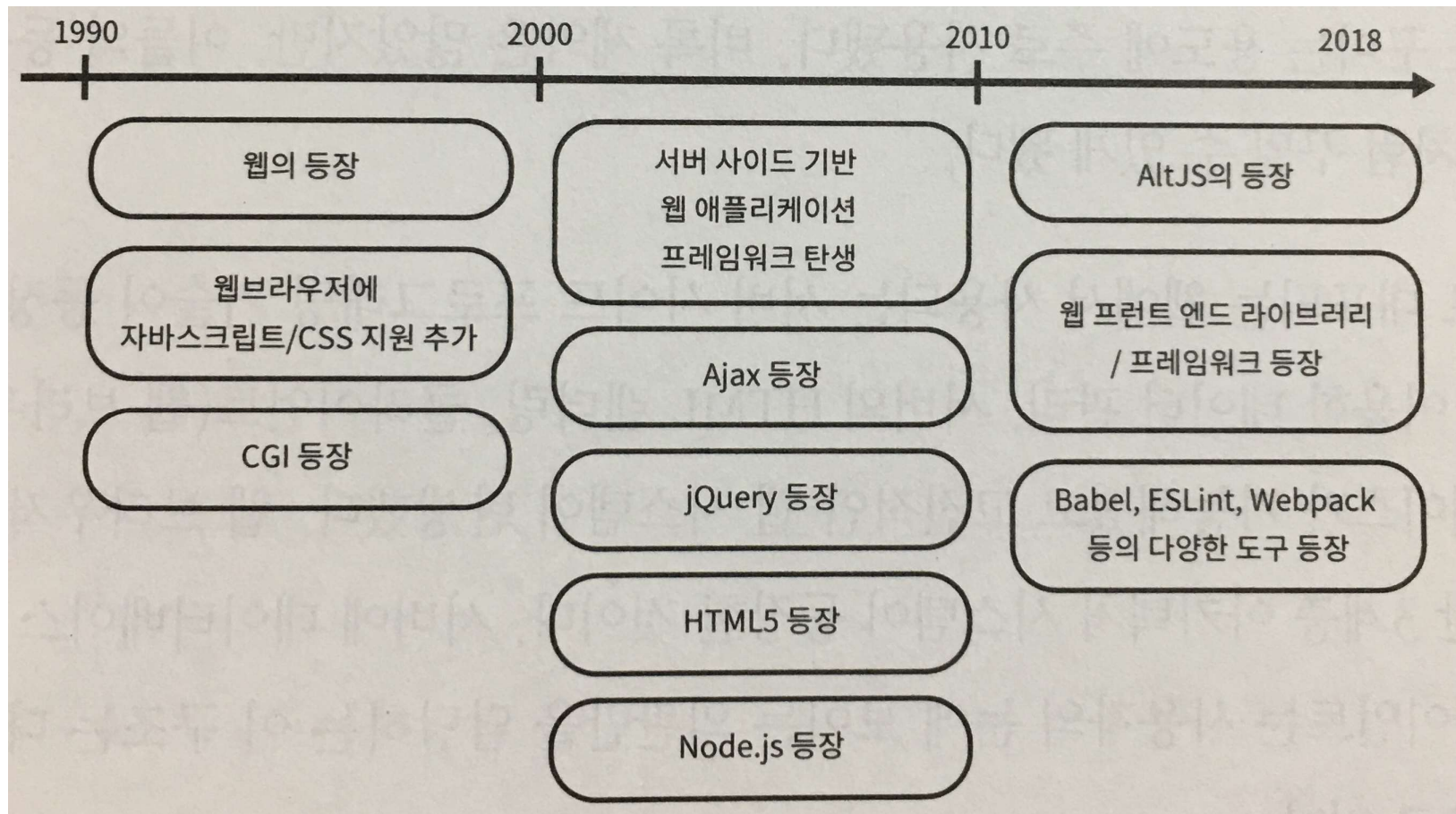
- Ajax
 - XMLHttpRequest, jQuery
- WebSockets
- Server-send events(SSE)
- Browser plugins
 - Silverlight, Flash, Java applets
- Data transport
 - XML, JSON

SPA (Cont.)

- Web Browser 및 Browser Rendering Engine 내부에서 동작하는 Web Application.
- Web Browser에서 최초 접근한 URL을 기점으로 하여 다양한 화면으로의 이동을 제공하지만 기본적으로 최초의 HTML 안에서 User Interface가 완결됨.
- Page 내의 User Interface 변화에 따라 URL이 순차적으로 변화하며, Browser History를 통해 앞의 Page로 거슬러 올라갈 수 있음.
- Page에서 필요로 하는 Data는 Server로부터 API 등의 형태로 필요할 때마다 단편적으로 제공됨.

Front-end 구현 기술의 최신 동향

■ Web Front-end History



Front-end 구현 기술의 최신 동향 (Cont.)

■ HTML5

- 2014년 Recommendation
- Web Application의 Platform化
- History API(Page 이동을 JavaScript로 Handling)를 통한 화면 이동 없이 SPA 가능.
- Presentation Layer의 Program들이 Server-side에서 Client-side로 이동
- Client-side에서 HTML Rendering 가능

Front-end 구현 기술의 최신 동향 (Cont.)

■ Node.js와 JavaScript Ecosystem의 진화

- 2009년 Node.js 등장
- JavaScript 실행환경이 Browser를 벗어났다.
- Front-end 개발과 Test에 매우 유용한 환경 제공
- NPM의 보급
- Module 개발 방법론 제시
- NPM을 통한 Module 배포
- NPM이전에는 JavaScript에 중앙 Package Repository가 없었다.
- 다양한 라이브러리(DOM 조작 Utility Library, JavaScript로 구현된 Application Build, Bundling Tools, Bundle 크기를 줄여주는 Tool, Application Test Library, Library 실행 환경, 정적 분석 도구, Compiler 등) 등장.

Front-end 구현 기술의 최신 동향 (Cont.)

- ES2015와 Programming Language로서의 진화
 - JavaScript 역사상 가장 큰 규모의 Update
 - JavaScript 문법 확장
 - Babel은 JavaScript를 JavaScript로 번역하는 Compiler.
 - Trans File(차세대 JavaScript 문법의 Script Code를 ES6가 지원되지 않는 Browser에서 사용할 수 있도록 변환하는 파일)

Front-end 구현 기술의 최신 동향 (Cont.)

■ React 등 Front-end Library의 등장

- Front-end와 관련된 규격과 기술의 고도화
- Application Dataflow를 Front-end로 이동
- 설계단계부터 개발 난이도 상승
- jQuery를 이용한 Application 구조화의 어려움
- MVC 같은 Application 구조를 지원하는 Framework 필요
- Backbone.js, AngularJS등 새로운 Web Application Framework와 Library 출현
- Facebook의 React(2013년, View Library)와 Flux(2014년, Application Architecture, 지금은 Redux라는 Architecture 겸 Library로 발전)의 등장
- React를 통해 가상 DOM을 이용한 빠른 DOM 조작
- JSX(React에서 사용하는 Template 문법), Dataflow에 대한 지식, Library 선택 및 학습 등 증가된 학습비용

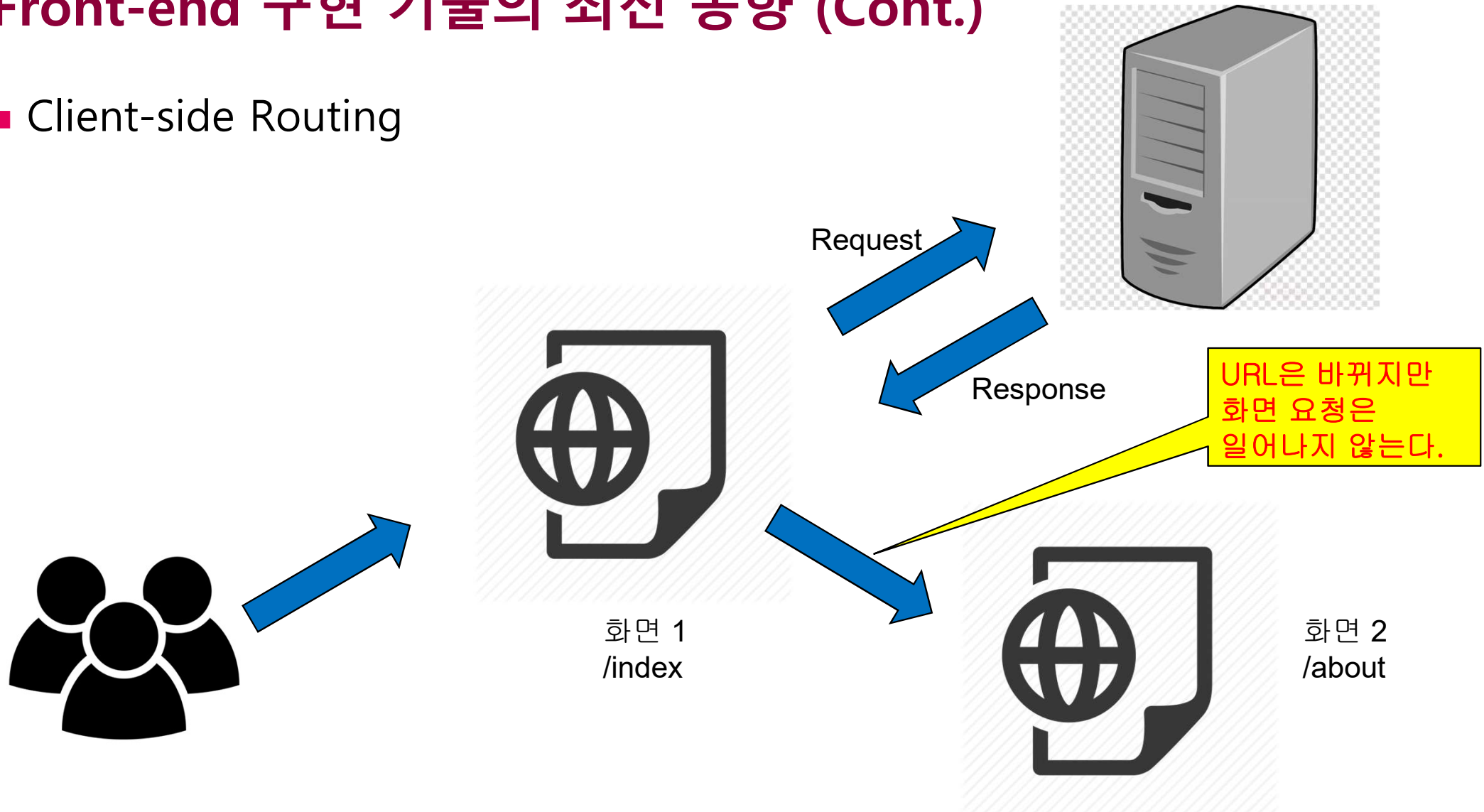
Front-end 구현 기술의 최신 동향 (Cont.)

■ Client-side Routing

- 일반적으로 Routing은 사용자의 Request가 들어온 URL에 대해 적절한 Response(HTML or JSON)을 돌려주기 위해 어떤 처리와 연결할 것인지를 결정하는 과정이다.
- 최근에는 Client 즉 Front-end에서도 Routing이라는 용어가 사용된다.
- SPA는 하나 이상의 화면 상태로 구성된다.
- 그러나 일반적인 Web Application과 달리 URL 단위로 Server에서 HTML Response를 받아오지 않는다.

Front-end 구현 기술의 최신 동향 (Cont.)

■ Client-side Routing



Front-end 구현 기술의 최신 동향 (Cont.)

■ Client-side Routing

- 이 경우 Browser의 이전 Button이나 Bookmark를 사용할 수 없고
- 중간 상태를 공유할 수 없기 때문에 User가 곤란해 진다.
- 따라서, Client 쪽에서도 URL마다 화면을 사전에 생성했거나 HTML5의 History API로 이전 Button을 사용할 수 있도록 해야 한다.
- 이런 내용들이 Server-side Routing과 구분하기 위해서 Client-side Routing이라고 한다.

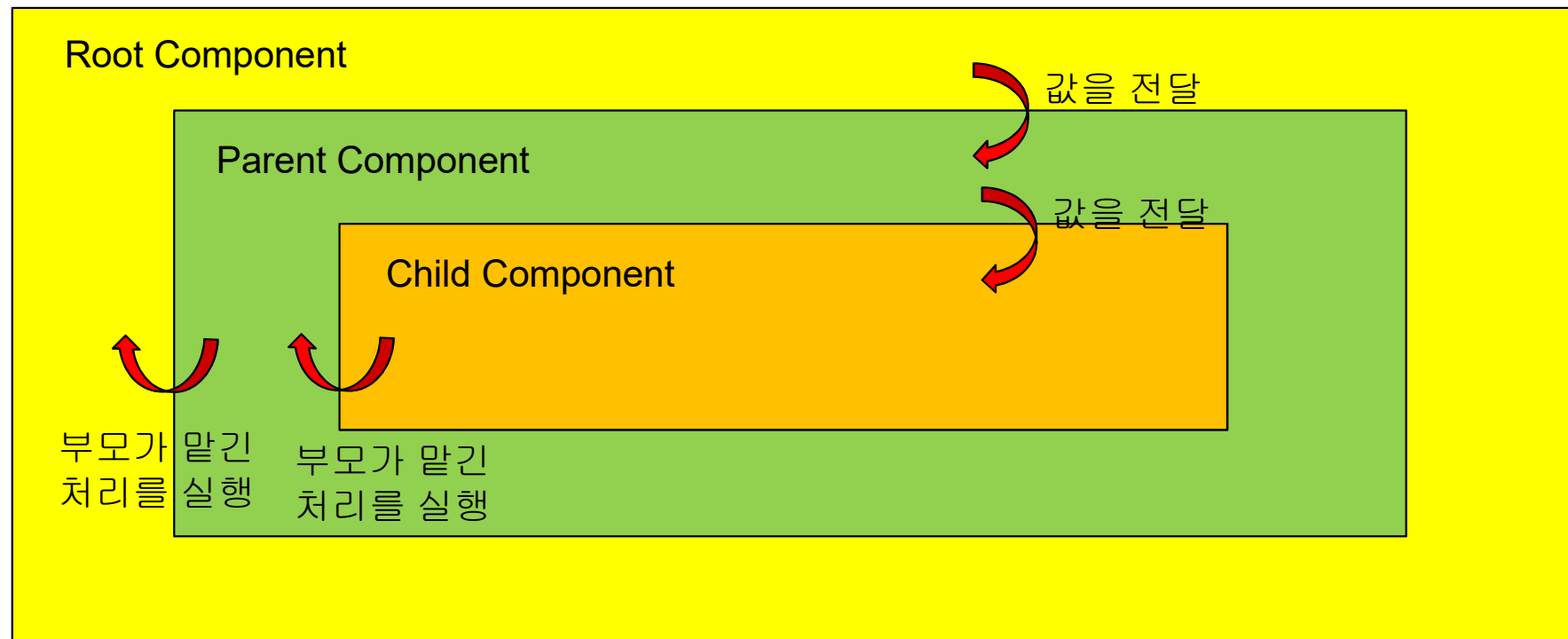
Front-end 구현 기술의 최신 동향 (Cont.)

■ Component-oriented

- Component와 Component간의 상호 작용 형태로 프로그램을 작성
- 만일, Radio Button이 Check되면 Send Button이 Click할 수 있는 상태가 되는 동작은 두 Component 사이의 상호작용이 있어야 가능하다.
- Component가 주체가 되어 Component 사이의 관계로 Application 전체를 구성하는 것
- Component-Oriented에서는 Application의 기본 정보를 포함하는 page를 통째로 하나의 Component(Root Component)로 정의한다.
- 이 Root Component 안에 다른 Component를 포함시키는 형태로 Component간의 부모 관계 및 형제 관계를 형성하고 Component 사이의 상호작용을 정의하여 Application을 제작한다.

Front-end 구현 기술의 최신 동향 (Cont.)

■ Component-oriented



Front-end 구현 기술의 최신 동향 (Cont.)

■ Component-oriented

- 구현시 Component 각각이 단독으로 필요한 기능을 수행할 수 있도록 Style과 Script를 포함하는 독립된 존재 즉 Capsulation이 필요하다.
- React에서는 Component에 필요한 CSS, Template, JavaScript를 File 하나에 기술한다.
- CSS는 Component 내부에만 적용되도록 범위가 제한된다.
- 이렇게 구현된 이유는 Framework 설계에 Component-Oriented가 반영되어 있기 때문이다.

Front-end 구현 기술의 최신 동향 (Cont.)

■ SSR과 Pre-rendering

- SSR(Server-Side Rendering)
- 일반적인 Web에서는 Browser는 Server에서 받아오는 HTML은 이미 Rendering이 끝난 HTML이다.
- 이와 달리 SPA에서는 비어 있는 HTML을 응답으로 받는다.

`<div id="app"></div>`

- 위의 code에서는 #app 안에 SPA의 Contents가 삽입되어 HTML이 동적으로 생성된다.
- 그래서 Client-side에서 Rendering이 이루어진다고 표현하는 것이다.
- 하지만, 현실적으로 Client-side에서 Rendering 할 때 문제점은 없는가?
- Page 표시 속도와 Crawler가 대응하지 못한다는 문제가 발생한다.

Front-end 구현 기술의 최신 동향 (Cont.)

■ SSR과 Pre-rendering

- Page 속도를 생각해보자.
 - 빈 HTML이 먼저 표시되고 그 후에 Contents가 Rendering되기 때문에 전체 HTML이 바로 표시되는 경우에 비해 속도가 느려진다.
 - 만일, Contents가 API를 경유하여 전달된다면, 더더욱 속도가 느려 질 것이다.
- 두번째, Crawler에서의 2가지 문제다.
 - SEO(Search Engine Optimization) → Page가 검색 결과에 잘 포함될 것인가?
 - SNS에 대한 문제 → 정상적으로 Web page가 공유될 것인가?
 - wget이나 curl같은 명령행 도구로 URL에 접근하는 것과 마찬가지로 JavaScript가 실행되기 전에 page가 수집되는 문제 발생
 - 이유는 Server로부터 들어오는 HTML은 Contents가 들어갈 자리가 비어있는 div 요소가 있기 때문이다.
- SPA의 약점은 기계 가독성이 낮다는 점.
- 그래서 Crawler의 유형과 상관없이 정상적으로 Contents를 제공하려면 일반적인 Web site와 동등한 정보를 HTTP Response에 넣어서 보낼 수 있는 방법이 필요.

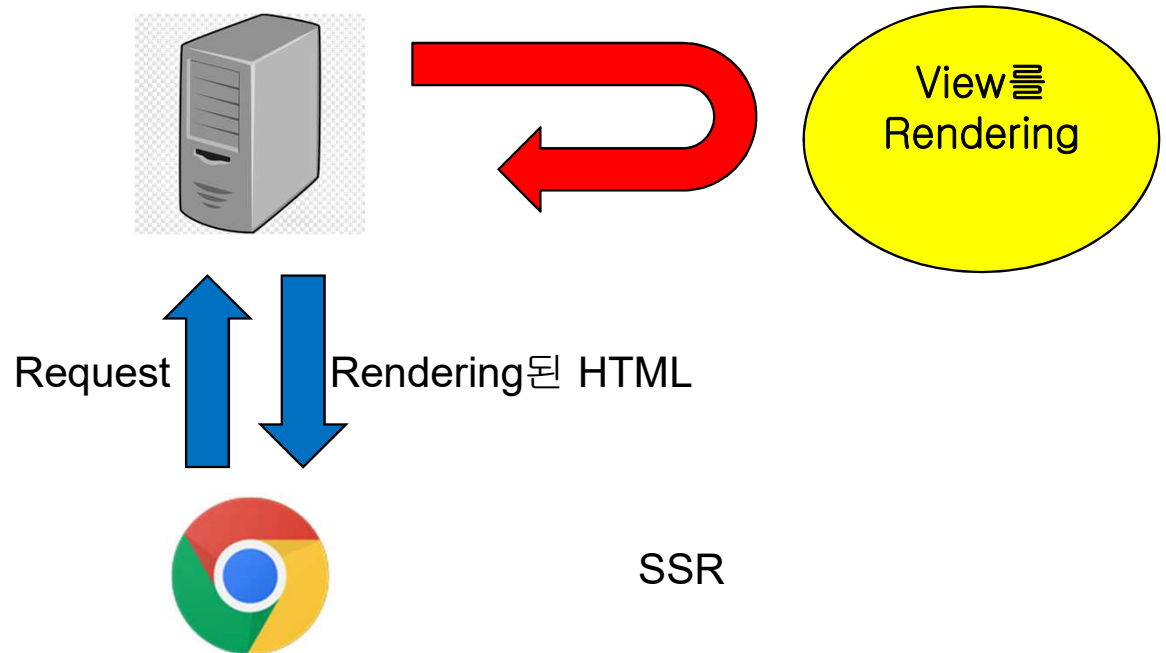
Front-end 구현 기술의 최신 동향 (Cont.)

■ SSR과 Pre-rendering

- 해결점1 – SSR(Server-side Rendering)

- Server에서 Client로 전달되는 Response 결과에 이미 Contents까지 Rendering을 마친 상태를 전달한다.
- 이를 구현하려면 어떤 방법이든지 Server-side에서 SPA에서 만들 Rendering 결과를 완성해야 한다.

- 해결점2 – Pre-rendering



Front-end 구현 기술의 최신 동향 (Cont.)

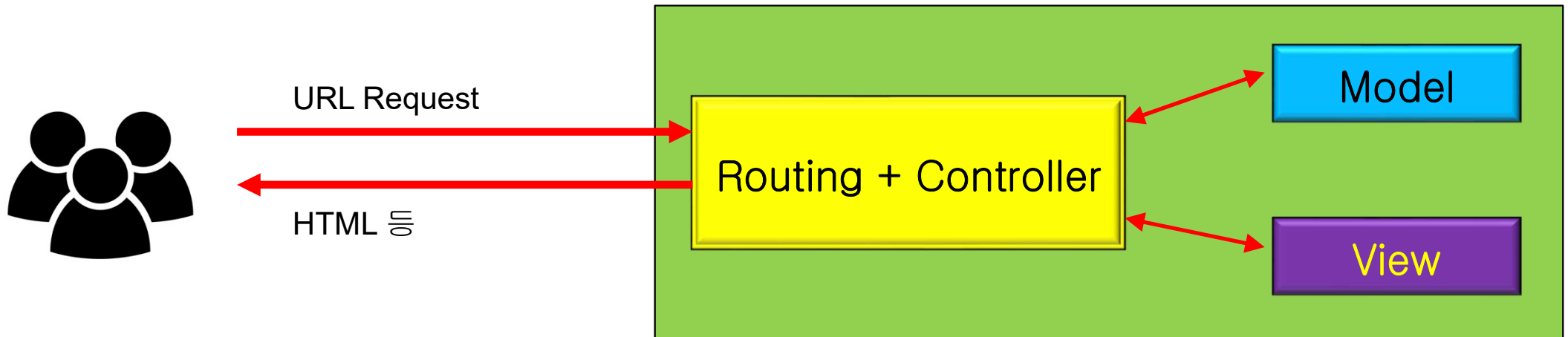
■ Virtual DOM

- JavaScript가 HTML을 Rendering 하는 방법 중 하나.
- 실제 DOM을 수정하는 과정은 JavaScript에서 Memory상의 객체를 수정하는 것보다 시간이 많이 소요된다.
- 따라서 일단 Memory 상에 있는 DOM 구조를 Update한 다음, 가상 DOM의 현재 상태와 이전 상태의 차이를 구해서 그 차이만을 실제 DOM에 반영하여 효율적으로 DOM을 수정하는 것이다.
- 가상 DOM을 도입하면 성능 향상 뿐만 아니라, 어떤 Data를 인자로 넘기면 HTML 구조 전체를 반환하는 함수를 통해 나머지는 가상 DOM이 어디를 수정할지 관리하고 직접 수정까지 가능하다.

Front-end 구현 기술의 최신 동향 (Cont.)

■ Web Application의 MVC

- Model-View-Controller
- Model은 Data와 Data에 접근하기 위한 기능
- View는 Contents를 어떻게 외부에 출력할지를 정의하는 부분, 사용자에게 전달되는 형태는 HTML일 수도 있고, JSON일 수도 있으며, 다른 Format일 수도 있다.
- Controller는 특정 URL을 요청받았을 때(Routing), 필요에 따라 Controller가 Model과 정보를 주고받은 다음 적절한 View를 골라 사용자에게 전달한다.



Front-end 구현 기술의 최신 동향 (Cont.)

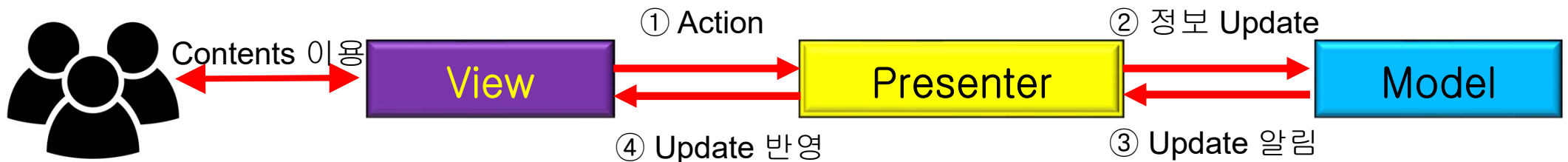
■ Front-end Application의 MV

- Front-end Application에서 MV(Model-View)는 MVC Application의 View 부분에 해당한다.
- Web Application은 본래 Contents가 HTML에 포함된 상태로 User에게 Response 해서 전달되기 때문에 View안에 MV와 같은 구성을 할 필요가 없었다.
- 그러나, SPA에서는 View안에도 Ajax등을 통한 Data 접근 계층이 추가되거나 필요에 따라 URL을 교체하는 처리 등이 포함되기 때문에 MV 구조가 필요하게 된다.

Front-end 구현 기술의 최신 동향 (Cont.)

■ MVP Pattern

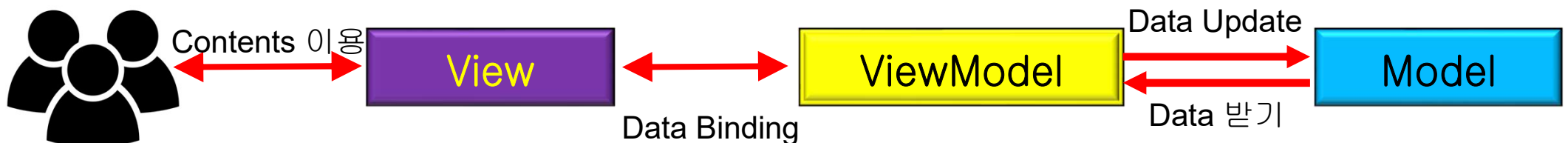
- Model-View-Presenter
- MV는 Model과 View이다.
- 여기에 P(Presenter)가 추가된 Pattern이다.
- Model과 View 사이에 Presenter가 위치하여 양자 간 입출력 Interface를 담당한다.
- User가 Model의 정보를 수정하거나 읽으려면 반드시 View를 거쳐서 Presenter가 제공하는 Interface를 통해야 한다.



Front-end 구현 기술의 최신 동향 (Cont.)

■ MVVM Pattern

- Model-View-ViewModel
- ViewModel은 Model과 View 사이에 위치한다는 점에서 Presenter와 비슷하지만, 그 역할이 View와 Model간의 Interface 대신 양방향 Data Binding을 담당한다는 점이 다르다.
- 여기서 양방향이란 View를 통해 변경하려는 Model의 값이 ViewModel을 거쳐 변경 내용을 탐지하고 Model에 변경 내용이 반영된다는 뜻이다.





Demo : Data Bindings



Front-end 구현 기술의 최신 동향 (Cont.)

■ PWA

- Progressive Web Apps
- Google
- <https://developers.google.com/web/progressive-web-apps>
- 여기서 Progressive란 단계적으로 적용이 가능하며, 최신 Browser에는 최신 기술을 적용하고, 그렇지 못한 환경에서도 Contents를 열람할 수 있도록 하는 Progressive Enhancement(단계적 개선)의 사상을 말한다.
- Service Worker를 통한 Contents 제어
- Application Shell(App Shell)을 이용한 Application化
- Web Notification과 Web Push를 이용한 Push Notification

Web Tech Terminology

■ Application State

- 전역으로 관리되는 Application의 상태
- Application의 어떠한 Component들도 이 상태에 접근할 수 있다.
- 그러나 쉽게 상태를 바꾸도록 허용되지는 않는다.
- 각 상태들은 Application의 Component 내부에서 특정 Event가 발생했을 때 이를 처리하는 변이들과 연결된다.

■ Bootstrap

- CSS에 대해 깊이 알지는 못해도 멋진 반응형 웹을 개발할 수 있도록 JavaScript 도구들과 관련 Style들을 제공

Web Tech Terminology (Cont.)

■ CDN

- Content Delivery Network
- 사용자에게 Data 고가용성, 고속으로 제공하는 데 특화된 Server를 말한다.
- Framework를 개발한 사람들이나 회사는 주로 CDN을 통해 배포하는데, CDN의 URL만 명시하는 방법으로 설치를 유도할 수 있기 때문이다.

■ Component

- 자체 Data를 가지고 재사용될 수 있는 Application의 구성 요소
- 집을 짓는 데 벽돌 같은 역할을 한다.

■ CSS

- Cascading Style Sheet
- HTML 문서에 적용하면 해당 문서를 깔끔하고 아름답게 만들어주는 Style들의 모음

Web Tech Terminology (Cont.)

■ 선언전인 View

- 일반 JavaScript 객체와 표현 간의 직접적인 Data Binding을 제공하는 View의 한 종류.

■ Directives

- React의 특별한 HTML Element로 다른 방식의 Data Binding을 지원

■ DOM

- Document Object Model
- HTML, XML, XHTML 같은 Markup 언어에서 Node들을 표현하는 방법
- 문서의 Node들은 DOM Tree 형태로 구성된다.
- DOM 작업을 한다는 것은 곧 HTML 요소들을 다룬다는 것

Web Tech Terminology (Cont.)

■ npm

- JavaScript의 Package 관리자로 JavaScript Package 검색, 설치, 관리를 도와줌.

■ Markdown

- Web에서 글을 작성할 때 사람이 읽기 쉬운 문법을 사용해서 HTML 문법과 Style에 구애받지 않도록 해준다.
- .md 확장자

■ MVVM

- Architecture의 한 종류로 View와 Data Model 사이의 Data 흐름을 다루는 중계자로서 역할하는 ViewModel이 중심이 된다.

■ One-way Data Binding

- Data Model의 변경이 View layer로 자동 전파되는 Data Binding 방식을 말한다.
- 반대 방향으로의 변경은 지원되지 않는다.

Web Tech Terminology (Cont.)

- Rapid Prototyping
 - Web에서 기본적인 Action들만 포함하는 UI의 Mockup을 빠르게 작성하는 것
- Reactivity
 - Web에서 Data Model에 변화가 일어났을 때 즉각적으로 View에 반영되는 것
- Two-way Data Binding
 - Data Model의 변화가 자동적으로 View layer에 전파되고 View layer의 변화도 Data Model에 반영되는 것
- UI
 - 사용자가 Application과 상호작용할 수 있게 해주는 시각적인 Component들.

화면 개발 방식의 변화

과거



현재



패키지 관리 도구



최신
자바스크립트 문법



모듈 번들러



타입에 엄격한
자바스크립트

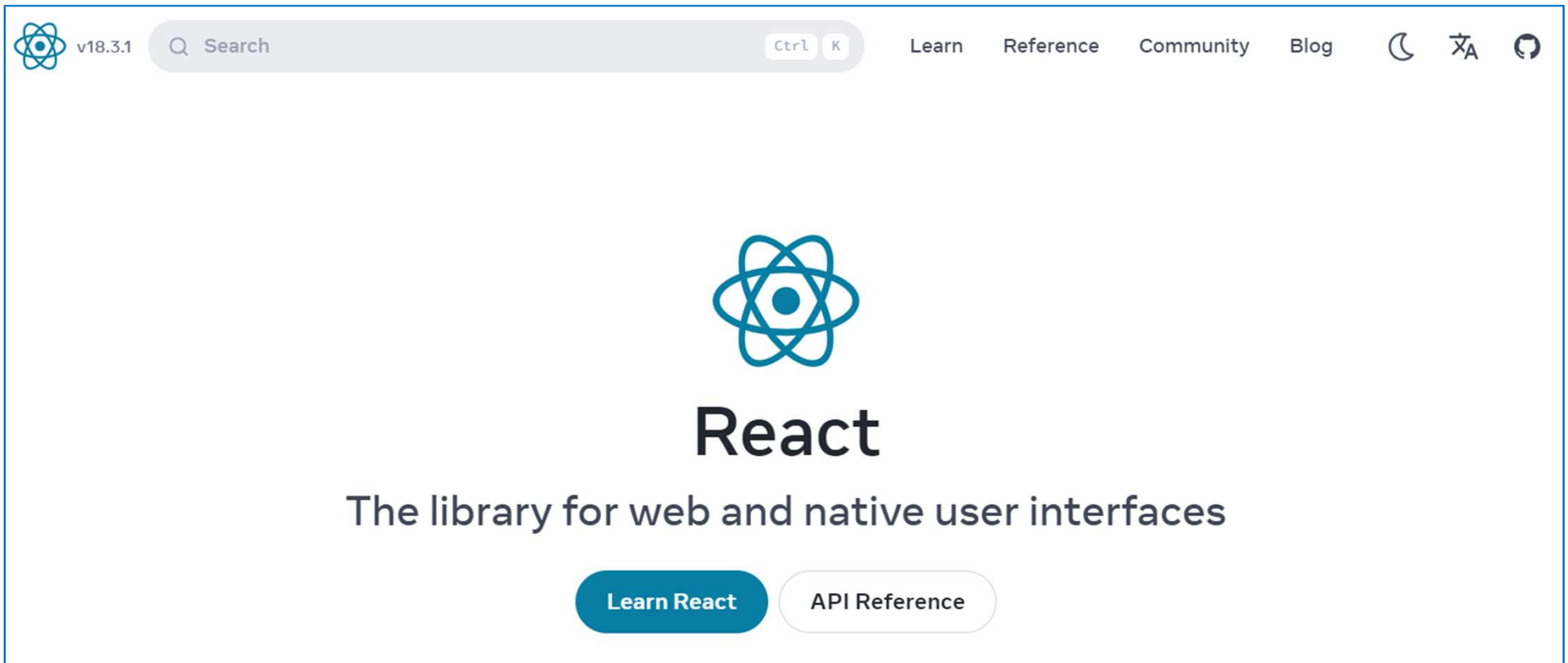
현재 당면 과제와 React

- HTML5 이후 Web이 Application Platform으로 가능하게 되면서 API가 고도화됨.
- Node.js Ecosystem의 발전과 개발 환경 구축의 난이도 증가
- ES2015 이후 문법이 보강되면서 학습할 내용 증가
- React 이후 Front-end 개발이 Framework化 되면서 그에 따른 학습 비용 증가

시기	Front-end의 역할	Server 역할	JavaScript Library / Framework
Web System 시대	외관 꾸미기	HTML 생성	N/A
Ajax 시대	Ajax 중심의 Interaction	HTML 생성 + API	jQuery, Prototype.js
현재	Application의 Presentation 전반	API	Vue.js, React, Angular

React


- <https://react.dev/>



React (Cont.)

- <https://ko.legacy.reactjs.org/>

This site is no longer updated. [Go to react.dev](#)

 **React** 문서 자습서 블로그 커뮤니티 🔍 검색 v18.2.0 🌐 Languages GitHub

React

사용자 인터페이스를 만들기 위한 JavaScript 라이브러리

[시작하기](#) [자습서 읽어보기 >](#)

선언형

React는 상호작용이 많은 UI를 만들 때 생기는 어려움을 줄여줍니다. 애플리케이션의 각 상태에 대한 간단한 뷰만 설계하세요. 그럼 React는 데이터가 변경됨에 따라 적절한

컴포넌트 기반

스스로 상태를 관리하는 캡슐화된 컴포넌트를 만드세요. 그리고 이를 조합해 복잡한 UI를 만들어보세요.

한 번 배워서 어디서나 사용하기

기술 스택의 나머지 부분에는 관여하지 않기 때문에, 기존 코드를 다시 작성하지 않고도 React의 새로운 기능을 이용해 개발할 수 있습니다.

React (Cont.)

- "The library for web and native user interfaces"
- Facebook, *Jordan Walke*, 2011년 Facebook 뉴스피드 시작
- JSConf 2013, 오픈소스로 발표
- UI(MV의 *View*)를 빠르게 개발하기 위한 Framework(Library).
- 즉 Web page 화면을 개발하기 위한 Front-end Framework
- 그래서 재사용 가능한 Component를 지원하는 동시에 Reactive한 Data Binding을 쉽고 유연성 있게 제공해서 빠르게 Prototype을 만들 수 있는 Library를 만들게 됨.

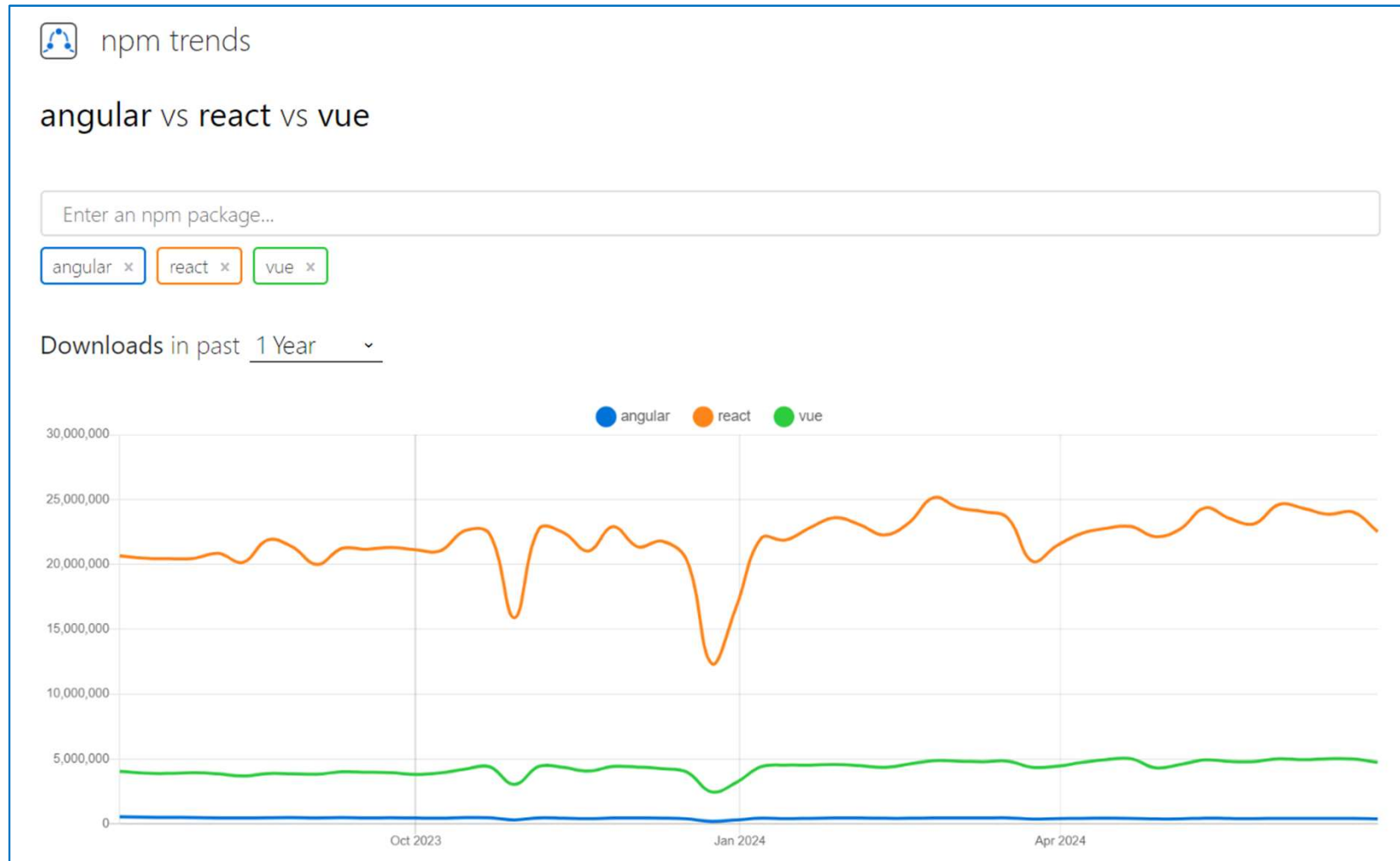
React (Cont.)



React (Cont.)



React (Cont.)



<https://npmtrends.com/angular-vs-react-vs-vue>

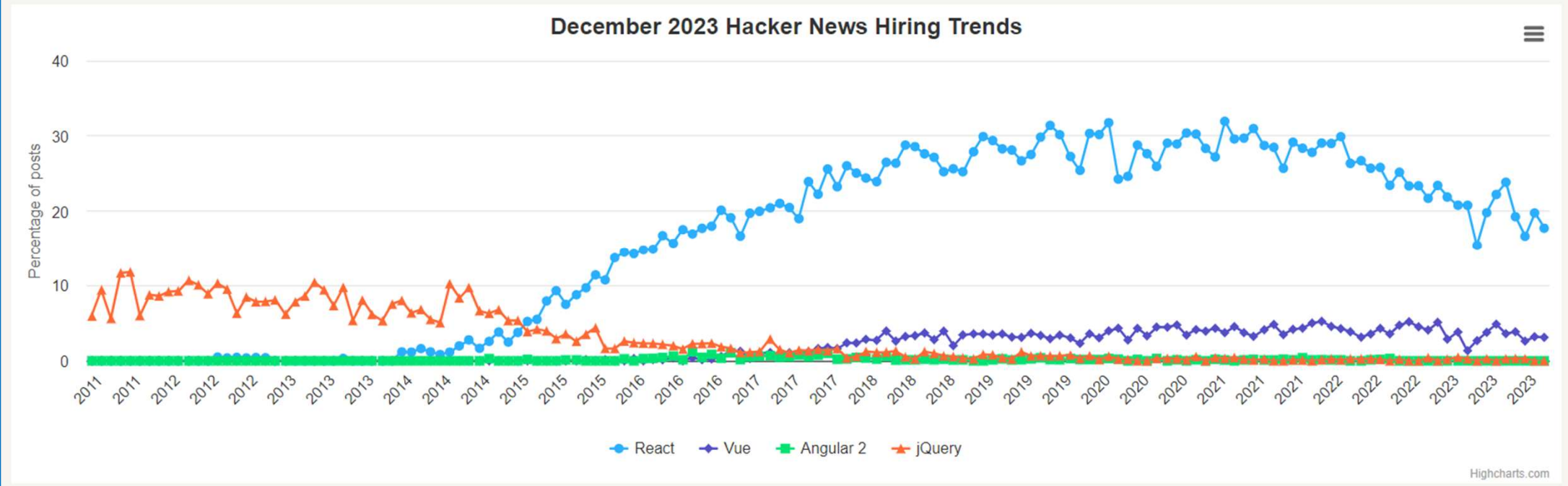
React (Cont.)

[Home and Archives](#) / [December 2023](#) / [Highlights and Analysis](#) / [Data](#)

Tough year for tech hiring comes to a close

December 2023 Hacker News Hiring Trends

Top 5 ▾ or








<https://www.hntrends.com/2023/december.html?compare=Angular+2&compare=jQuery&compare=React&compare=Vue>

React (Cont.)

Front-end Frameworks

2023 ▾ ⏪ ⏩

1		React The library for web and native user interfaces.	+16.9k ☆
2		htmx Access AJAX, WebSockets and Server Sent Events directly in HTML	+15.6k ☆
3		Svelte Cybernetically enhanced web apps	+10.3k ☆
4		Million <1KB Virtual DOM Implementation	+8.2k ☆
5		Vue.js A progressive, incrementally-adoptable JavaScript framework for b...	+7.9k ☆

React

In 2023, React maintains its position as a frontrunner in the JavaScript ecosystem, a trend persisting from 2021 and 2022, despite occasional criticism and controversies.

Celebrating its 10th anniversary, React earned a documentary (watch it [here](#)); it says a lot about its popularity!

However, React is not without its skeptics. Some view it as a relic of the past, a sentiment expertly explored in [this article](#), which also delves into alternative solutions. On the flip side, proponents laud React for its remarkable adaptability and capacity for reinvention.




One point of contention revolves around the absence of "signals", a reactivity handling mechanism employed by libraries such as [Solid](#) or [Preact](#), while React enthusiasts appreciate the predictability offered by React's data flow.


From [a tweet](#) by Devon Govett, the creator of Parcel, about signals:

<https://risingstack.com/react-a-front-end-framework-that-is-not-dying/>



React (Cont.)

bestofjs[Home](#)[Projects](#)[Tags](#)[Monthly](#)[More ▾](#)Ctrl K







 **All Tags**

Sort: By number of projects ▾

Showing 1 - 20 of 236







React 362

Ecosystem related to the `React` library









Vue 117

Ecosystem related to the `Vue.js` library









Component set 94

Collections of styled components for specific UI libraries (React, Vue.js...)



Testing 76

Tools to help with automatic testing: unit tests, smoke tests, end-to-end tests...



React (Cont.)

FRONT-END FRAMEWORKS

Front-end frameworks and libraries

FRONT-END FRAMEWORKS RATIOS OVER TIME

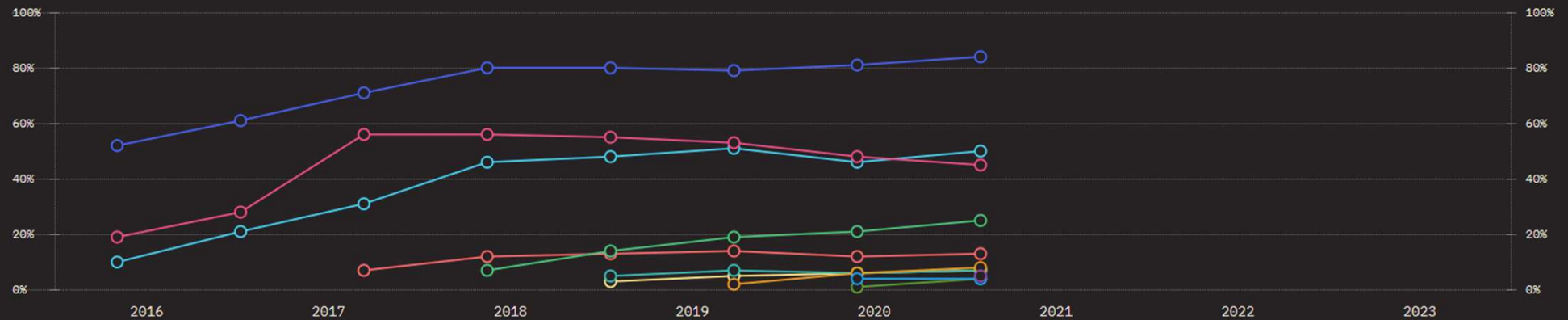
All Respondents

Add Filters...

React Vue.js Angular Preact Svelte Alpine.js Lit Solid Qwik Stencil HTMX

Mode: Value Rank

View: Usage Awareness Interest Retention Positivity



<https://2023.stateofjs.com/en-US/libraries/front-end-frameworks/>

React의 장점

■ Component

- 작고 독립적인 코드 블록을 조합하여 빠르고 효율적으로 화면 구성

■ Virtual DOM

- JavaScript는 화면이 커질수록 화면 Rendering 시간이 길어진다는 단점.
- 화면의 일부분만 수정되어도 화면 전체를 다시 Rendering 한다는 단점.
- 다음에 나타날 화면의 일부(Node)를 미리 그려 놓고 변경된 화면의 일부(Node)만 수정하는 개념 도입⁽¹⁾
- 즉, 실제 DOM과의 차이를 비교하고 **변경된 부분만을 실제 DOM에 반영**함으로써 DOM 조작을 최소한으로 줄임.

■ 한국어 문서 제공

⁽¹⁾ <https://github.com/FEDevelopers/tech.description/wiki/%EA%B0%80%EC%83%81-%EB%8F%94%EA%B3%BC-%EB%8F%94%EC%9D%98-%EC%B0%A8%EC%9D%B4%EC%A0%90>

React의 Key Concept

- 일부 Web Framework가 MVC 또는 MVW 등의 구조를 지향하는 것과 달리 React는 오직 View만 담당한다.
- React는 Framework가 아니라 Library이다.
- 다른 Framework가 Ajax, Data modeling, Routing 등과 같은 기능을 내장하고 있는 반면, React는 정말 View만 신경쓴다.
- 따라서, 기타 기능은 직접 구현하여 사용해야 한다.
 - Routing → react-router
 - Ajax → axios, fetch
 - 상태관리 → redux, MobX

jQuery, Vue.js

jQuery

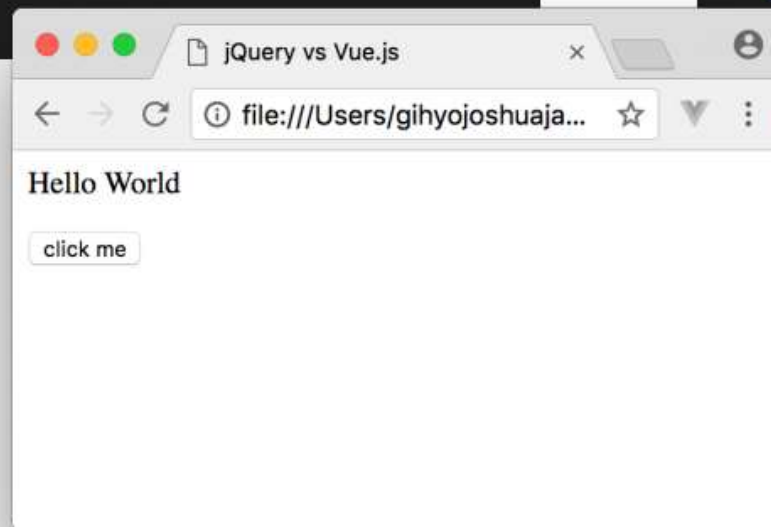
```
<body>
  <div>
    <p></p>
    <button>click me</button>
  </div>
  <script
    src="http://code.jquery.com/jquery-3.3.1.js"></script>
  <script>
    $('p').text('Hello World');
    $('button').click(function() {
      alert('hi');
    });
  </script>
</body>
```

CDN 라이브러리 로딩

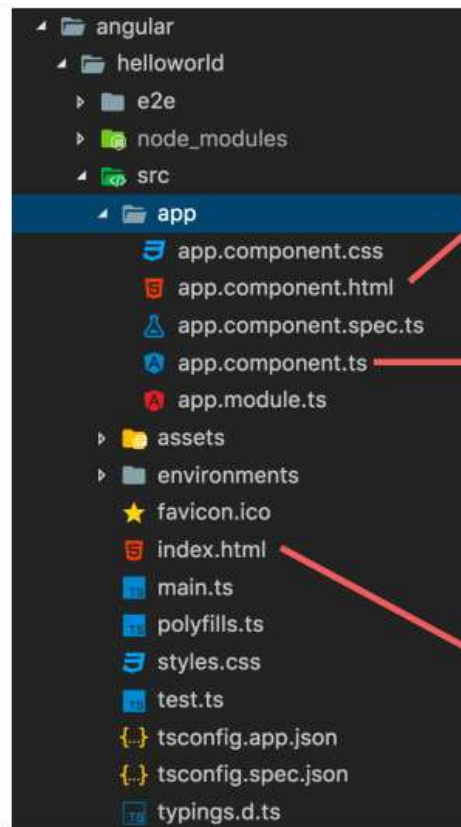
Vue.js

```
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="clickBtn">click me</button>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
<script>
  new Vue({
    el: '#app',
    data: {
      message: 'Hello World'
    },
    methods: {
      clickBtn: function() {
        alert('hi');
      }
    }
  });
</script>
```

CDN 라이브러리 로딩



Angular



app.component.html

```
<div>
  {{ message }}!
</div>
```

app.component.ts

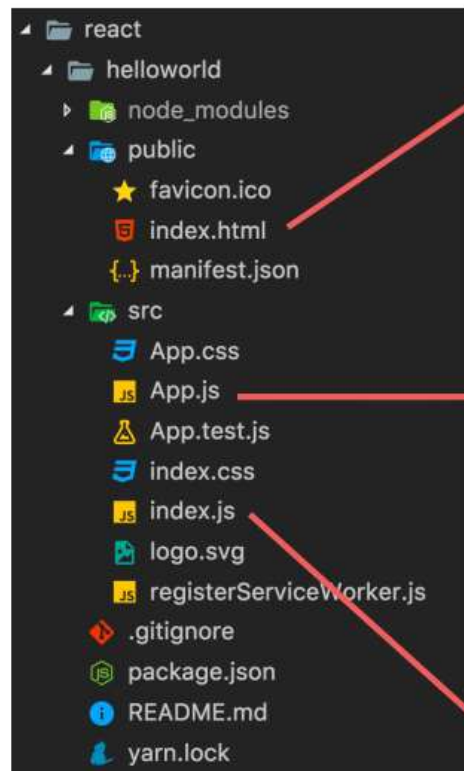
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  message = 'Hello Angular';
}
```

index.html

```
<body>
  <app-root></app-root>
</body>
```

React



index.html

```
<div id="root"></div>
```

App.js

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        {this.props.message}
      </div>
    );
  }
}

export default App;
```

index.js

```
import React from 'react'; 7.1K (gzipped: 3.1K)
import ReactDOM from 'react-dom'; 93.4K (gzipped: 29.8K)
import './index.css';
import App from './App';

ReactDOM.render(<App message="Hello React"/>, document.getElementById('root'));
```

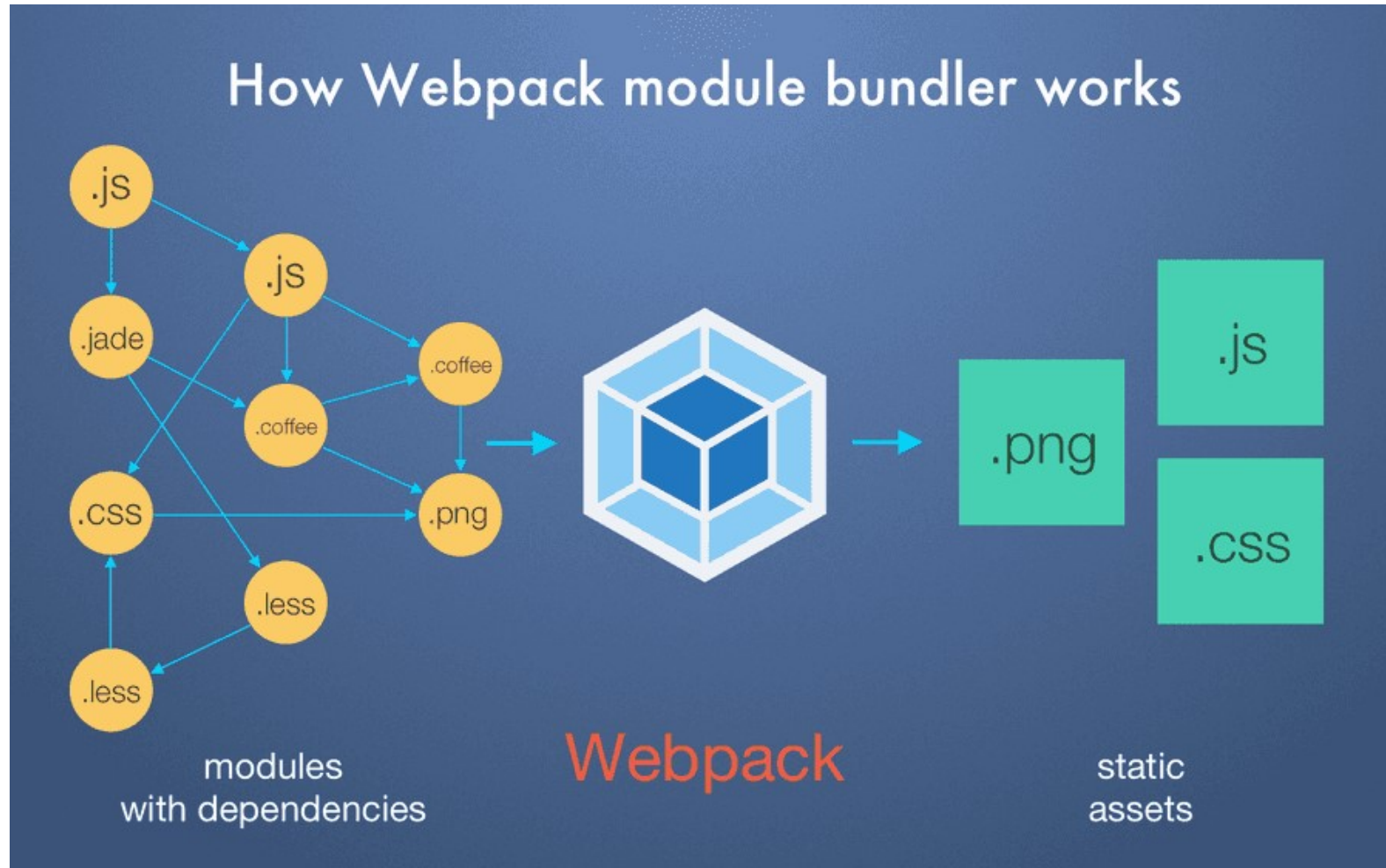
Framework에서 필요한 기술 Stack



Webpack

- Project에 사용된 파일을 분석하여 기존 웹 문서 파일로 변환하는 도구.
- 필요한 이유는 Framework가 .js, .css, .jpg와 같은 기본 웹 문서 파일을 사용하지 않기 때문이다.
- Twitter Bootstrap templat는 웹 문서 스타일을 .css가 아닌 .sass 파일로 작성한다.
- 웹 문서는 .sass 파일을 해석하지 못하기 때문에 중간에 무언가가 이 파일을 해석해야 한다.
- 이 중간자 역할이 바로 webpack이다.
- .js, .png, .jpg 같은 파일을 적절한 크기로 자르거나 묶어주는 역할도 수행.
- 불필요한 파일을 제외하거나 압축하여 프로젝트의 용량을 줄여줌.
- 또한, 간단한 node 기반의 웹 서버를 구동하기도 함.

Webpack (Cont.)



References

- MDN
 - <https://developer.mozilla.org/ko/docs/Web/JavaScript>
- NPM
 - <https://www.npmjs.com/>
- Babel
 - <https://babeljs.io/>
- babel plugin
 - <https://babeljs.io/docs/en/plugins/>
- Axios
 - <https://yamoo9.github.io/axios/>
- React:
 - <https://ko.reactjs.org/>
- Redux
 - <https://ko.redux.js.org/introduction/getting-started>