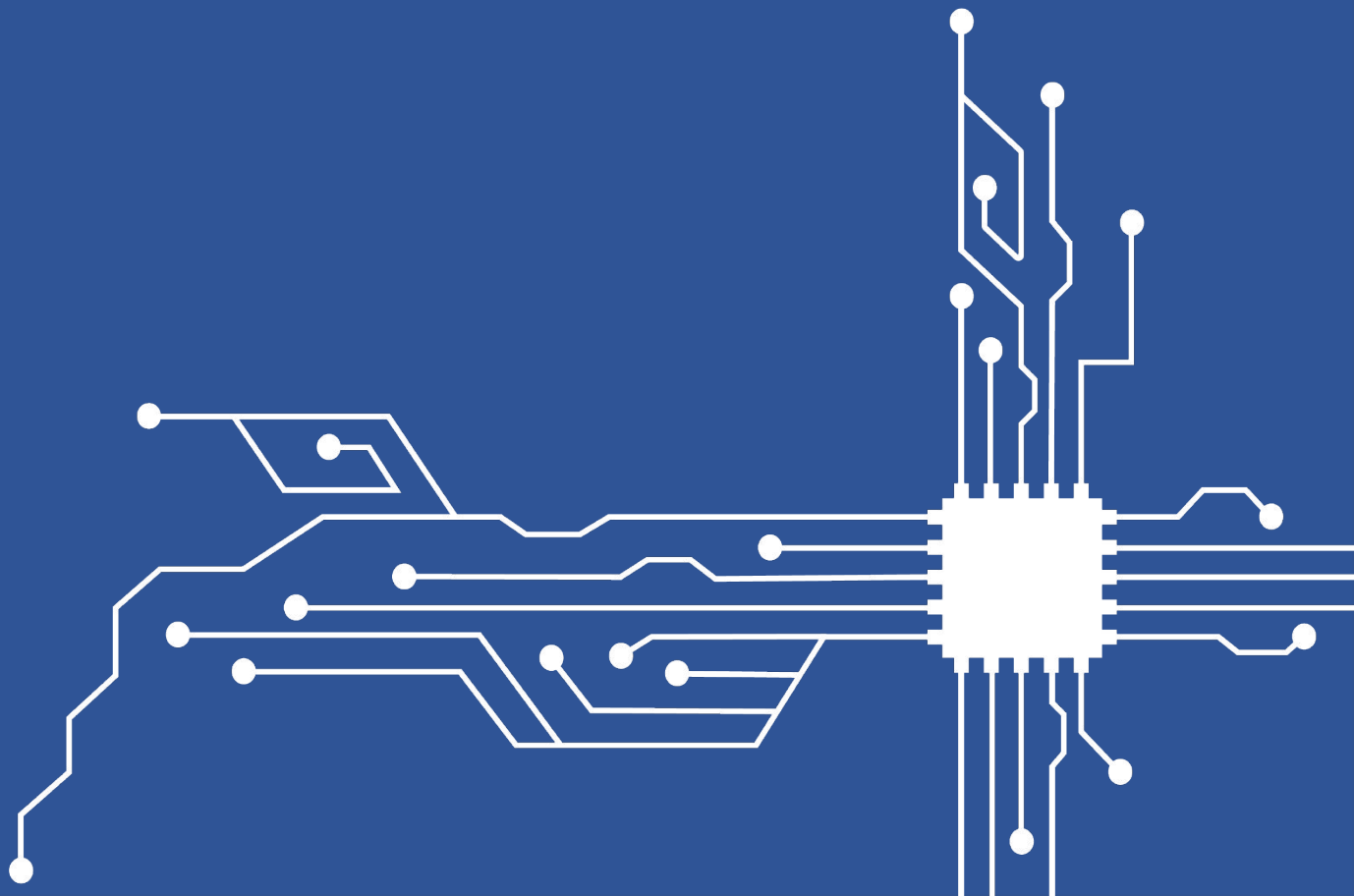


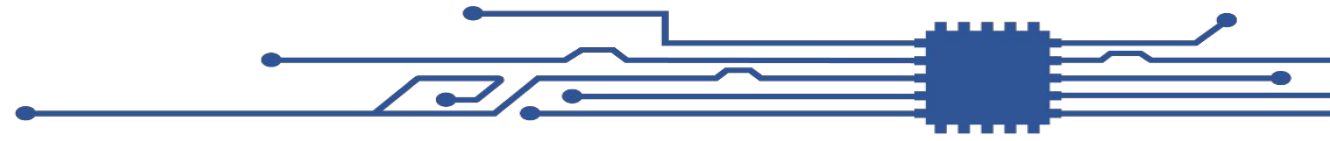


ReactiveUI

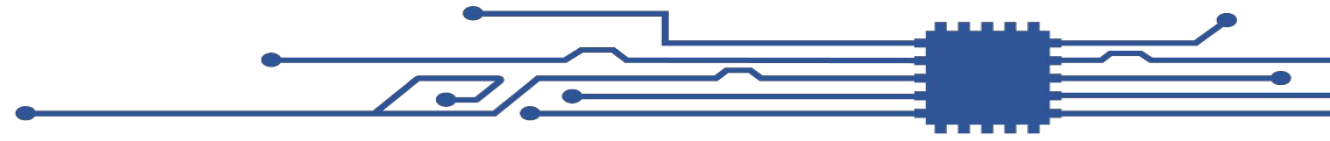
2020. 12. 01

펜타큐브 (주)







Reactive UI





Reactive UI

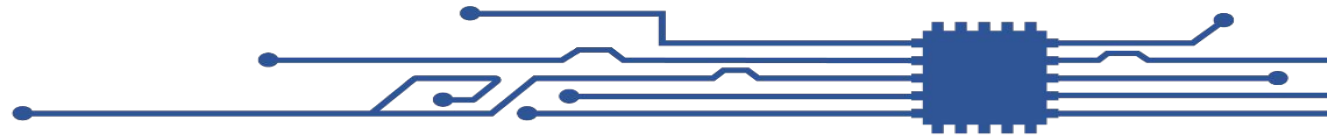
; 반응하는 UI?

reactive 미국·영국 [ri'æktiv]  영국식 

1. 반응을 보이는
2. (화학) 반응을 하는

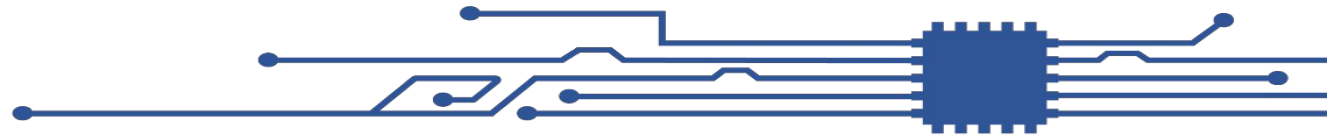
react(reactive) 미국·영국 [ri'ækt]  영국식  ★★

1. 반응하다, 반응을 보이다
2. (특정 물질에 대해 신체적으로 좋지 않은) 반응을 보이다
3. 화학 반응을 일으키다



Reactive UI

→ Reactive한 UI 구현을 돕는 라이브러리



Reactive Programming

반응형 프로그래밍



Reactive Programming

반응형 프로그래밍



ReactiveX is a combination of the best ideas from the **Observer pattern**, the **Iterator pattern**, and **functional programming**

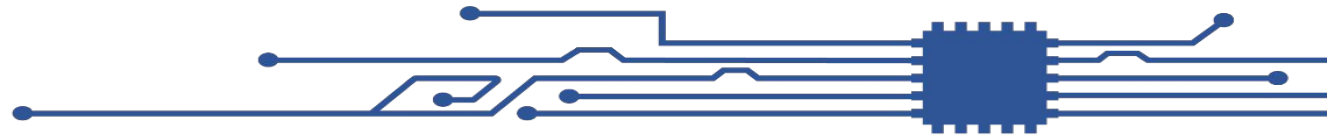


Reactive Programming

반응형 프로그래밍



.NET에는 ReactiveX 구현체로 System.Reactive(<https://github.com/dotnet/reactive>)가 있음



Reactive Programming

반응형 프로그래밍



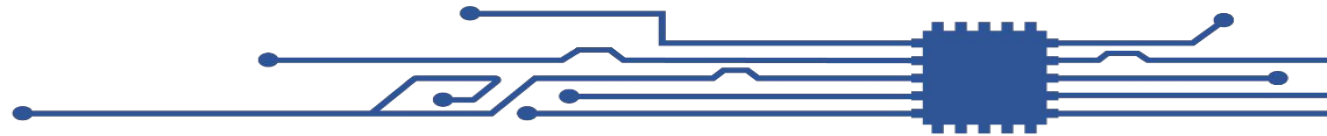
.NET에는 ReactiveX 구현체로 `System.Reactive`(<https://github.com/dotnet/reactive>)가 있음
실제로 `ReactiveUI.WPF`에서도 `System.Reactive`를 참조

`.NETFramework 4.7.2`

`ReactiveUI (>= 12.1.5)`

`System.Reactive (>= 4.4.1)`

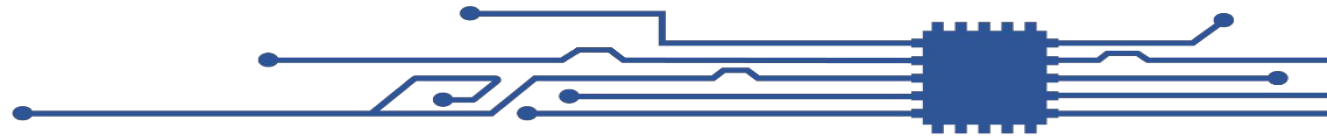




Reactive UI

어디에 좋은가?

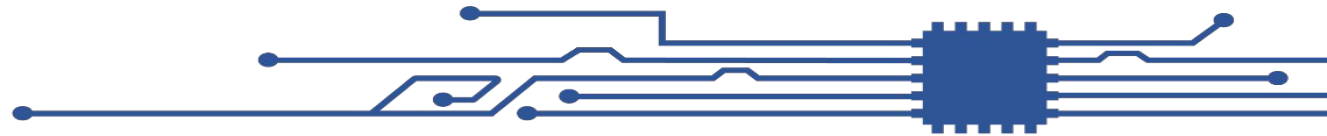
- 선언적 코드를 사용하기 때문에 관계 파악 용이
- Threading이 편함
- Throttle 지원
- Nested Property 지원 -> 중재자 패턴 구현 가능



Reactive UI

언제 쓰면 좋은가?

- Model, ViewModel이 복잡해질 때
- Delay, Throttling 같이 시간과 관련된 작업이 필요할 때
- Exception 처리가 필요할 때
- Thread를 넘나들며 처리해야할 때



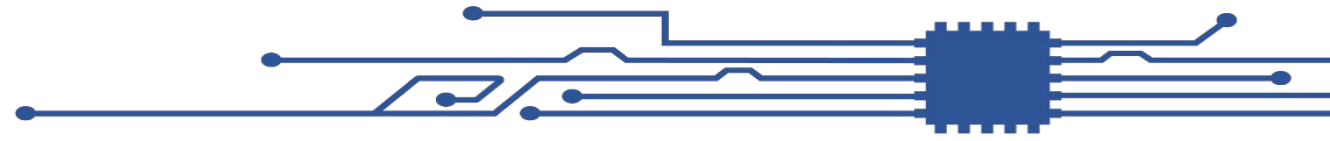
Prism vs ReactiveUI

뭐가 더 좋은가?

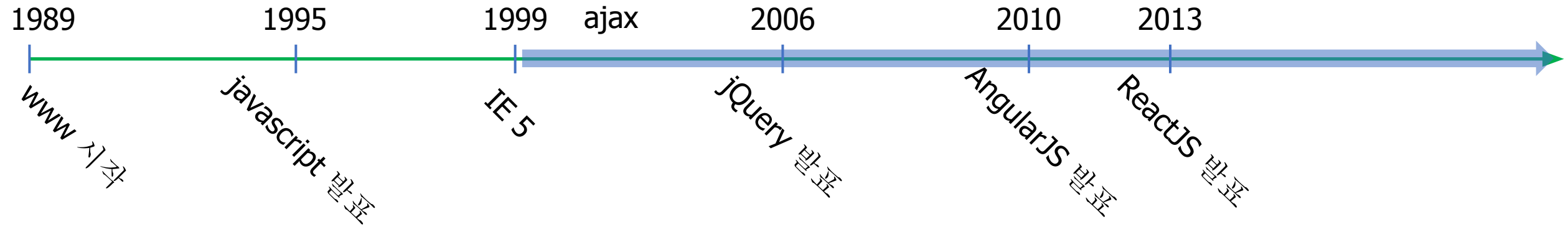
- 프레임워크
- 다양한 기능이 있음
- ViewModel 작성은 비교적 약함
- 함수 집합
- VM을 작성하는데 다양한 기능이 있음
- 프레임워크로는 비교적 약함

=> Prism을 쓰다 복잡한 기능이 필요한 부분은 ReactiveUI 사용

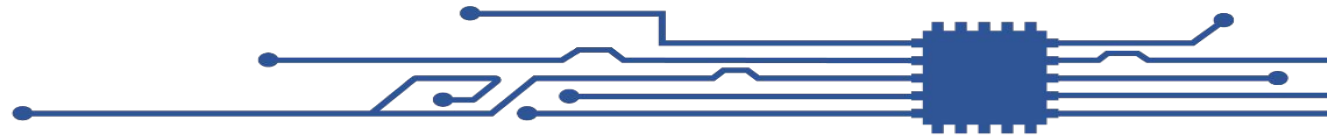
데이터 바인딩



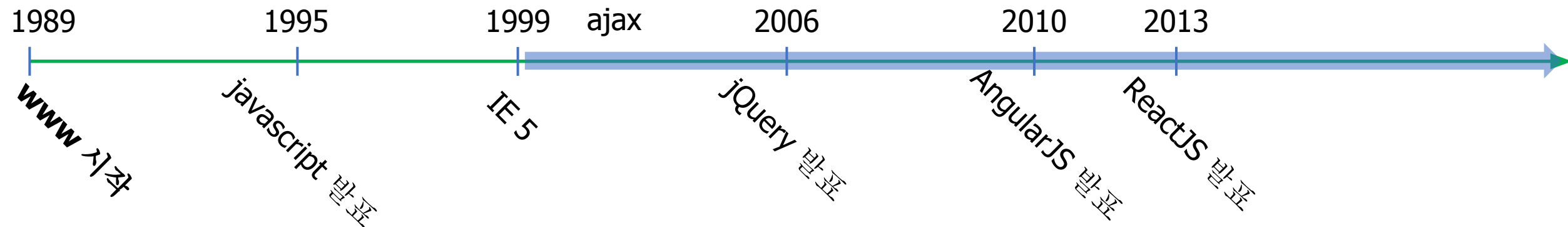
웹 + 프론트엔드의 역사 일부분



데이터 바인딩



웹 + 프론트엔드의 역사 일부분



World Wide Web

최초의 웹사이트 (<http://info.cern.ch/>)

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#)

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) ,X11 [Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#).)

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

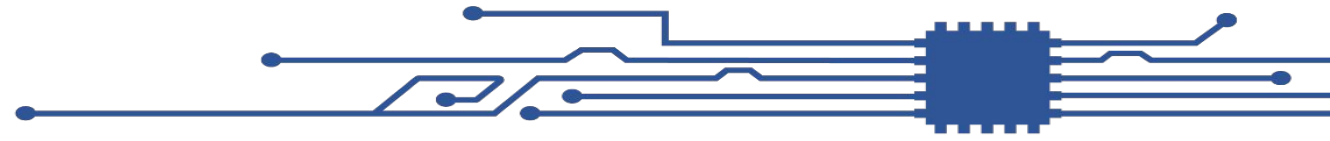
If you would like to support the web..

[Getting code](#)

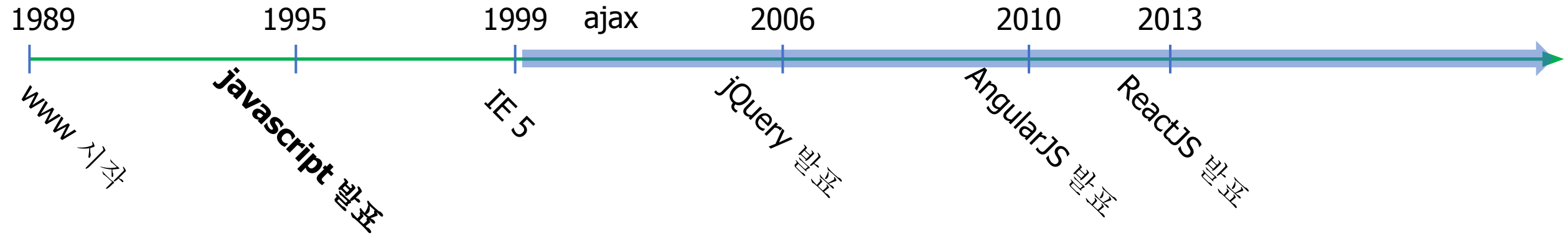
Getting the code by [anonymous FTP](#) , etc.



데이터 바인딩



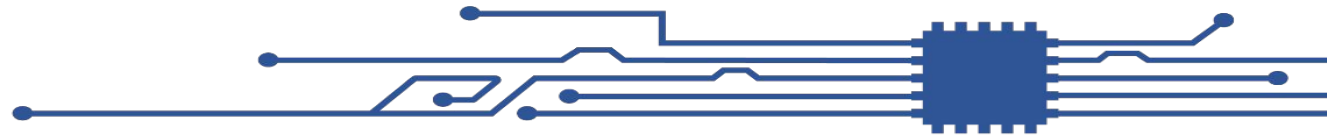
웹 + 프론트엔드의 역사 일부분



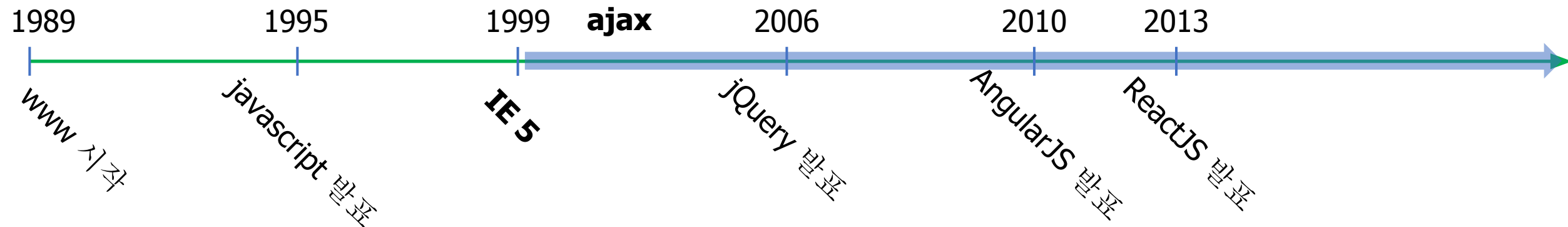
넷스케이프 브라우저에 탑재

Java와 이름이 비슷한 건 Java의 유명세를 빌리기 위해서라는 느낌

데이터 바인딩

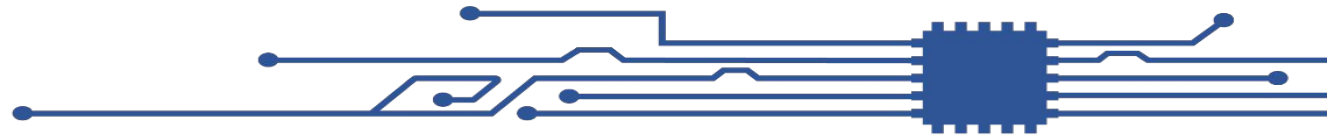


웹 + 프론트엔드의 역사 일부분

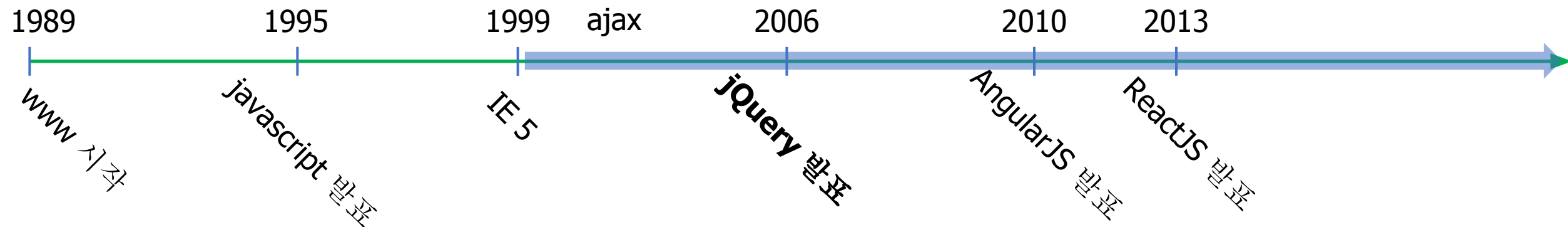


IE 5에 XMLHttpRequest 기능이 탑재, 이 기능을 기반으로 구현된 것이 ajax
Asynchronous JavaScript and XML의 축약
초기에는 알음알음 쓰이다가 본격적으로 쓰이는건 2000년대 중반 (이름도 이때 붙음)

데이터 바인딩



웹 + 프론트엔드의 역사 일부분

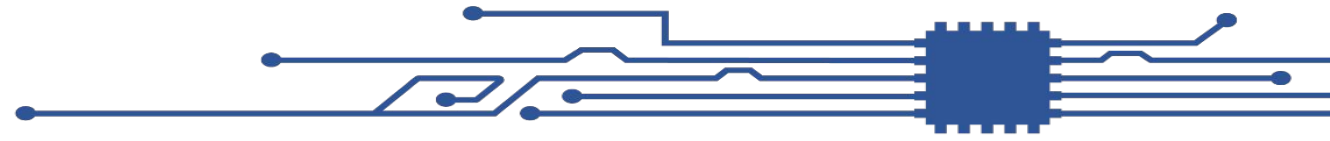


DOM 조작, 이벤트 처리, ajax 등등 다양한 기능을 가진 라이브러리
나온지 14년이 지난 지금도 점유율 최상위권

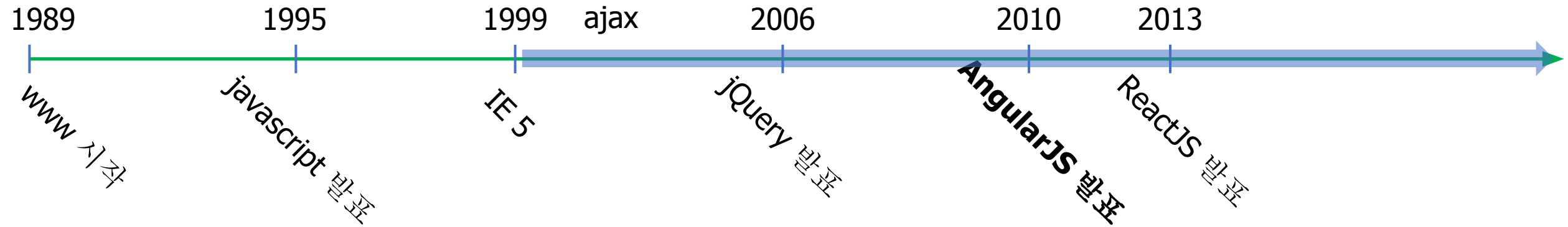
	2011 1 Jan	2012 1 Jan	2013 1 Jan	2014 1 Jan	2015 1 Jan	2016 1 Jan	2017 1 Jan	2018 1 Jan	2019 1 Jan	2020 1 Jan	2020 1 Dec
None	61.8%	49.1%	39.6%	38.2%	35.0%	28.7%	25.5%	24.0%	24.3%	23.8%	20.1%
jQuery	28.3%	42.8%	54.5%	57.4%	61.5%	68.3%	71.9%	73.1%	73.6%	74.2%	77.1%
Bootstrap					5.9%	10.5%	14.2%	16.5%	18.3%	20.1%	21.4%



데이터 바인딩

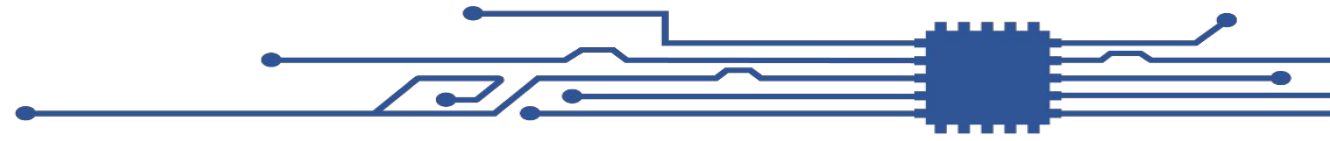


웹 + 프론트엔드의 역사 일부분

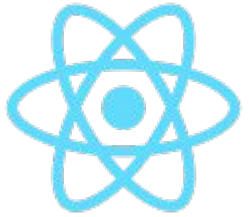
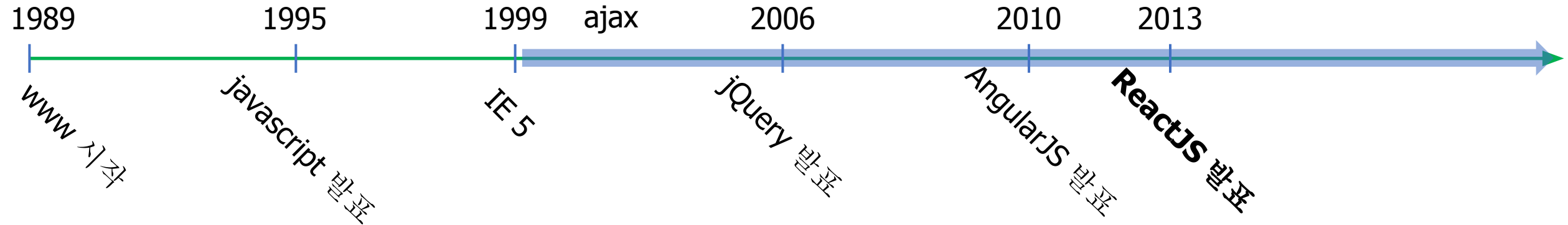


구글에서 만든 SPA(Single Page Framework) 프레임워크
프로젝트 구조 초기화, 양방향 데이터 바인딩 등을 지원

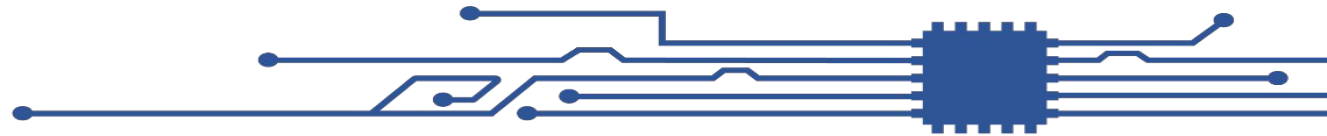
데이터 바인딩



웹 + 프론트엔드의 역사 일부분

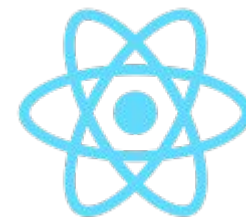


페이스북에서 만든 Component 라이브러리
단방향 데이터 바인딩 기반으로 개발



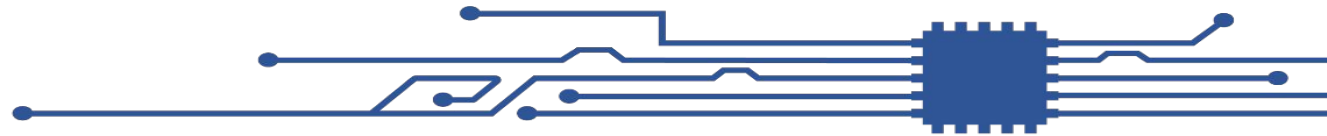
양방향 바인딩

데이터의 흐름이 양방향
보통 단방향보다 코드가 짧음
WPF MVVM과 유사
데이터 흐름이 복잡해지면 개발이 매우 힘들어짐



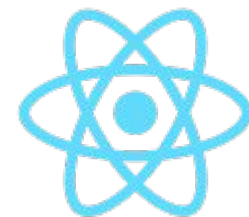
단방향 바인딩

데이터의 흐름이 단방향
양방향보다 코드가 긴 경우가 많음
WPF MVVM에서 제한적으로 사용 가능
데이터 흐름이 복잡해져도 상대적으로 쉬움



양방향 바인딩

```
template:
  '<ul>' +
    '<li ng-repeat="phone in $ctrl.phones">' +
      '<span>{{phone.name}}</span>' +
      '<p>{{phone.snippet}}</p>' +
    '</li>' +
  '</ul>',
controller: function PhoneListController() {
  this.phones = [
    {
      name: 'Nexus S',
      snippet: 'Fast just got faster with Nexus S.'
    }, {
      name: 'Motorola XOOM™ with Wi-Fi',
      snippet: 'The Next, Next Generation tablet.'
    }, {
      name: 'MOTOROLA XOOM™',
      snippet: 'The Next, Next Generation tablet.'
    }
  ]
};
```



단방향 바인딩

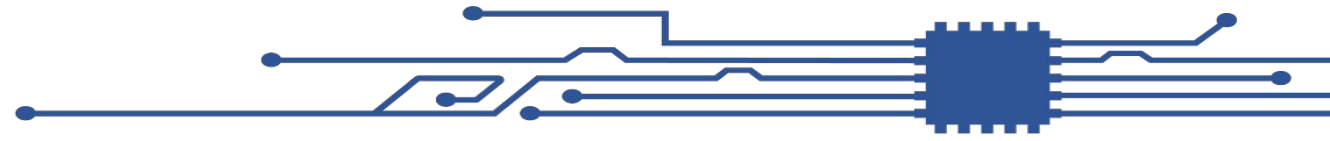
```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // 콜백에서 `this`가 작동하려면 아래와 같이 바인딩 해주어야 합니다.
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(state => ({
      isToggleOn: !state.isToggleOn
    }));
  }

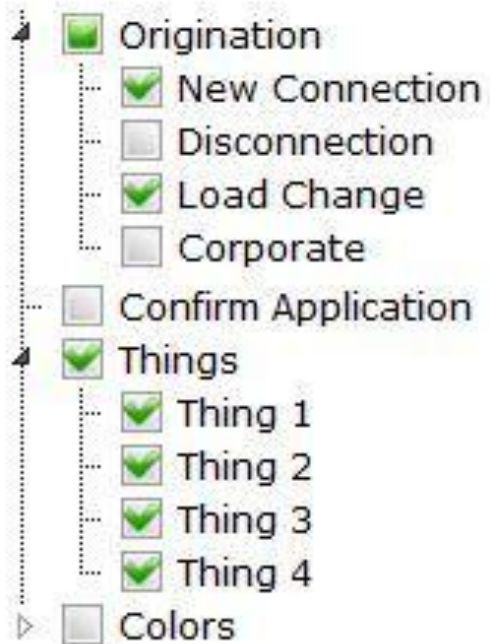
  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}
```



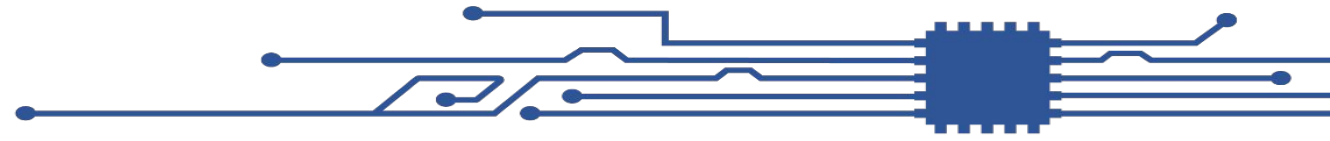


Check Tree 상태 관리

jsTree demo

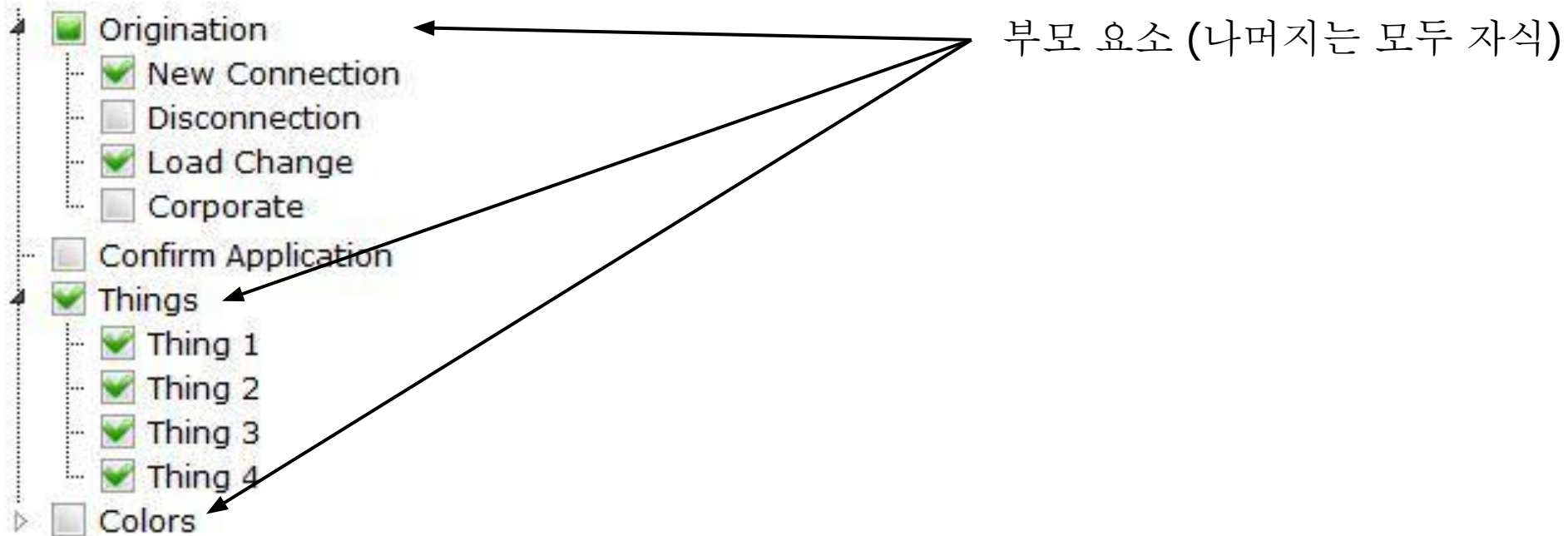


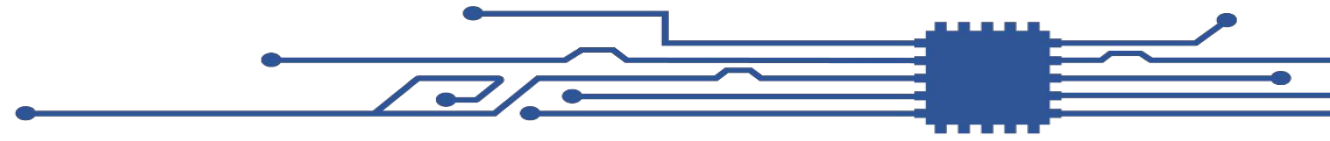
각 요소의 Check 상태를 어떻게 하는가?



Check Tree 상태 관리

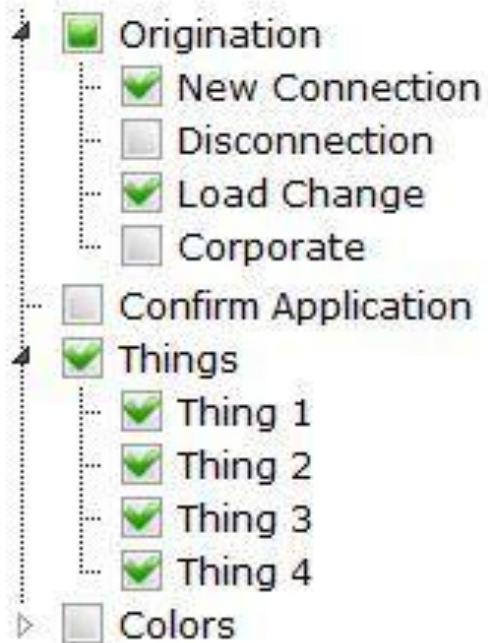
jsTree demo



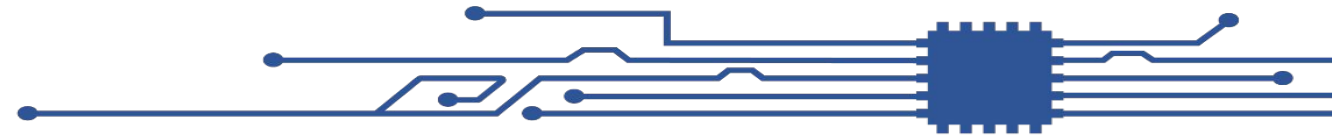


Check Tree 상태 관리

jsTree demo

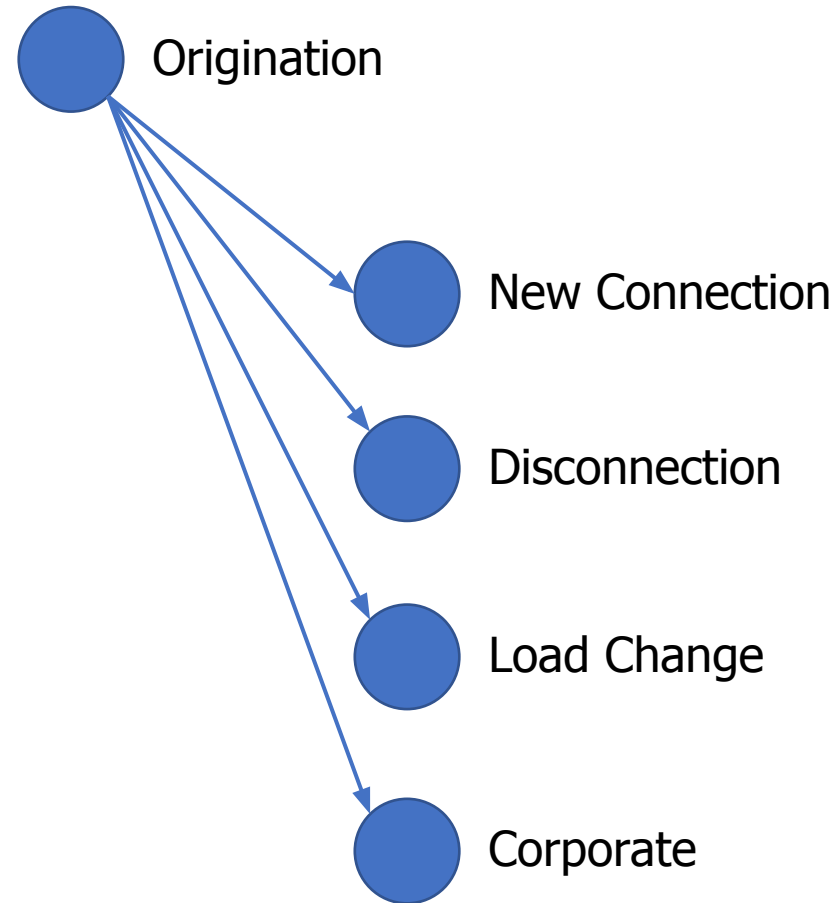
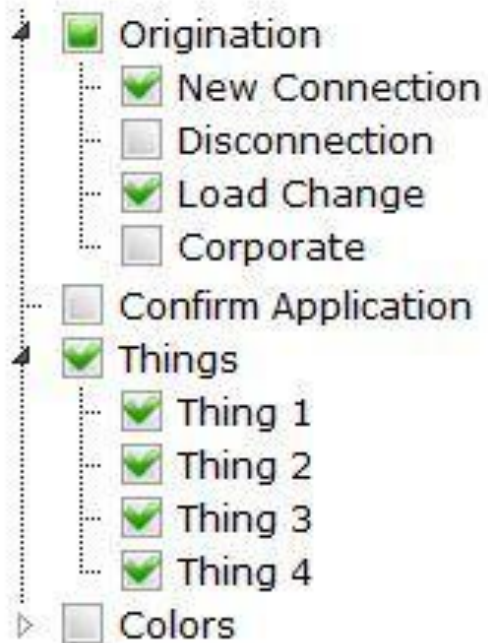


- 자식(트리의 Leaf)은 항상 Check/Uncheck
- 부모는 모든 자식의 Check 상태가 같다면 같은 상태로 설정 (자식 A, B, C가 모두 체크라면 부모도 체크 상태)
- 자식이 없거나 상태가 다른 자식이 있다면 중립 상태



Check Tree 상태 관리

jsTree demo





77
E

