JavaScript Framework 소개

2019.03

박재성

목차

1. Framework
2. 브라우저간 차이
3. 스펙 지원
4. 코드 작성
5. 라이브러리 소개

목차

1. Framework
2. 브라우저간 차이
3. 스펙 지원
4. 코드 작성
5. 라이브러리 소개

Framework란?

"소프트웨어의 구체적인 부분에 해당하는 설계와 구현을 재사용이 가능하게끔 일련의 협업화된 형태로 클래스들을 제공하는 것"

랄프 존슨(Ralph Johnson)

"GoF의 디자인 패턴" (Design Patterns : Elements of Reusable Object-Oriented Software) 저자

JavaScript Framework

- 동적이고 다양한 상호 작용을 하는 웹 애플리케이션을 개발할 때 필요한 많은 기능을 포함하고 있으며, 기능 단위의 컴포넌트로 분리해 컴포넌트의 재사용성과 UI를 손쉽게 개발할 수 있는 다수의 기능들을 제공한다.
- 컴포넌트 단위에만 집중하게 만들고, 그 외의 DOM 인터렉션 등은 자체적인 알고리즘을 통해 효율적으로 관리해 최적화된 성능을 이끌어 내도록 한다.

JavaScript Framework의 주요 기능 :

- 브라우저 차이에 따른 호환 처리와 브라우저 정보 제공 기능
- 웹 페이지와 구성 요소에 대한 정보 확인 및 조작 기능
- DOM에 대한 정보 확인 및 조작 기능
- AJAX를 사용한 동적인 데이터 처리 기능

Browser Wars

브라우저 전쟁

1990년대 말 마이크로소프트사의 '인터넷 익스플로러'(IE)와 Netscape사의 'Netscape' 브라우저 간에 벌어졌던 브라우저의 시장 점유율을 두고 벌인 경쟁

시장점유율을 높이기 위해 새로운 스펙의 경쟁적 도입

→ 표준을 벗어난 자체 스펙
→ IE의 윈도우 OS 기본 번들 포함으로 MS의 승리
→ IE의 높은 시장점유율을 바탕으로 IE6 이후 업데이트 소홀
→ 넷스케이프 소스를 기반으로 한 Mozilla 재단 설립,
그리고 2004.11 Firefox 1.0 발표
→ 2008.9.2 구글 크롬 브라우저 발표

크롬 발표 이후, 신규 스펙들의 구현과 실 사용의 싸이클이 빨라졌으며 현재 대다수의 모던 브라우저 들은 자체적으로 최신 업데이트로 자동 업데이트 된다.

이러한 브라우저는 '**Evergreen Browser**'라 부른다.
→ http://tomdale.net/2013/05/evergreen-browsers/

목차

1. Framework
2. 브라우저간 차이
3. 스펙 지원
4. 코드 작성
5. 라이브러리 소개

브라우저의 차이점

ES spec 지원여부

- 매해 새로운 표준 spec 들이 릴리스 되지만, 브라우저들에서 즉시 또는 빠른 시일 내로 구현되진 않는다.
- 대상 사용자들의 브라우저 버전에 따라, 작업된 JavaScript는 ES5 등과 같은 보다 낮은 버전의 syntax로 변환되어야 한다.

		Compilers/polyfills						Desktop browsers										
	98%	56%	71%	71%	50%	68%	17%	5%	11%	96%	96%	96%	98%	98%	98%	98%	98%	98%
Feature name ▸	Current browser	Traceur	Babel 6 + core-js	Babel 7 + core-js	Closure 2018.10	Type-Script + core-js	es6-shim	Kong 4.14[1]	IE 11	Edge 17	Edge 18	Edge 19 Preview	FF 60 ESR	FF 61	FF 62	FF 63 Beta	FF 64 Nightly	CH 69, OP 56
Optimisation																		
proper tail calls (tail call optimisation) ▸	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Syntax																		
default function parameters ▸	7/7	4/7	4/7	4/7	5/7	5/7	0/7	0/7	0/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
rest parameters ▸	5/5	4/5	3/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
spread syntax for iterable objects ▸	15/15	15/15	13/15	13/15	11/15	14/15	0/15	0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
object literal extensions ▸	6/6	6/6	6/6	6/6	5/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
for..of loops ▸	9/9	9/9	9/9	9/9	6/9	9/9	0/9	0/9	0/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9
octal and binary literals ▸	4/4	2/4	4/4	4/4	2/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
template literals ▸	5/5	4/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
RegExp "y" and "u" flags ▸	5/5	3/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
destructuring, declarations ▸	22/22	20/22	21/22	21/22	20/22	21/22	0/22	0/22	0/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22
destructuring, assignment ▸	24/24	23/24	24/24	24/24	22/24	24/24	0/24	0/24	0/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24
destructuring, parameters ▸	24/24	19/24	21/24	21/24	20/24	21/24	0/24	0/24	0/24	23/24	23/24	23/24	24/24	24/24	24/24	24/24	24/24	24/24
Unicode code point escapes ▸	2/2	1/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
new.target ▸	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2

https://kangax.github.io/compat-table/es6/

Transpiler

트랜스파일러는 특정 syntax를 낮은 버전의 syntax로 '변환'해 준다.

```
// ES6 arrow function code
var sum = (num1, num2) => num1 + num2;

// transpiled to ES5
var sum = function(num1, num2) { return num1 + num2; }
```

장점:

- 개발자는 최신 syntax를 사용하더라도, transpiler를 통해 코드를 여러 브라우저 환경에서 실행되는 코드로 변환할 수 있기 때문에, 브라우저 버전의 중요성이 감소하게 된다.

단점:

- 변환 작업을 위한 도구의 설정 및 매번 변환(Transpiling/Compile) 단계를 거쳐야 한다.
- '변환'된 코드는 작성된 코드와는 다르기 때문에 디버깅과 코드의 가독성이 어려워 질 수 있다.

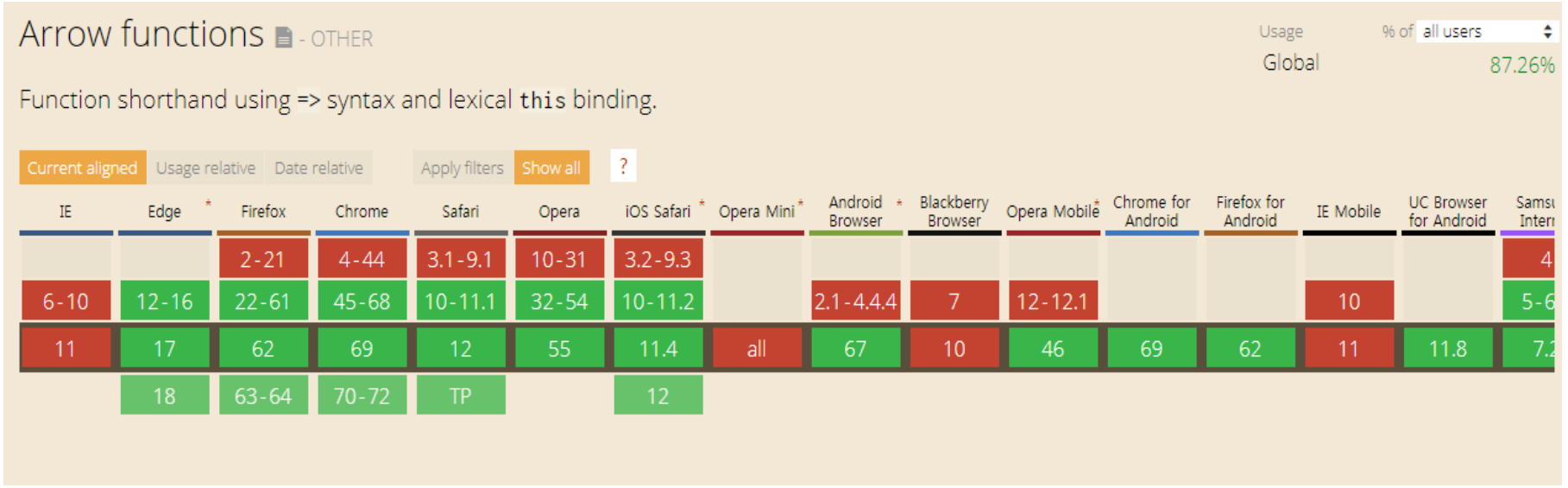
목차

1. Framework
2. 브라우저간 차이
3. **스펙 지원**
4. 코드 작성
5. 라이브러리 소개

스펙 지원 여부

브라우저에서 모든 스펙이 지원되지 않음

- ES6의 Arrow Function을 사용할 수 있는 브라우저는?



→ https://caniuse.com/#feat=arrow-functions

목차

1. Framework
2. 브라우저간 차이
3. 스펙 지원
4. **코드 작성**
5. 라이브러리 소개

직관적이고 간결한 코드의 사용

브라우저의 네이티브 DOM API는 많은 코드의 작성을 필요로 해 코드의 복잡도가 쉽게 증가한다.

- 요소 선택

```
var el = document.getElementById("element_id");
```

→ var el = $("element_id");

- 클릭 이벤트 바인딩

```
document.getElementById("element_id")
    .addEventListener("click", clickHandler, false);
```

→ $("#element_id").click(clickHandler); // jQuery

목차

1. Framework
2. 브라우저간 차이
3. 스펙 지원
4. 코드 작성
5. 라이브러리 소개

jQuery

- 존 레식(John Resig)이 2006년 초 개발
- 네이티브 객체(Native Object)를 감싸 래퍼 객체(Wrapper Class)로 기능 확장 및 메서드 체이닝으로 코드를 쉽게 작성
- DOM 요소를 쉽게 선택 또한 다양한 플러그인이 개발되어 있어 여러 기능을 쉽게 구현 가능
- 2013년까지 필수 라이브러리로 사용되었으나, 브라우저 환경 개선과 새로운 프레임워크 들의 등장으로 인해 현재는 그 사용이 점점 줄어드는 추세

- 특화된 버전 제공 :
  - jQuery Core - http://www.jquery.com : 일반적인 웹 환경에서 사용
  - jQuery UI - http://jqueryui.com : 애니메이션과 이펙트 처리
  - jQuery Mobile - http://jquerymobile.com : 모바일 환경 지원

React

- Facebook이 2013년 5월 발표한 프레임워크
- 컴포넌트 단위의 UI를 개발할 수 있으며, MVC 구조의 V(Views) 영역을 구현
- 직접적인 DOM 핸들링을 하지 않으며, 가상 DOM(Virtual DOM)을 통해 보다 효율적인 렌더링을 구현
- View만을 담당하기 때문에, 라우팅과 데이터 흐름 등을 담당하는 추가적인 도구를 같이 사용해야 한다.
- React 앱은 React Native 도구를 통해 네이티브 모바일 앱으로 개발될 수 있어, 단일 코드 베이스를 통해 웹, 모바일 앱 등의 개발이 가능

Angular

- 2009년 Miško Hevery과 Adam Abrons에 의해 개발된 MVC(또는 MVW) 프레임워크
- SPA(Single Page Application) 형태의 웹 어플리케이션 개발에 적합
- DOM 제어에 중점을 두지 않고 데이터 변화와 출력에 중점
- 2016년 9월에 Angular 2.0이 출시되었으나, 인기를 얻었던 이전 버전(AngularJS)와는 호환되지 않으며, 너무 많은 변화를 포함시켜 국내 환경에서는 이전과 같은 인기를 얻지 못하고 있다.

Vue.js

- 뷰 레이어에 집중하는 프레임워크로, Google에서 AngularJS 개발에 참여하기도 했던 Evan You가 2014년에 발표
- AngularJS에서 가장 마음에 드는 특징을 복제하는 것에서 시작
- MVVM(model-view-viewmodel) 패턴으로 MVC 패턴의 컨트롤러와 같이 데이터 관리 및 액션 처리에 집중
- React와 유사하게 데이터 바인딩과 컴포넌트에만 집중
- 빠른 속도로 성장을 이어나가고 있으며, 주요 컴포넌트로 성장해 Angular를 밀어내고 React vs Vue.js 대결 구도가 만들어 짐

고맙습니다.