ZeroPage

알아두면 쓸데있는

신기한 웹학사전

3주차

앞과 뒤? Front and Back?

셋째 주의 주제는 'Front and Back'입니다.

저번 시간에 우리는 웹의 앞 화면을 조작하며 재밌는 일들을 했습니다.

그러나, 우리가 조작하고 바꾼 내용은 네이버 웹 페이지를 새로고침하면 사라지게 되었습니다.

우리 마음대로 잘 바꿨는데, 네이버에는 왜 반영되지 않았을까요?

그 이유를 찾아보기 위해,

이번 시간에는 Front and Back에 대해 알아보겠습니다.

Front?

Front의 영어 단어의 뜻은 '앞'이라는 뜻입니다.

보통 웹 프로그래머들은 Front-end라고 하는데요,

Front-end는 말 그대로 '앞에 보이는 것'을 보통 지칭합니다.

우리가 화면으로 보고, 버튼을 클릭하고, 무언가 이미지를 보는 부분들은 전부

Front-end

Back?

Front의 영어 단어의 뜻은 '뒤'라는 뜻입니다.

보통 웹 프로그래머들은 Back-end라고 하며,

Back-end는 말 그대로 '앞이 아닌 뒤에서 돌아가고 있는 것'을 보통 지칭합니다.

우리가 보는 화면이 아니라,

뒤에서 길을 찾는 알고리즘, 어딘가에 저장되어 있는 동영상을 불러오는 등의 기능을 하는 것이 바로

Back-end

Front-end

Front-end는 우리가 저번 시간에 잠깐 봤었던 언어들로 이루어져 있습니다.

HTML, CSS, JS로 보통 이루어져 있다고 생각하면 됩니다.

여러분이 저번 시간에 웹 사이트의 변천사를 봤듯,

1~2년을 주기로 엄청나게 트렌드가 많이 바뀌는 분야가 Front-end라고 할 수 있습니다.

눈에 보이기 때문에 직관적이기도 하고,

디자인적 요소(UI/UX)도 많이 고려해야 하는 부분이며,

SEO(검색 엔진 최적화)도 해야하며,

Sematic HTML도 접목해야 하는,

이제는 꽤나 어려워진 분야입니다.

Trend : Front-end

프론트엔드를 대표하는 단어는 바로 Trend입니다.

Trend가 엄청나게 자주 바뀔 뿐 아니라,

새로운 내용, 새로운 것들이 수도 없이 많이 튀어나옵니다.

그렇지만, 우리는 웹을 배우는 입장이니 모를 수 없습니다.

따라서 간단하게 예전의 트렌드와 바뀐 트렌드를 알아보도록 하겠습니다.

OLD VER

한 10년?쯤 전만해도 프론트엔드는 엄청나게 간단한 분야였습니다.

HTML과 CSS로 페이지를 예쁘게 꾸미는 것에서 보통 끝나고,

JS로 일부 기능들을 짜임새 없는 코드로 엉망진창 넣기로 마무리 짓는

그냥 막 짜는 코드들이 대부분이었습니다.

But, In nowadays

Now,

HTML, CSS는 기본입니다. 심지어 HTML에 의미를 담아서 짜기도 합니다.

js는 어떻게 되었냐고요?

Angular, React, Vue와 같이 동적으로 웹을 바꿀 수 있는 기술들이 등장했습니다.

사용자가 무언가 입력을 주었다면,

화면의 전체가 Refresh(새로고침) 되며 보여주는 것이 아닌,

하나의 Page(Single Page)에서 일부 내용만 바꿔서 보여줄 수도 있는,

SPA(Single Page Application)의 형태를 취할 수 있게 되었습니다.

그리고 이 모든 코드는 대부분, js로 짜게 되었습니다.

Angular, React, Vue

이 세 가지, 프론트엔드를 지탱하는 기술들은

각각의 특징들이 있습니다. 다만, 각각의 특징은 실제 프론트 개발을 하지 않으면 느끼기 어렵습니다.

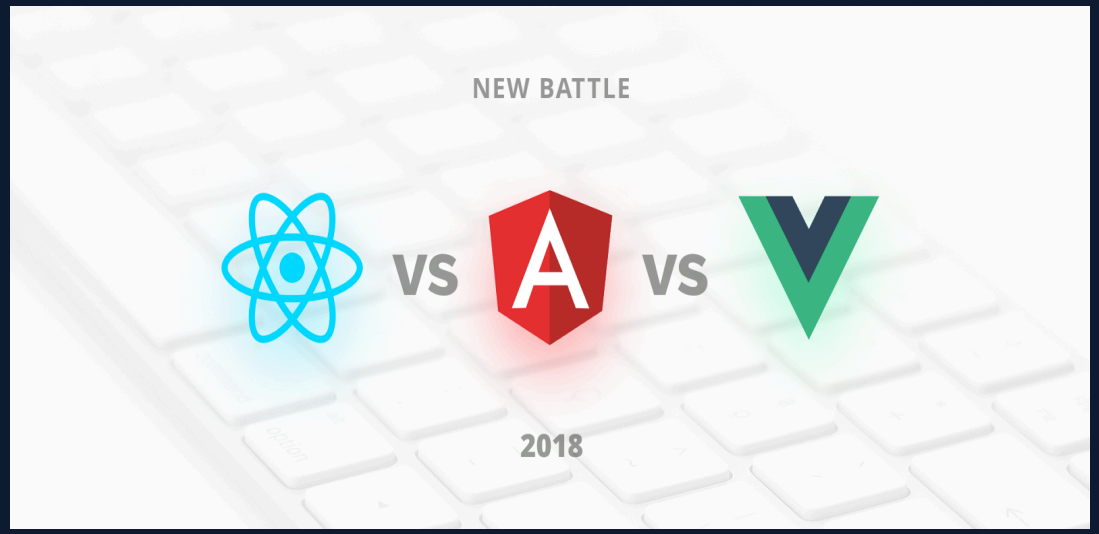
그렇지만, 한 가지 확실하게 말할 수 있는 것은

Angular는 구글이, React는 Facebook이 리드하고 있으며, Vue는 디자이너들에게 각광받고 있습니다.

한국에서는 보통 React 개발자를 많이 채용하지만,

사실 js를 어느정도 깊게 팠다면

이 세 가지에 덜 구애 받을 수 있습니다.



개발 상식 뽀개기) 라이브러리 OR 프레임워크

우리가 실제로 업무적 개발을 하다 보면 이 두 단어를 많이 접하게 됩니다.

하지만 많은 개발자들이 이 둘의 차이점을 굳이 알려고 하지도 않고,

중요하다고 생각하지도 않습니다.

물론, 개발에 영향은 없습니다. 그러나, 정확한 뜻을 알고 있는 것은 매우 중요합니다.

라이브러리는, 자체적으로 개발에 필요한 모든 요소들을 제공하지 않으며,

제대로 활용하기 위해선 추가적인 구성이 필요합니다.

프레임워크는, 자체적으로 개발에 필요한 모든 요소들을 제공할 수 있는 것을 말합니다.

참고로, Angular는 Framework이며 React는 UI 구성요소를 위한 Library입니다.

아무쪼록,

Javascript는 이제 프론트엔드에서 구박받던 천민이 아니라,

귀족이 되어버렸습니다.

자바스크립트를 못한다? === 당신 프론트엔드 개발자 맞아?

라는 공식이 성립할 정도입니다.

이 뿐 아니라, TypeScript(TS)라는 MS에서 나온 프로젝트로 인해

기존의 자바스크립트에 TS를 덧입혀 사용하는 것이 이제는 일반적입니다.
(정확하게는 js 중 ES5 버전의 Superset, 즉 상위 확장입니다.)

개발 용어 뿌시기) TS

TypeScript는 이제 모르면 안 되는 부분이 되어버렸습니다.

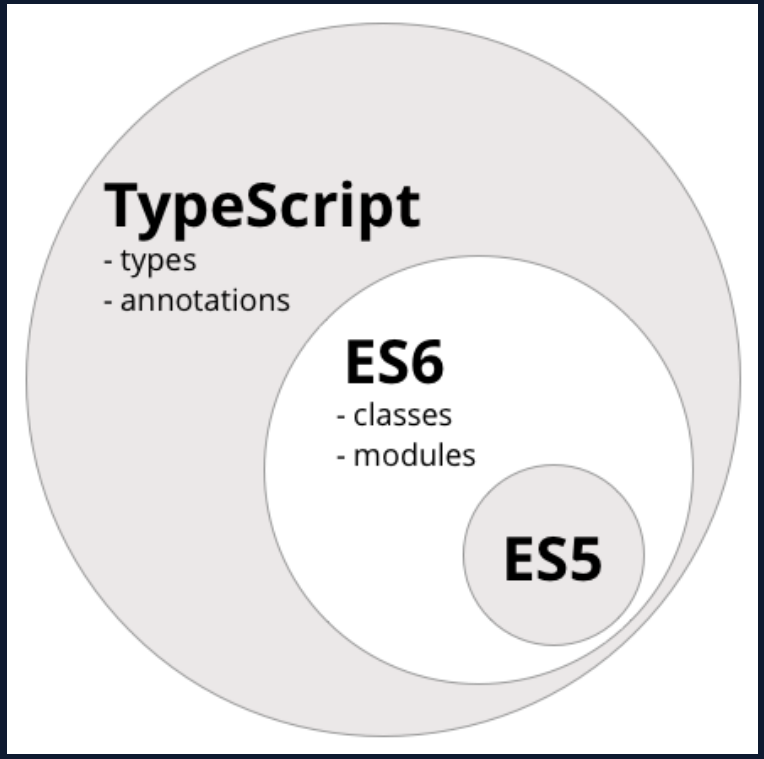
기존의 자바스크립트는 버전이 있는데,

현대 자바스크립트는 5 버전 이상을 일컫는 말입니다.
(Javascript == EcmaScript이며, 5 버전을 ES5라고 한다)

TS는 JS에 비해 여러 장점이 있습니다.
1. 정적 타입 지원
2. 도구의 지원(Intellisense, 타입 체크, 리팩토링)
3. 인터페이스, 제네릭 지원

단점은 하나라고 할 수 있는데,

'프로젝트에 설정하기 귀찮다.'입니다.



Back-end

백엔드를 우리는 보통 Server라고 부릅니다.

물론, 정확한 표현이라고 보기 어렵고 가끔 백엔드에 인프라(네트워크 등)이 포함되기도 하지만

편의를 위해서 Server라 부릅시다.

Server의 뜻은 뭘까요?

정말 말 그대로, '서빙하는 사람, 적합한 내용을 적합한 위치에 전달해주는 사람'이라고 할 수 있습니다.

실제로 백엔드도 이런 일을 한다고 생각하시면 됩니다.

사용자가 요구한 내용을,

잘 만들거나 가져와서,

요구한 사용자나 위치에 잘 전달해주는 것, 그것이 바로 Server입니다.

우리가 보는 것들은 From..

우리가 보는 것들은 다 Server에서 온다고 생각하면 됩니다.

Server로 온 것들을 Front에서 적절히 잘 보여주는 것이죠.

한 가지 예를 들자면,

네이버 지도에 중앙대학교까지 길 찾기를 검색한다면

중앙대학교 입력 후 서버로 전송(Front)

-> 길 찾는 알고리즘 실행, 소요시간 / 길(Path) 등 다양한 정보를 Front로 전송(Back)

-> 받은 정보를 바탕으로 잘 보여주기(Front)

이런 방식으로 진행이 됩니다.

우리가 보는 것들은 From..

또한 대량의 이미지 혹은 비디오를 다운받는 과정에서도

이미지와 비디오를 관리하고,

사용자가 요청한 이미지와 비디오를 선택적으로 가져오며,

이것을 Front에 잘 전송하는 메커니즘은

바로 Server에서 하게 됩니다.

HTTP? 그거 주소 앞에,

우리가 URL, 즉 www.~~~ 내용을 칠 때 항상 붙이는 단어가 있습니다.

HTTP / HTTPS라는 단어입니다.

HTTP는 어떤 방식으로 통신을 하고 싶은지 사람들끼리 정한 규약(Protocol)이라고 할 수 있습니다.

특히 HTTP는 웹을 위한 규약이라고 할 수 있습니다.

그렇기 때문에,

웹에서는 HTTP라는 규약에 맞춰 데이터를 전송하고 받고 합니다.

나중에 기회가 된다면 구체적으로 설명하겠습니다.

다채로운 Backend의 세계

Backend의 세계는 정말로 다채롭습니다.

아까 설명했던 언어에 따른 프레임워크가 다양한 것도 그렇지만,

다양한 기능을 설계할 수 있기 때문이죠.

단순히 웹사이트에 대한 정보를 줄 수도 있고,

알고리즘에 대한 결과값을 전송하기 위한 API 형태로 쓸 수도 있으며,

이미지와 동영상을 저장하고 꺼내 오는 용도로만 쓸 수도 있습니다.

이처럼 본인이 어떻게 설계하는지에 따라, 어떤 알고리즘을 사용하는지에 따라,

어떤 기술을 사용하는지에 따라 다양한 것들을 만들 수 있습니다.

Backend에서 주로 하는 고민들

Backend에서 주로 하는 고민들도 다양합니다.

하지만 우리가 학부에서 배우는 많은 내용들이 Backend와 아주 밀접한 관계가 있습니다.

몇 가지만 소개하자면,

1. 어떻게 하면 코드를 잘 관리할 수 있을까?
2. 어떻게 하면 더 확장성 있는 코드를 짤 수 있을까?
3. 어떻게 하면 더 효율적인 코드를 짤 수 있을까?
4. 효율성과 잘 관리된 코드 중에 어떤 것을 선택해야 할까?

와 같은 여러 가지 고민들을 하게 됩니다.

따라서 개인적으로는,

다양한 CS 이론들에 대한 지식, 알고리즘 이해 능력 등이 중요하다고 생각합니다.

Backend에서 주로 하는 고민들2

한 번 간단하지만 어려운 질문을 해보겠습니다.

우리가 쇼핑몰을 만든다면 어떤 기능들이 필요할까요?

혹은

우리가 로그인 기능만 만든다면?

만약 여러분이 제대로 만들려고 했다면

정말로 엄청나게 많은 것들을 생각하고 고민하며,

알고 있어야 합니다.

아 그렇다고 해서,

백엔드가 엄청난 거 같지만,

모든 것을 한 번에 잘 하는 사람은 없습니다.

또한 각 분야의 스페셜리스트들과 함께 팀을 꾸리게 되거나

주 목적이 한 가지인 팀에 속하게 됩니다.
(예를 들면, 사내 시스템 데이터베이스 최적화를 맡은 팀이라던지)

따라서 열심히 학교 공부를 하시는 것이 꽤나 도움이 된다고 할 수 있습니다.

Front vs Back

웹 개발을 하면서 많이들 고민하는 부분이 바로 이 부분입니다.

그래서 나는 Back을 해야 할까? Front를 해야할까?

정답은 생각보다 아주 간단합니다.

둘 다 하세요.

둘 다 하는 것이 가능하냐고요?

적어도 학생의 입장에서는 가능합니다.

다음 슬라이드에서 둘 다 해야 하는 이유를 설명하겠습니다.

Full Stack Developer

프론트엔드와 백엔드를 둘 다 할 수 있는 개발자를 Full Stack Developer라고 합니다.

하지만 이 말은 유니콘과 같은 존재라고 할 수 있습니다.

아니.. 근데 전 슬라이드에서는 둘 다 하라면서요?!

네, 둘 다 해야 합니다.

그 이유는 다음과 같습니다.

1. 여러분의 프로젝트는 작습니다.

2. 둘이 연결되어 있는데 하나만 알고 있다는 게 말이 될까요?

3. 선택의 폭이 넓어집니다.

따라서 가능하다면 두 개를 구분 짓기 보다는 둘 다 할 수 있는 능력을 지향해야 합니다.

자 오늘은 끝!

오늘은 웹의 기본 구조라고 할 수 있는,

Front와 Back에 대해서 배웠습니다.

혹시 저번 시간에 했던 질문에 답할 수 있을까요?

왜 내가 바꿨던 것들이 새로고침하면 유지가 안 되는 건가요?

왜 내가 바꿨던 것들이 새로고침하면 유지가 안 되는 건가요?

이유는 바로,

Server에 업데이트 되지 않았을 뿐더러,

Frontend의 내용이 바뀌도록 실행한 코드가 없기 때문입니다.

이제는 조금 이해가 되셨나요?

다음 시간에는.

일단 웹의 골격이라고 배웠던 HTML && CSS에 대해 간단히 배우고

실제로 써 보면서 페이지를 하나 만들어 볼 예정입니다.

수고하셨습니다!