2장 JS란?

역사 (대충 알쓸잡식)

1995년 - 넷스케이프 커뮤니케이션즈는 웹페이지의 보조적인 기능을 수행하기위해 브라우저에서 동작하는 경량 프로그래밍 언어를 도입하기로 결정 → JS 탄생

1996년 - 7월 넷스케이프 커뮤니케이션즈의 브라우저 넷스케이프 내비게이터 2에 탑재되었고 그때 당시에는 모카라는 이름으로 쓰임

1996년 - 9월 라이브스크립트로 이름이 바뀌었음

1996년 - 11월 ECMA 인터네셔널에 표준화 신청

1996년 - 12월 자바스크립트라는 이름으로 최종 명명

1997년 - 7월 emca-262라 불리는 표준화된 JS 초판(ECMAScript 1) 사양이 완성되었으며 상표권 문제로 자바스크립트는 ECMAScript로 명명됨

1999년 - ECMAScript 3(ES3)이 공개

2009년 - ECMAScript 5(ES5)가 HTML 5와 같이 웹 표준 사양이 됨

2015년 - ECMAScript 6(ECMAScript 2015 || ES6)은 let, const 키워드와 화살표함수, 클래스, 모듈 등과 같이 범용 프로그래밍 언어로서 갖춰야 할 기능들을 대거 도입하는 큰 변화

표준 업데이트 리스트 (2020 까지)

버전	연도	특징
ES1	1997	초판
ES2	1998	ISO/IEC 16262 국제 표준과 동일한 규격 적용
ES3	1999	정규식 표현, TRY … CATCH
ES5	2009	HTML 5와 함께 출현한 표준안 JSON, strict mode, 접근자 프로퍼티, 프로퍼티 어트리뷰트 제어, 향상된 배열 조작 가능 (forEach, map, filter, reduce, some, every)

버전	연도	특징
ES6 (ECMAScript 2015)	2015	let/const, 클래스, 화살표 함수, 템플릿 리터럴, 디스트럭처링 할당, 스프레드 문법. rest 파라미터, 심벌, 프로미스. Map/Set, 이터러블, for … of, 제너레이터, Proxy, 모듈 import/export
ES7 (ECMAScript 2016)	2016	지수(**) 연산자, Array.prototype.includes, String.prototype.includes
ES8 (ECMAScript 2017)	2017	async/await, Object 정적 메서드(Object.value, Object.entries, Object.getOwnPropertyDescriptors)
ES9 (ECMAScript 2018)	2018	Object rest/spread 프로퍼티, Promise.prototype.finally, async generator, for await … of
ES10 (ECMAScript 2019)	2019	Object.fromEntries, Array.prototype.flat, Array.prototype.flatMap, optional catch binding
ES11 (ECMAScript 2020)	2020	String.prototype.matchAll, BigInt, globalThis, Promise.allSettled, null 병합 연산자, 옵셔널 체이닝 연산자, for … in enumeration order

랜더링

- HTML, CSS JS로 작성된 문서를 해석해서 브라우저에 시각적으로 출력하는 것으로 말함
- 서버에서 데이터를 HTML로 ㅂㄴ환해서 브라우저에게 전달하는 과정을 뜻하기도 함

💡
1. Ajax에 대해서 알아오시오
2. jQuery에 대해서 알아오시오
3. V8 엔진에 대해서 알아오시오
4. Nodejs에 대해서 알아오시오
5. SPA 프레임워크의 뜻을 알아오시오

JS 특징

→ 웹 브라우저에서 동작하는 유일한 프로그래밍 언어

기본 문법은 C, Java와 비슷하며 Self에서는 프로토타입 기반 상속을 선호하고 스킴에서는 일급함수의 개념을 차용했다.

→ 인터프리터 언어

대부분 모던 JS엔진 (크롬 V8, 파이어폭스의 SpiderMoney, 사파리의 JavaScriptCore, MS 엣지의 Chakra 등) 인터프리터와 컴파일러의 장점을 결합해 비교적 처리속도가 느린 인터프리터 언어의 단점을 해결

V8 엔진에 대해서 정리해보시오

→ 명령형, 함수형, 프로토타입 기반 객체지향 프로그래밍을 지원하는 멀티 패러다임 프로그래밍 언어

컴파일러 언어	인터프리터 언어
코드가 실행되기 전 단계인 컴파일 타임에 소스코드 전체를 한번에 머신코드로 변환한 후 실행함	코드가 실행되는 단계인 런타임에 문 단위로 한 줄씩 중간 코드인 바이트코드로 변환한 후 실행함
실행 파일을 생성함	실행 파일을 생성하지 않음
컴파일 단계와 실행 단계가 분리되어 있음. 명시적인 컴파일 단계를 거치고, 명시적으로 실행 파일을 실행함	인터프리트 단계와 실행 단계가 분리되어 있지않음 인터프리터는 한 줄씩 바이트코드로 변환하는 즉시 실행함
실행에 앞서 컴파일은 단 한번 수행됨	코드가 실행될 때마다 인터프리트 과정이 반복 수행함
컴파일과 실행 단계가 분리되어 있으므로 코드 실행 속도가 빠름	인터프리트 단계와 실행 단계가 분리되어 있지 않고 반복 수행되므로 코드 실행 속도가 비교적 느림