1. 노드 시작하기

1.1 핵심 개념 이해하기

1.1.1 서버

- 서버 : 네트워크를 통해 클라이언트에 정보나 서비스를 제공하는 컴퓨터 또는 프로그램

- 클라이언트 : 요청을 보내는 주체로 브라우저일 수도 있고, 데스크톱 프로그램일 수도 있고,   
모바일 앱일 수도 있고, 다른 서버에 요청을 보내는 서버일 수도 있음.

- 서버라고 해서 요청에 대한 응답만 하는 것은 아님. 다른 서버에 요청을 보낼 수도 있으며,   
이때는 요청을 보낸 서버가 클라이언트 역할을 함.

1.1.2 자바스크립트 런타임

- 노드는 자바스크립트 런타임이다. (자바스크립트 실행기)

- 런타임은 특정 언어로 만든 프로그램들을 실행할 수 있는 환경

- 노드는 V8과 더불어 libuv라는 라이브러리를 사용  
(libuv 라이브러리는 노드의 특성인 이벤트 기반, 논 블로킹 I/O 모델을 구현하고 있음)

1.1.3 이벤트 기반

- 이벤트 기반(event-driven)이란, 이벤트가 발생할 때 미리 저장해둔 작업을 수행하는 방식

- 이벤트 기반 시스템에서는 특정 이벤트가 발생할 때 무엇을 할지 미리 등록해두어야 함.  
이를 이벤트 리스너(event listener)에 콜백(callback) 함수를 등록한다고 표현

- 노드도 이벤트 기반 방식으로 동작하므로, 이벤트가 발생하면 이벤트 리스너에 등록해둔 콜백 함수를 호출. 발생한 이벤트가 없거나 발생했던 이벤트를 다 처리하면, 노드는 다음 이벤트가 발생할 때까지 대기

- 이벤트 루프 : 이벤트 발생 시 호출할 콜백 함수들을 관리, 호출된 콜백 함수의 실행 순서 결정

- 백그라운드 : setTimeout 같은 타이머나 이벤트 리스너들이 대기하는 곳

- 테스크 큐 (콜백 큐) : 이벤트 발생 후, 백그라운드에서는 테스크 큐로 타이머나 이벤트 리스너들의 콜백 함수들을 보냄

1.1.4 논 블로킹 I/O

- 논 블로킹 : 이전 작업이 완료될 때까지 대기하지 않고 다음 작업을 수행함  
(논 블로킹과 동시가 같은 의미가 아님)

블로킹 : 이전 작업이 끝나야만 다음 작업을 수행함

- 노드에서는 동기와 블로킹이 유사, 비동기와 논 블로킹이 유사

1.1.5 싱글 스레드

- 노드가 채택하고 있는 방식 : 싱글 스레드, 논 블로킹 모델

- 노드가 싱글 스레드로 동작하지 않는 경우 : 스레드풀, 워커 스레드

1.2 서버로서의 노드

- 노드의 장점 : 멀티 스레드 방식에 비해 적은 컴퓨터 자원 사용, I/O 작업이 많은 서버로 적합, 멀티 스레드 방식보다 쉬움, 웹 서버가 내장되어 있음, 자바스크립트를 사용함, JSON 형식과 쉽게 호환됨

- 노드의 단점 : 기본적으로 싱글 스레드라서 CPU 코어를 하나만 사용, CPU 작업이 많은 서버로는 부적합, 하나뿐인 스레드가 멈추지 않도록 관리가 필요함, 서버 규모가 커졌을 때 서버를 관리하기 어려움, 어중간한 성능

1.3 서버 외의 노드

- 노드 기반으로 돌아가는 대표적인 웹 프레임워크 : 앵귤러(Angular), 리액트(React), 뷰(Vue) 등

1.4 개발 환경 설정하기

- 노드 설치

- npm 버전 업데이트

- 비주얼 스튜디오 코드 설치

2. 알아두어야 할 자바스크립트

2.1 ES2015+

2.1.1 const, let

- 보통 자바스크립트를 배울 때는 var로 변수를 선언하는 방법부터 배움. 하지만 var은 이제 const와 let이 대체함.

- const와 let의 차이 : const는 한 번 값을 할당하면 다른 값을 할당할 수 없음, 초기화할 때 값을 할당하지 않으면 에러 발생 (상수)

2.1.2 템플릿 문자열

- 백틱( ` ) : 문자열 안에 변수를 넣을 수 있음

2.1.3 객체 리터럴

- ES5 : { name: name, age: age }

- ES2015 : { name, age }

2.1.4 화살표 함수

- 화살표 함수에서는 function 선언 대신 => 기호로 함수를 선언함. 또한 변수에 대입하면 나중에 재사용 가능

2.1.5 구조분해 할당

- 구조분해 할당을 사용하면 객체와 배열로부터 속성이나 요소를 쉽게 꺼낼 수 있음.

2.1.6 클래스

- 다른 언어처럼 클래스 기반으로 동작하는 것이 아니라, 자바스크립트는 여전히 프로토타입 기반으로 동작함.

2.1.7 프로미스

- 자바스크립트와 노드에서는 주로 비동기를 접함 (특히 이벤트 리스너를 사용할 때 콜백 함수를 자주 사용)

- 프로미스를 쉽게 설명하자면, 실행은 바로 하되 결과값은 나중에 받는 객체 (결과값은 실행이 완료된 후 then이나 catch 메서드를 통해 받는다)

- 프로미스를 return한 경우에는 프로미스가 수행된 후 다음 then이나 catch가 호출된다.

- 프로미스 여러 개를 한 번에 실행하는 방법 : Promise.all

2.1.8 async/await

- 노드 7.6 버전부터 지원되는 기능, ES2017에서 추가됨

- 함수 선언부를 일반 함수 대신 async function으로 교체한 후, 프로미스 앞에 await를 붙임

- for await of : 프로미스 배열을 순회할 수 있음

- async 함수의 반환값은 항상 Promise로 감싸짐

2.2 프런트엔드 자바스크립트

2.2.1 AJAX (Asynchronous Javascript And XML)

- 비동기적 웹 서비스를 개발할 때 사용하는 기법

- 이름에 XML이 들어 있지만 꼭 XML을 사용해야 하는 것은 아니며, 요즘에는 JSON을 많이 사용

- 쉽게 말해 페이지 이동 없이 서버에 요청을 보내고 응답을 받는 기술

- 웹 사이트 중에서 페이지 전환 없이 새로운 데이터를 불러오는 사이트는 대부분 AJAX 기술을 사용하고 있다고 보면 됨

- AJAX 요청 : jQuery나 axios 같은 라이브러리를 이용해서 보냄

2.2.2 FormData

- HTML form태그의 데이터를 동적으로 제어할 수 있는 기능 (주로 AJAX와 함께 사용)

- append 메서드 : 키-값 형식의 데이터를 저장할 수 있음, has 메서드 : 주어진 키에 해당하는 값이 있는지의 여부를 알림, get 메서드 : 주어진 키에 해당하는 값 하나를 가져옴, getAll : 해당하는 모든 값을 가져옴, delete 메서드 : 현재 키를 제거, set : 현재 키를 수정

2.2.3 encodeURIComponent, decodeURIComponent

- AJAX 요청을 보낼 때, 주소에 한글이 들어가는 경우

- 서버 종류에 따라 다르지만 서버가 한글 주소를 이해하지 못하는 경우가 있는데, 이럴 때는 window 객체의 메서드인 encodeURIComponent 메서드를 사용

- 받는 쪽에서는 decodeURIComponent를 사용

2.2.4 데이터 속성과 dataset

- 보안 : 프런트엔드에서 데이터를 내려보낼 때 첫 번째로 고려해야 할 점임. (클라이언트를 믿지 마라!)

- 데이터 속성(data attribute) : HTML5에서 HTML과 관련된 데이터를 저장하는 공식적인 방법

3. 노드 기능 알아보기

3.1 REPL 사용하기

- 자바스크립트는 스크립트 언어이므로 미리 컴파일을 하지 않아도 즉석에서 코드를 실행할 수 있음.

- REPL(Read Eval Print Loop) : 입력한 코드를 읽고(Read), 해석하고(Eval), 결과물을 반환하고(Print), 종료할 때까지 반복한다(Loop)

- 노드의 REPL을 직접 사용하기 : 윈도우에서는 명령 프롬프트, 맥이나 리눅스에서는 터미널을 열고 node를 입력, VS Code에서는 Ctrl + ` 을 누르면 터미널을 켤 수 있음

- REPL 종료 : Ctrl + C 를 두 번 누르거나, REPL 창에 .exit을 입력

- REPL은 한두 줄짜리 코드를 테스트해보는 용도로는 좋지만 여러 줄의 코드를 실행하기에는 불편함. 긴 코드인 경우에는 코드를 자바스크립트 파일로 만든 후 파일을 통째로 실행하는 것이 좋음.

3.2 JS 파일 실행하기

- 자바스크립트 파일을 생성

- 콘솔에서 node [자바스크립트 파일 경로]로 실행. 확장자(.js)는 생략해도 됨

3.3 모듈로 만들기

- 노드는 코드를 모듈로 만들 수 있다는 점에서 브라우저의 자바스크립트와 다르다.

- 모듈이란 특정한 기능을 하는 함수나 변수들의 집합, 모듈 자체로도 하나의 프로그램이면서 다른 프로그램의 부품으로도 사용할 수 있음.

- 모듈로 만들어두면 여러 프로그램에 해당 모듈을 재사용할 수 있음

3.4 노드 내장 객체 알아보기

3.4.1 global