학습결과서

이름 : 박병찬

날짜 : 2020.07.03(금)

오늘 학습 한 내용

- Array 메소드들(reduce 메소드, find 메소드, IndexOf 메소드, forEach 메소드)

- 크롬 디버거 사용법

- 자바스크립트 ES6 문법

- 웹페이지 기초 레이아웃 잡기

오늘 알게된 사실

- 알고리즘 문제 풀이 시 모든 경우의 수에 있어서 배열 전체 요소를 다 탐색하는 것은 너무나 비효율적이다. 오늘 푼 알고리즘 문제의 경우에는 key에는 참가자의 이름, value 에는 참가자 수를 프로퍼티로 가지는 개체로 만들어준 후 이를 효율적으로 해결하였다. 무조건 for문처럼 배열 전체 요소를 탐색하는 것보다 reduce를 이용하여 객체를 만들어 해결할 수 있다는 것을 알게 되었다. Reduce의 사용법에 대해 더 자세히 알 수 있었다.

- 크롬 디버거 사용법에 대해 처음 알게 되었다. 중단점 설정 및 watch 탭에서 중단점 전에 내가 테스트해보고 싶은 코드를 입력하면 그 결과값을 즉각적으로 알 수 있어서 앞으로 디버깅 시에 많이 활용할 것같다.

- 이전에 grid layout에 대해 학습했었는데 개념이 명확이 정립되어 있지 않았다. 이번 기술면접 문제를 풀면서 grid-column, grid-row등 각각의 grid item들이 각각의 칸을 어떤 식으로 차지하는지에 대해 확실히 알게 되었다.

- spread 연산자와 Rest 파라미터 둘 다 …을 사용해서 헷갈렸었는데 기술면접 문제를 풀 때 예시코드를 보면서 더 잘 알게 되었다.

- 코딩 스타일

**중괄호**

if (n < 0) {alert(`Power ${n} is not supported`);}

코드가 짧다면 중괄호 없이 한 줄에 쓰는 방법도 괜찮다.

if (n < 0) {

alert(`Power ${n} is not supported`);

}

가장 추천하는 방법은 다음과 같다.

코드 블록을 사용하는 방법이 가장 가독성이 좋으므로 이 방법을 추천합니다.

**가로 길이**

가로로 길게 늘어진 코드를 읽는 걸 좋아하는 개발자는 없습니다. 코드의 가로 길이가 길어진다면 여러 줄로 나눠 작성하는 게 좋습니다. 최대 가로 길이는 팀원들과 합의해 정하는게 좋습니다. 대개 80자나 120자로 제한하는 게 일반적입니다.

// 백틱(`)을 사용하면 문자열을 여러 줄로 쉽게 나눌 수 있습니다.

let str = `

ECMA International's TC39 is a group of JavaScript developers,

implementers, academics, and more, collaborating with the community

to maintain and evolve the definition of JavaScript.

`;

If 문이라면 아래와 같이 작성할 수 있다.

if (

id === 123 &&

moonPhase === 'Waning Gibbous' &&

zodiacSign === 'Libra'

) {

letTheSorceryBegin();

}

**들여쓰기**

가로 들여쓰기 : 요즘엔 탭 대신 스페이스를 이용하는 게 더 우위에 있는 것같다. (스페이스 두 개 혹은 네 개를 사용해 만듬.)

show(parameters,

aligned, // 스페이스 다섯 개를 이용해 들여쓰기 함

one,

after,

another

) {

// ...

}

세로 들여쓰기 : 논리 블록 사이에 넣어 코드를 분리해주는 새 줄

함수 하나에 논리 블록 여러 개가 들어갈 수 있습니다. 아래 예시에서 변수 선언, 반복문, 리턴문 사이에 세로 들여쓰기를 해주는 빈 줄을 넣어 코드를 분리해 보았습니다.

function pow(x, n) {

let result = 1;

// <--

for (let i = 0; i < n; i++) {

result \*= x;

}

// <--

return result;

}

이렇게 여분의 줄을 넣어주면 코드의 가독성이 좋아집니다. 읽기 쉬운 코드를 만들려면 세로 들여쓰기 없이 코드를 아홉 줄 이상 연속해서 쓰지 마세요.

**세미콜론**

자바스크립트 엔진에 의해 무시되더라도 모든 구문의 끝엔 세미콜론을 써주는 것이 좋다.

**중첩 레벨**

가능하면 너무 깊은 중첩문은 사용하지 않도록 합시다. 반복문을 사용할 때 중첩문의 깊이가 깊어지면 continue 지시자를 쓰는 게 좋은 대안이 될 수도 있습니다.

for (let i = 0; i < 10; i++) {

if (!cond) continue;

... // <- 추가 중첩 레벨이 추가되지 않습니다.

}

**함수의 위치**

헬퍼’ 함수 여러 개를 만들어 사용하고 있다면 아래와 같은 방법을 사용해 코드 구조를 정돈할 수 있습니다.

코드를 먼저, 함수는 그 다음에 선언하기

// 헬퍼 함수를 사용하는 코드

let elem = createElement();

setHandler(elem);

walkAround();

// --- 헬퍼 함수 ---

function createElement() {

...

}

function setHandler(elem) {

...

}

function walkAround() {

...

}

사람들은 코드를 이 코드가 '무엇을 하는지’를 생각하며 코드를 읽기 때문에 코드가 먼저 나오는 것이 자연스럽기 때문입니다. 이름만 보고도 헬퍼 함수의 역할을 쉽게 유추할 수 있게 헬퍼 함수 이름을 명명했다면 함수 본문을 읽을 필요도 없습니다.

내가 부족하다고 느낀 부분

- 알고리즘 문제를 풀 때 자바스크립트의 배열 및 객체 메서드 사용에 아직 미숙해서 많은 예시들을 보는 것이 필요합니다.

- 다양한 웹페이지 레이아웃에 대한 이해가 부족하여 그 부분을 채워나갈 예정입니다.

- ES6 문법 중 Promise, async / await 에 대해서 아직 잘 몰라서 더 많은 공부가 필요합니다.

수업에 바라는 점

- 화장실에 휴지가 없습니다.

-

-