9/20 강의자료

이전자료 중 추가자료

08 반복문으로 구구단 완성하기

▼ document.writeln()을 사용하면 줄바꿈이 되는거 아닌가요?

네 맞습니다. 하지만 HTML은 '\n'을 공백으로 인식하기 때문에 줄바꿈이 되지 않습니다.

줄바꿈을 위해서는 형식을 그대로 표현하는 <pre>태그나 줄바꿈을 해주는
태그를 추가해줘야 합니다.

▼ <pre>태그는 뭔가요?

<pre>요소는 미리 서식을 지정한 텍스트를 나타내며 HTML에 작성한 내용 그대로 표현합니다

하지만 유지보수 차원에서 잘 사용하지는 않는 태그입니다. 따라서
태그를 사용하시길 권장 드립니다.

플러그인 추천

code Runner

VScode에서 바로 JS코드 실행해 콘솔창을 볼 수 있다.

1교시

1-01 값 입력받기

▼ readline 보충설명

readlie은 js에 내장된 모듈로 한 줄 씩 입출력을 처리할 수 있게 도와주는 모듈입니다.

먼저 require("모듈이름")을 통해서 모듈을 불러오고 변수에 저장합니다.

readline 안의 메서드인 createInterface()를 통해서 인터페이스 객체를 생성하고 input에 사용자의 입력을 수신하는 모듈인 process.stdin를 ouput에 출력을 하게 해주는 모듈인 process.stdout을 할당합니다.

```
rl.on("line", line⇒ {
    // your code
    rl.close()
```

}

을 통해 입력받은 줄을 line에 저장해서 사용한 후 rl.close를 통해 닫아줍니다.

rl.on('close', () ⇒ {

    // your code

}

을 통해 입력이 끝난 후 하고싶은 일을 작성합니다.

참고자료

readline 모듈 : https://nodejs.org/api/readline.html

process모듈 : https://nodejs.org/api/process.html

console.log와 process.stdout.write의 차이점 : https://www.geeksforgeeks.org/difference-between-process-stdout-write-and-console-log-in-node-js/

▼ 모듈을 불러오는 방법은?

require문법과 import 문법이 있습니다.

(require / exports) 는 NodeJS에서 사용되고 있는 CommonJS 키워드이고 Ruby 언어 스타일과 비슷하다라고 볼수 있습니다.

cf. CommonJS는 JS에서 모듈화작업을 할 수 있게 만들어 줍니다.

(import / export) 는 ES6(ES2015)에서 새롭게 도입된 키워드로서 Java나 Python 언어 방식과 비슷하다.

차이점

1. require()는 코드를 적은 곳에서 불러와 어느 지점에서나 호출이 가능한 반면 import는 항상 맨 위로 이동하여 파일 시작부분에서만 실행됩니다.(호이스팅)

- 하지만 비동기를 사용하면 나중에 불러올 수 있습니다.

  참고 : https://inpa.tistory.com/entry/JS-📚-모듈-사용하기-import-export-정리?category=889099#브라우저(HTML)에서_모듈_사용_하기

- cf. 호이스팅이란 쉽게 생각하면 실행될때 최 상단에 선언된것으로 인식해서 먼저 실행된다고 이해하시면 됩니다.

  호이스팅 예제 : https://simplejs.gitbook.io/olaf/09.-hoisting

2. 일반적으로 import는 모듈의 필요한 부분만 불러올수 있기 때문에 더 선호되며 성능적으로 우수합니다.
3. 하지만 Import 를 사용하기 위해서는 webpack이나 Babel등의 컴파일러가 필요하다는 단점이있습니다.

- cf. 컴파일러란 우리가 작성한 언어(JavaScript)를 기계가 알아들을 수 있는 기계어(어셈블리어)로 번역해 준다.

(기본 개발 용어 알아보기 : https://www.youtube.com/watch?v=GYmuQJiPeM4)

참고 자료

require과 Import의 차이점 : https://inpa.tistory.com/entry/NODE-📚-require-⚔️-import-CommonJs와-ES6-차이-1

비동기로 import하기 : https://inpa.tistory.com/entry/JS-📚-모듈-사용하기-import-export-정리?category=889099#브라우저(HTML)에서_모듈_사용_하기

▼ var | const | let 뭐가 다를까?

var는 다른 두가지와 변수 선언 방식에서 차이점이 있습니다.

차이점

var는 js의 초기 변수 선언방법으로 여러가지 문제점이 있었습니다.

1. 같은 변수명으로 선언을하게 되면 덮어쓰게됨

   : 이는 코드량이 많아진다면 어디서 잘못 된건지 찾기 힘든 치명적인 문제점이 있습니다.

2. 변수가 선언되지 않아도 참조 가능 (변수 호이스팅)

   : var는 function level scope이기 때문에 함수 밖에 선언된 var는 모두 전역변수가 됩니다.

   이는 메모리를 잡아먹어 성능적으로 낮아지며 변수명 중복으로 인한 오류를 야기합니다.

```
// var
var a;
a = "test1";
console.log(a); // test1

var a = "test1";
console.log(a); // test1

a = "hello1";
console.log(a); // hello1

var a = 1;
console.log(a); // 1
```

이를 보안하기 위해 js 기술 규격을 정하는 Ecma 인터네셔널이라는 기구에서 2015년 es6문법을 발표했는데요 이중 const, let이라는 상수 변수를 통해 var를 대체하게 되었습니다.

const는 변수 재선언, 재할당이 안되는 변수로 바꿀수 없는 값인 상수의 역할을 가집니다. 또한 선언과 할당을 동시에 해야합니다.

```
// const
const b; // error
b = test2;

const b = "test2";
console.log(b); // test

b = "hello2";
console.log(b); // error

const b = 2;
console.log(b); // error
```

let은 변수에 재할당이 가능하며 선언과 할당을 따로할 수 있습니다. 하지만 재선언은 안됩니다.

```
// let
let c;
c = "test3";
console.log(c); // test3

let c = "test3";
console.log(c); // test3

c = "hello3";
console.log(c); // hello3

let c = 3;
console.log(c); // error
```

더 깊게 들어간다면 변수 호이스팅, function level scope VS block level scope 등의 이슈가 있으니 참고하시길 바랍니다.

참고자료

js의 역사 : https://erokuma.tistory.com/entry/자바스크립트의-역사와-ECMAScript-대해

var의 문제점 : https://happycording.tistory.com/entry/let-const-란-왜-써야만-하는가-ES6

1-02 앨리스 토끼의 연금술

▼ funciton VS arrow funciton

차이점1

함수 호이스팅

function은 호이스팅에 의해 최상단에 선언되지만 arrow function은 호이스팅되지 않습니다.

```
// 호이스팅 차이
// function
func(); // hi~
function func() {
    return console.log("hi~");
}

// arrow function
arrFunc1(); // error
const arrFunc1 = () => console.log("hi~");

// arrFunc1(); // hi~

const arrFunc2 = () => {
    return console.log("hi~");
};
```

차이점2

this의 사용법

일반function은 호출한 객체를 가리키지만

arrow function 은 부모(상위 스코프)의 this를 가리킨다 (렉시컬 스코프)

```
// this 차이
// function
function fun() {
    this.name = "하이";
    return {
        name: "바이",
        speak: function () {
            console.log(this.name); // 바이
        },
    };
}

const fun1 = new fun();
fun1.speak(); // 바이

// arrow func
function arrFun() {
    this.name = "하이";
    return {
        name: "바이",
        speak: () => {
            console.log(this.name); // 하이
        },
    };
}

const fun2 = new arrFun();
fun2.speak(); // 하이
```

차이점3

생성자 함수로 사용여부

일반 함수에는 prototype 프로퍼티가 있기 때문에 생성자로 사용할 수 있지만

화살표 함수에는 prototype 프로퍼티가 없어 생성자로 사용할 수 없습니다.

```
// func
function fun() {
    this.num = 1234;
}

const funA = new fun();
console.log(funA.num); // 1234

//arrow func
const arrFun = () => {
    this.num = 1234;
};

const funB = new arrFun(); // Error
```

화살표함수를 사용하면 안되는 경우

1. 메소드를 화살표 함수로 정의하는 것은 지양해야합니다.

   : this가 메소드를 호출한 객체가 아닌 상위 스코프의 this를 가르키기 때문입니다.

2. 생성자를 만들면 안됩니다.

   : 화살표 함수는 prototype 프로퍼티를 가지고 있지 않기 때문에 사용할 수 없습니다.

참고자료

정리 : https://poiemaweb.com/es6-arrow-function

호이스팅 : https://bgpark.tistory.com/21

▼ for VS forEach

forEach는 ES6문법으로 ‘향상된for’으로 불립니다.

```
const testArr = [
    ["hello", "world"],
    [1, 2, 3],
];

// for
for (let i = 0; i < testArr.length; i++) {
    let arr = testArr[i];
    console.log(arr);
    for (let j = 0; j < arr.length; j++) {
        let tmp = arr[j];
        console.log(tmp);
    }
}
```

```
/*
[ 'hello', 'world' ]
hello
world
[ 1, 2, 3 ]
1
2
3
*/

console.log("\n-------------------------------------\n");

// forEach
testArr.forEach((arr) => {
    arr.forEach((tmp) => {
        console.log(tmp);
    });
});

/*
[ 'hello', 'world' ]
hello
world
[ 1, 2, 3 ]
1
2
3
*/
```

1. 동기(sync), 비동기(async)의 차이

   :for 는 동기방식으로 error가 나면 이후 이벤트를 진행하지 않지만 forEach는 비동기 방식으로 error가 나도 아래 이벤트 들을 진행합니다.

2. 성능 차이

   for 보다 forEach가 가변적인 배열이나 리스트의 크기를 구할 필요없어 복잡한 반복문에 사용하기 좋으며 수행속도도 빠릅니다.

3. 하지만 forEach는 반복문 내에 배열이나 리스크 값을 추가할 수 없고 읽기 전용으로 불러오기 때문에 데이터 수정이 불가능합니다.

참고자료

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=rlaalsdn456456&logNo=221818157118

map VS filter

▼ map VS filter

배열의 각각의 요소를 한번씩 순서대로 불러 반환값으로 새로운 배열을 생성합니다.

하지만 map은 각 요소를 콜백함수에 인자로 넣어 반환값을 새로운 배열로 반환하는 반면

filter는 각 요소들중 주어진 조건에 만족하는 값만을 새로운 배열로 반환합니다.

```
const arr = ["월요일", "화요일", "수요일", "목요일", "금요일"];

// map
const map = arr.map((item) => "출근하는 " + item);
console.log(map);
/*
"출근하는 월요일",
"출근하는 화요일",
"출근하는 수요일",
"출근하는 목요일",
"출근하는 금요일"
*/

// filter
const filter = map.filter((item, idx) => idx % 2 !== 0);
console.log(filter);
/*
"출근하는 화요일",
"출근하는 목요일",
*/
```

참고자료

map : https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/map

Filter : https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

▼ 다른 풀이방법은 없을까?

문제에서 요구하는 사항을 잘 파악하면 다른 풀이가 있지않을까요?! 😊

forEach 사용해보기

map 사용해보기

1-03 탐사선의 메세지

▼ key값에 해당하는 value값을 찾는 방법은?

점표기법과 대괄호 표기법 사용해서 접근할 수 있습니다.

```
const obj = {
    name: "엘리스",
    age: 3,
    text: "응애",
};

console.log(obj.name); // 엘리스
console.log(obj["age"]); // 3

const str = "text";
console.log(obj[str]); // 응애
```

▼ 다른 풀이 방법은 없을까?

map과 3항연산자를 사용한다면??

참고자료

3항연산자 :
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Operators/Conditional_Operator

2교시

1-04 옷 진열하기

▼ 다양한 JS메서드들

str.repeat(int) : str을 int번 반복해서 생성

str.trim() : 문자열의 양끝 공백을 없애는 메서드

arr.join() : 인자로 받은 값을 연결자로 array의 인자를 연결해 string을 만든다

▼ 다른 풀이 방법은 없을까?

입력값을 Obj로 만들어서 검색을 하는 방법과 arr로 만들어서 검색하는 방법 있어요! 어떻게 할 수 있을까요?

1-05 금 거래소

▼ 문자열을 다루는 방법은 뭐가 있을까?

str.substr(start[, length])

: 인덱스가 start 부터 length길이의 문자열을 반환

⇒ 사라지는 메서드!

str.substring(indexStart[, indexEnd])

: 문자열str의 indexStart로 부터 종료 인덱스 전 까지 문자열의 부분 문자열을 반환

str.slice(beginIndex[, endIndex])

: 문자열str의 indexStart로 부터 종료 인덱스 전 까지를 추출하면서 새로운 문자열을 반환

str.replace(regexp | substr, newSubstr | function)

: 문자열str안에 첫번째 인자로 받은 값을 두번째 인자로 받은 값으로 변경

str.split([separator] [, limit])

: 문자열str안에 separator값을 기준으로 나눠 arr로 반환

```
const str = "엘리스 SW엔지니어 3기 과정 레이서분들 안녕하세요";

console.log(str.substr(0, 3)); // 엘리스

console.log("\n" + "===========" + str + "===========");
console.log(str.substring(0, 3)); // 엘리스
console.log(str.substring(3)); // SW엔지니어 3기 과정 레이서분들 안녕하세요
console.log(str.substring(4, 10)); // SW엔지니어

console.log("\n" + "===========" + str + "===========");
console.log(str.slice(0, 3)); //  엘리스
console.log(str.slice(3)); //  SW엔지니어 3기 과정 레이서분들 안녕하세요
console.log(str.slice(4, 10)); // SW엔지니어

console.log("\n" + "===========" + str + "===========");
console.log(str.replace("안녕하세요", "화이팅~!")); // 엘리스 SW엔지니어 3기 과정 레이서분들 화이팅~!
console.log(str); // 엘리스 SW엔지니어 3기 과정 레이서분들 안녕하세요

console.log("\n" + "===========" + str + "===========");
console.log(str.split(" ")); // [ '엘리스', 'SW엔지니어', '3기', '과정', '레이서분들', '안녕하세요' ]
console.log(str.split("")); // 한글자씩 나눠서 arr로 반환
console.log(str.split("SW")); // [ '엘리스 ', '엔지니어 3기 과정 레이서분들 안녕하세요' ]
console.log(str.split()); // [ '엘리스 SW엔지니어 3기 과정 레이서분들 안녕하세요' ]
console.log(str); // 엘리스 SW엔지니어 3기 과정 레이서분들 안녕하세요
```

참고자료

substr() :
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/String/substr

substring()

: https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/String/substring

slice() : https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/String/slice

replace() :
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/String/replace

split() : https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/String/split

▼ 다른 풀이 방법은 없을까?

3자리마다 , 를 찍는 함수 만들기

toLocaleString('ko-KR') 사용하기

참고자료

toLocaleString() :
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/toLocaleString