9회차 과목평가 대비

원시 타입과 참조 타입의 분류

- 원시 타입 : Number, String, Empty Value, Boolean, undefined, null, Symbol
- 참조 타입 : Object - Array, Function, ..etc.

let, const, var 과 다른 것이 나오면 안된다.

const 같은 경우는 재할당 불가능

js array의 경우, 인덱스에 접근하는 법(음수는 안됐지만 최근에는 가능하다.)

배열(Array)

- 키와 속성들을 담고 있는 참조 타입의 객체
- 순서를 보장한다는 특징이 있음
- 주로 대괄호([])를 이용하여 생성하고, 0을 포함한 양의 정수 인덱스로 특정 값에 접근 가능
- 배열의 길이는 array.length 형태로 접근 가능 (배열의 마지막 원소는 array.length -1 로 접근 가능

array help method : 어떤 결과가 나오는지 (return 값이 obj면 obj 배열이면 배열)

→ 첫 번째 교제 133부터 보기

for…in, for… of 와 const

- for 문
  - for (let i = 0; i < arr.length; i++) {…}의 경우에는 최초 정의한 i를 재할당 하면서 사용하기 때문에 const를 사용하면 에러 발생
- for…in, for… of
  - 재할당이 아니라, 매 반복 시 해당 변수를 새로 정의하여 사용하므로 에러가 발생하지 않음

주관식 메서드 스펠링이랑 어디서 대문자가되는지 체크

선택관련 메서드 :

1. document.querySelector

- 제공한 선택자와 일치하는 element 한 개 선택
- 제공한 CSS selector를 만족하는 첫 번째 element 객체를 반환(없다면 null 반환)

2. document.quertSelectorAll(selector)

- 제공한 선택자와 일치하는 여러 element를 선택
- 매칭 할 하나 이상의 셀렉터를 포함한 유효한 CSS seletor를 인자(문자열)로 받은
- 제공한 CSS selector를 만족하는 NodeList를 반환

파이썬과 다른 Boolean

bollean() → false

bollean(0) → false

bollean(’’) → false

bollean(1) → true

bollean([]) → true

const a = {} const b= [] 뭐가 재할당 되는지 확인

const a = {}

const b = []

빈 것도 잘 만들어진다.

a → {}

b → []

==, === 눈에 익히기

1. 동등 연산자(==)
    - 두 연산자가 같은 값으로 평가되는지 비교 후 boolean 값을 반환
    - 비교할 때 `암묵적 타입변환` 통해 타입을 일치시킨 후 같은 값인지 비교
    - 두 피연산자가 모두 객체일 경우 메모리의 같은 객체를 바라보는지 판별
    - 예상치 못한 결과가 발생 할 수 있으므로 특별한 경우를 제외하고 사용하지 않음

```
const a = 1
const b = '1'

console.log(a==b)       // true
console.log(a==true)    // true

//자동 형변환 예시
console.log(8 * null)      // 0
console.log('5'- 1)        // 4
console.log('5' + 1)       // 51
console.log('five' * 2)    // NaM
```

2. 일치 연산자(===)
    - 두 피연산자의 값과 타입이 모두 같은 경우 true를 반환
    - 같은 객체를 가리키거나, 같은 타입이면서 같은 값인지를 비교
    - 엄격한 비교가 이뤄지며 `암묵적 타입 변환이 발생하지 않음` → 두 비교 대상의 타입과 값 모두 같은 지 비교하는 방식

```
const a = 1
const b = '1'

console.log(a===b)       // false
console.log(a===Number)    // true
```

addEventListener() 무조건 콜백 함수로 들어가야 한다.

event 중간에 막는 거 스팰링 확인하기.

- event.preventDefault()

eventlister(인자 뭐가 들어가는지 파악하기)

- EventTarget.addEventListener(type, listener[, options])

조작 관련 메서드(생성)

- document.createElement(tagName) - 작성한 tagName의 HTML 요소를 생성하여 반환
- Node.innerText
  - Node 객체와 그 자손의 텍스트 컨텐츠를 표현
  - 사람이 읽을 수 있는 요소만 남김
  - 즉, 줄 바꿈을 인식하고 숨겨진 내용을 무시하는 등 최정적으로 스타일링이 적용된 모습으로 표현됨/
- Node.appendChild()
  - 한 Node를 특정 부모 Node의 자식 NodeList 중 마지막 자식으로 삽임
  - 한번에 오직 하나의 Node만 추가할 수 있음
  - 추가된 Node 객체를 반환
  - 만약 주어진 Node가 이미 문서에 존재하는 다른 Node를 참조한다면 현재 위치에서 새로운 위치로 이동

조작 관련 메서드(삭제)

- Node.removeChild()
  - DOM에서 자식 Node를 제거
  - 제거된 Node를 반환

조작 관련 메서드(속성 조회 및 설정)

- Element.getAttribute(attributeName)
  - 해당 요소의 지정된 값(문자열)을 반환
  - 인자는 값을 얻고자 하는 속성의 이름
- Element.setAttribute(name, value)
  - 지정된 요소의 값을 설정
  - 속성이 이미 존재하면 값을 갱신, 존재하지 않으면 지정된 이름과 값으로 새 속성을 추가

map 필터 체이닝??

f-string `` 백틱 사용한다

settimeout, callback 갑자기 대문자 되는 거 없음

promise 이론 봐야한다

성공할 때 나오는 거 스팰링 .then링 확인하기

스코프

엑시노스 문법

페이지 일부분 바꾸는거

형변환에 대해서 알아보기

1 < 2 true

‘2’ < ‘3’ true

‘x’ < ‘y’ true

this 객체 안에 메서드 안에 디스는 객체를 가르키고

this 뭐시기 뭐시기

tag 만들 때 createElement

AJAX page의 전체가 아닌 일부분만 바꾼다

append와 appendchild의 차이점

- 어팬드는 모두 appendchild는 한 개

클래스 추가하는 방법

리스트를 직접 조작

setattribute

class name 직접 추가

게터와 세터?

게터 - 가져와서 직접 보는 함수, get이 들어가는 건 확인만 하는거

set 뭔가 변경을 가함 세터와 게터(attribute) 확인

class 설정을 어떻게 하는지

typeof 다 확인하기,

typeof null → ‘object’

typeof undifined → ‘undefined’

typeof NaN → number

typeof {} → object

typeof ‘g’/’2’ → NaN

자바스크립트 함수 인자 넉넉함 많으면 짜르고 모자르면 undefined

- 매개변수보다 인자의 개수가 많을 경우

```
const noArgs = function () {
  return 0
}

noArgs(1, 2, 3)  // 0

const twoArgs = function (arg1, arg2) {
  return [arg1, arg2]
}

twoArgs(1, 2, 3) //[1, 2]
```

- 매개변수보다 인자의 개수가 적을 경우

```
const threeArgs = function (arg1, arg2, arg3) {
  return [arg1, arg2, arg3]
}

treeArgs()             //[undefinde, undefinde, undefinde]
treeArgs(1)            //[1, undefinde, undefinde]
treeArgs(1, 2)         //[1, 2, undefinde]
```

함수 선언 방식 선언식과 표현식 (호이스팅 차이)

1. 함수 선언식

- 일반적인 프로그래밍 언어의 함수 정의 방식

```
function add(num1, num2) {
  return num1 + num2
}

add(2, 7) //9
```

2. 함수 표현식
   - 표현식 내에서 함수를 정의하는 방식, 함수 표현식은 이름을 생략한 익명함수로 정의 가능

```
const sub = function (num1, num2) {
  return num1 - num2
}

sub(7, 2) //5
```

   - 표현식에서 함수 이름을 명시하는 것도 가능, 다만 이 경우 함수 이름은 호출에 사용 되지 못하고 디버깅 용도로 사용됨

```
const mySub = function nameSub(num1, num2) {
  return num1 - num2
}

mySub(1, 2) // -1
namedSub(1, 2) //ReferenceError : namedSub is not defined
```

호이스팅

1. 선언식
   - 함수 선언식으로 의한 함수는 var로 정의한 변수처럼 호이스팅이 발생
   - 즉 함수 호출 이후에 선언해도 동작

```
add(2, 7)

function add (num1, num2) {
  return num1 + num2
}
```

2. 표현식
   - 반면 함수 표현식으로 선언한 함수는 함수 정의 전에 호출 시 에러 발생
   - 함수 표현식으로 정의된 함수는 변수로 평가되어 변수의 scope 규칙을 따름

```
add(7, 2)

const sub = function (num1, num2) {
  return num1 - num2
}
```

화살표 함수는 표현식에서 나왔다.

화살표 함수 (Arrow Function) 예시

```
const arrow1 = function (name) {
  return `hello, ${name}`
}

// 1. function 키워드 삭제
const arrow2 = (name) => { return `hello, ${name}` }

// 2. 인자가 1개일 경우에만 () 생략 가능
const arrow3 = name => { return `hello, ${name}` }

// 3. 함수 바디가 return을 포함한 표현식 1개일 경우에 {} & return 삭제 가능
const arrow4 = name => `hello, ${name}`
```

❖ 명확성과 일관성을 위해 항상 인자 주위에는 괄호(`()`)를 포함하는 것을 권장

화살표 함수 예외

화살표 함수 (Arrow Function) 응용

```
// 1. 인자가 없다면? () or _ 로 표시 가능
let noArgs = () => 'No args'
noArgs = _ => 'No args'


// 2-1. object를 return 한다면
let returnObject = () => { return { key: 'value'} }  // return 을 명시적으로 적어준다.


// 2-2. return을 적지 않으려면 괄호를 붙여야 함
returnObject = () => ({ key: 'value' })
```

this가 등장 할 때 화살표 화살표 함수 달랐음

json도 한번 보기 약자 파악하기

- JavaScript Object Notation
- Key-Value 형태로 이루어진 자료 표기법
- JavaScript의 Object와 유사한 구조를 가지고 있지만 Object는 그 자체로 타입이고, JSON은 형식이 있는 “문자열”
- 즉, JSON을 Object로 사용하기 위해서는 변환 작업이 필요

JSON 변환

```
const jsObject = {
  coffee: 'Americano',
  iceCream: 'Cookie and cream',
}
```

```
// Object -> JSON

const objToJson = JSON.stringify(jsObject)

console.log(objToJson)  // {"coffee":"Americano","iceCream":"Cookie and cream"}
console.log(typeof objToJson)  // string
```

```
// JSON -> Object

const jsonToObj = JSON.parse(objToJson)

console.log(jsonToObj)  // { coffee: 'Americano', iceCream: 'Cookie and cream' }
console.log(typeof jsonToObj)  // object
```

❖ Django와 같은 API 서버에서 JSON을 응답한 것을 받아 다음과 같이 변환해야 하는 것 !

161

const bookshop = {books, magazines } ':'' 안써도

- object에 넣을 때 키 밸류 같으면 밸류 생략 가능

파이썬: for 문 밖에서 i를 찍을 때 마지막 인자가 출력됨(죽는 거 아님)

js: for 문 밖에서 i를 찍면 에러발생

switch 케이스 결과 값 (break 생각)

호이스팅 구현 방법

console.log(c)

var c = 3 → undefinde

let d; → undefined

const v; → 에러

settimeout - > 원리 파악하기 0이여도 바로 실행 안됨