[Part 01. ECMAScript 6 기초]

Contents.

1. JS와 ECMAScript Version 4. const 키워드

2. ES6의 새로운 기능

3. let 키워드

5. Template Strings

6. Ouiz

7. 구조분해 할당

8. Quiz Drill

1. JS와 ECMAScript Version

1) JavaScript 탄생 → 1995년 Brendan Eich(브랜든 아이크)에 의해 발명되었습니다.

1997년 ECMA 표준이 되었으며, ECMAScript가 Official name(오피셜 네임, 공식 이름)입니다.

• 참고. ECMA는 European Computer Manufacturers Accociation, 유럽 컴퓨터 제조업 협회)으로 컴퓨터 하드웨어, 통신 및 프로그래밍 언어의 표준을 개발하는 비영리 조직입니다.

ECMA 관련 URL. https://ecma-international.org/

https://developer.mozilla.org/ko/docs/Glossary/ECMA

2) ECMAScript Editions(에크마 스크립트 출판)

구분	기타표현	Official Name		
오리지널	ES1	ECMAScript 1 (1997, 에크마 스크립트)		
자바스크립트	ES2	ECMAScript 2(1998)		
	ES3	ECMAScript 3(1999)		
첫 번째 개정판	ES5	ECMAScript 5(2009) → 첫 번째 주요 개정판(Revision), "strict mode", String 관련 메서드, 등		
두 번째 개정판		ECMAScript 2015 → 두 번째 주요 개정판(Revision), let, const, Template Strings, 등		
연간 네이밍 버전	ES6 또는 ESNext	ECMAScript 2016 ECMAScript 2017 ECMAScript 2018 ECMAScript 2023		

- 참고. ES4, ECMAScript 4는 출시하지 않고 폐기함.
- 2. ES6 의 새로운 기능
- ① let, const 키워드
- ② Template Strings(템플릿 스트링즈)
- ③ Destructuring(디스트럭처링, 구조분해 할당)
- ④ Arrow Functions(애로우 펑션, 화살표 함수)
- ⑤ Spread Operator(스프레드 오퍼레이터, 펼침 연산자)
- ⑥ For/Of Loop(for/of 순환 제어문)
- ⑦ For/In Loop(for/in 순환 제어문)
- ⑧ Map, Set Objects(맵 객체, 집합 객체)
- ⑨ Classes, Promises, Symbol (클래스, 프라미스, 심벌)
- ⑩ Function Rest Parameter(함수 나머지 매개변수)

- 3. let 키워드 → ES6부터 변수를 생성할 때 기존에 사용하던 var 키워드를 대신하여 사용할 수 있습니다.
 - let 키워드를 사용하여 생성된 변수는 다음과 같은 기능이 적용됩니다.
 - ① 같은 레벨의 변수 중복 선언 불가 ② 호이스팅 제한
 - ③ 변수 공유 문제 개선
- ④ 블록 스코프 사용

1) 같은 레벨의 변수 중복 선언 불가

변수 중복 선언 허용	변수 중복 선언 불가	
<script></td><td colspan=2><script></td></tr><tr><td>var x = 5;</td><td colspan=2>let x = 5;</td></tr><tr><td>var x = 2.4;</td><td colspan=2>let x = 2.4;</td></tr><tr><td></script>		
오류 없음	오류 발생	

• 참고. 같은 레벨은 함수 블록, 제어문 블록 등을 사용한 구간을 의미합니다.

2) 호이스팅(Hoisting, 끌어올림) 제한

호이스팅 허용	호이스팅 불가
<pre><script> console.log(x); var x = 3; </script></pre>	<pre><script> console.log(x); let x = 3; </script></pre>
오류 없음	오류 발생

3) 변수 공유 문제 개선

잘못된 출력(출력 결과 모두 동일)	정상 출력
<script></th><th><script></th></tr><tr><td>for (var i=0; i<5; i++) {</td><td>for (let i=0; i<5; i++) {</td></tr><tr><td><pre>setTimeout(function(){</pre></td><td><pre>setTimeout(function(){</pre></td></tr><tr><td><pre>console.log(i);</pre></td><td><pre>console.log(i);</pre></td></tr><tr><td>}, 1000);</td><td>}, 1000);</td></tr><tr><td>}</td><td>}</td></tr><tr><td></script> <td></td>	
	1

• 참고. setTimeout(함수, 시간); 메서드는 지정된 밀리세컨즈 단위의 시간이 흐른 후에 함수를 실행하는 메서드입니다. setInterval() 메서드와 다른 점은 반복없이 1회만 실행합니다.

4) 블록 스코프 사용

```
var은 함수 스코프만 허용
<script>
                         <script>
 var x = 5;
                          var x = 5;
                          if (x > 1) {
 function test() {
  var x = 10;
                            var x = 10;
  console.log(x);
                            console.log(x);
 console.log(x); // 5
</script>
                        </script>
```

console.log(x); // 10 • 참고. let은 함수 스코프 및 제어문 스코프, 블록 스코프가 모두 적용됩니다.

4. const 키워드 → 재입력 안됨 + 선언과 동시에 초기화 해야 함.

단, 변수 자체의 값은 재입력 할 수 없지만, 구성요소의 값은 초기화 할 수 있음.

재초기화 안됨	선언과 동시에 초기화 필요	구성요소는 변경할 수 있음.
const PI = 3.141592; PI = 3.14;	const PI; PI = 3.141592;	<pre>const cars = ["Saab", "Volvo", "BMW"]; cars[0] = "Kia";</pre>
		console.log(cars);
오투	- - - 발생	"Saab" → "Kia" 로 변경됨

- 5. Template Strings(템플릿 스트링즈) → Back-Tics(백틱, 억음부호, `)를 사용한 문자열 표현방법입니다. 아래 4가지 방식이 적용됩니다.
- ① 문자열에 따옴표 혹은 홑따옴표가 아닌 백틱을 사용합니다.
- ② 백틱 내용 내부에 홑따옴표나 따옴표를 그대로 사용할 수 있습니다.
- ③ 백틱 내용 내부에 줄바꿈이 있더라도 연결연산자 없이 사용할 수 있습니다.
- ④ 백틱 내용 내부에 보간(interpolation, 인터폴레이션)법을 적용할 수 있습니다. 보간 또는 보간법은 문자열 내부에서 스트링(\$) 기호와 중괄호({ })로 \${변수}, \${표현식} 형식으로 사용하는 방법입니다. 표현식은 문자열, 변수 및 연산식이 적용된 구문을 의미합니다.

[Template Strings 사용법]

항목	내용		
8 T	Template Strings 적용	기존 방식	
① 문자열에 백틱 사용	<pre>let txt1 = `Hello World`; console.log(txt1);</pre>	"Hello World";	
② 백틱 내용 내부에 따옴표 사용	let txt2 = `선생님은 "오늘 주번?"이라고 말했다.`; console.log(txt2);	"선생님은 \"오늘 주번?\"이라고 말했다.";	
③ 백틱 내용 줄바꿈	let txt3 = `여름 수박 겨울 딸기`; console.log(txt3);	let txt3 = "여름 수박\n" + " 겨울 딸기";	
④ 백틱 내부에 보간법을 보간법을 사용한 표현식	let num1 = 2; let num2 = 3; let txt = `토마토`; let res = `\${txt + " " + num1 * num2} 개`; console.log(res);	let res = txt + " " + (num1*num2) + " 개";	

- 참고. Template Strings를 Template Literals(템플릿 리터럴즈) 또는 String Templates(스트링 템플릿츠) 라고도 합니다. 표현식 내부의 따옴표는 홑따옴표(') 또는 백틱(`)을 사용할 수 있습니다. let res = `\${txt + ' ' + num1 * num2} 개`; 또는 let res = `\${txt + ` ` + num1 * num2} 개`;
- 6. Quiz. 2 이상의 숫자를 입력한 후 어떤 숫자의 배수인지 템플릿 스트링을 사용하여 출력하세요.
- 홈디렉토리명 → p01/quiz html파일명 → /multiple.html JS파일명 → 임의지정

출력결과1		출력결과2
숫자입력	27 결과확인	숫자입력 8 결과확인
입력하신 숫자 27은(는) 3의 배수입니다.		입력하신 숫자 8은(는) 2의 배수입니다.

- 7. 구조분해 할당(Destructuring Assignment 또는 Destructuring, 디스트럭처링 어사인먼트, 비구조 할당)
 - → 배열 값과 객체의 속성 값을 변수에 쉽게 할당하는 방법입니다.
- 1) 객체 구조분해 할당(Object Destructuring)
- → 속성(=키, 이름)과 속성값으로 구성된 커스텀 객체를 동일한 속성명으로 여러 개의 속성을 할당할 수 있습니다.

```
① 형식 커스텀 객체(Object) 구조분해 할당(Destructuring Assignment)

const person = {
  firstName: "John",
  lastName: "Wick",
  age: 35
  };
  // 객체 구조분해 할당(Destructuring Assignment)

let { firstName, age } = person;
  // 변수명1 변수명2 객체명

// 객체 구조분해 할당에 사용하는 변수는 중괄호를 사용해야 합니다.
  // 변수명은 객체의 키(Key, 속성)와 동일한 이름을 사용해야 합니다.
  // 변수명은 객체의 키 순서와 상관없이 사용할 수 있습니다.

// 속성: 속성값
```

```
② 보기 const person = {
    firstName: `John`,
    lastName: `Wick`,
    age: 35
};
let { firstName, age } = person;
console.log(`${firstName +' '+ age}`); // Template String
```

• 속성 값에 따옴표, 홑따옴표, 백틱 사용가능

③ 누락되거나 정의되지 않은(undefined) 속성의 기본값을 설정할 수 있습니다.

```
const person = { firstName: "John", lastName: undefined, age: 35 };
let { firstName, lastName="Wick", country="US" } = person;
console.log(`${firstName + `` + lastName} 은 ${country} 사람`);
```

④ 객체 속성에 별칭(alias, 얼라이어스)을 사용할 수 있습니다.

```
const person = { firstName: `John`, lastName: `Wick`, age: 35 };
let { firstName: f, lastName: l="Wick", country: c="US" } = person;
console.log(`${f + ` ` + l} 은 ${c} 사람`);
```

- 2) 배열 구조분해 할당(Array Destructuring)
- → 배열을 배열타입의 구조에 분해 할당할 수 있습니다. 배열요소의 이름은 임의 지정할 수 있으며 순서대로 적용됩니다.

```
① 형식 배열 구조 분해할당

const fruit = [`Apple`, `Banana`, `Mango`, `Lemon`]; let [n1, n2] = fruit;
```

- 변수 구조에 배열기호(=대괄호)를 사용하며 배열의 순서대로 할당됩니다.
- 변수 이름은 임의 지정할 수 있으며 서로 다른 이름을 사용할 수 있습니다. 보기. let [a, k2, m_] = fruit;
- ② 배열 값 건너뛰기
 배열
 구조 분해할당

 const fruit = [`Apple`, `Banana`, `Mango`, `Lemon`];
 let [n1,,,n2] = fruit;
 - 원하는 인덱스의 순서만큼 쉼표를 기재합니다. 즉 아래와 같이 적용됩니다. 변수 → n1, , n2 할당 값 → `Apple`, `Banana` `Mango` `Lemon`

③ 인덱스로 배열 위치 값 지정

배열	구조 분해할당
<pre>const fruit = [`Apple`, `Banana`, `Mango`, `Lemon`];</pre>	let { [0]:a, [2]:b } = fruit;

- 주의. 배열의 구조 분해 할당에 사용되는 변수 구조는 대괄호([])를 사용하지만 인덱스로 배열 위치 값을 Distructuring(디스트럭처링, 분해 할당)할 경우에는 중괄호를 사용합니다.
- ④ Rest(레스트) 속성 → 나머지 속성(...)을 사용하여 남은 배열을 처리할 수 있습니다.

```
const numAry = [10, 20, 30, 40, 50];
let [a, b, ...rest] = numAry; // ... 기호는 필수, rest는 다른 이름 사용가능
console.log(`a 값 : ${a}`);
console.log(`b 값 : ${b}`);
console.log(`배열의 나머지 값 : ${rest}`);
```

⑤ 문자열 구조분해 할당 → 문자열에서 한 글자씩 분해하여 할당할 수 있습니다.

문자열 분해 할당 사용법1	문자열 분해 할당 사용법2
<pre>let txt = `lemon`;</pre>	<pre>let txt = `lemon`;</pre>
let [a1, a2, a3] = txt;	let [s] = txt;
<pre>console.log(`a2 : \${a2}`);</pre>	<pre>console.log(`s : \${s}`);</pre>
// e 출력	// l, e, m, o, n 출력
let [a1, a2, a3] = txt; console.log(`a2 : \${a2}`);	<pre>let [s] = txt; console.log(`s : \${s}`);</pre>

- 구조 분해는 문자열뿐만 아니라 모든 반복 가능한 객체에 사용할 수 있습니다.
- 8. Quiz Drill. 귤의 개수와 상품 크기를 선택하여 상품 포장에 상자의 개수와 남은 상품 수를 템플릿 리터럴을 사용하여 출력하시오. (단, 상품수는 20개 이상을 입력할 수 있도록 유효성 검사를 실시합니다.)
 - 포장단위
 - → 대과는 1상자에 8개를 포장할 수 있습니다.
 - → 중과는 1상자에 11개를 포장할 수 있습니다.
 - → 소과는 1상자에 15개를 포장할 수 있습니다.

출력결과1	출력결과2	출력결과2
개수 입력(20개 이상 입력) 27	개수 입력(20개 이상 입력) 35	개수 입력(20개 이상 입력) 16
◉ 대과 ○ 중과 ○ 소과	○ 대과 ○ 중과 ◉ 소과	○ 대과 ● 중과 ○ 소과
결과확인	결과확인	결과확인
대과 3상자를 포장하고 3개 남습니다.	소과 2상자를 포장하고 5개 남습니다.	입력오류! 20개 이상 입력해주세요.

- 내용 끝.