Javascript 복습 심화

15번째 세션

NEXT X LIKELION 양효령

목차

자바스크립트

웹페이지에 생동감을 불어넣기 위해 만들어진 프로그래밍 언어

자바스크립트 엔진이 있는 모든 디바이스에서 동작

- 브라우저: V8(Chrome, Opera), SpiderMonkey(Firefox) 등
- 브라우저 뿐 아니라 서버에서도 실행 가능 -> Node.js

자바스크립트(참고)

1. **브라우저의 자바스크립트**
   - 페이지에 새로운 HTML을 추가하거나 기존 HTML, 혹은 스타일 수정하기
   - 마우스 클릭이나 포인터의 움직임, 키보드 키 눌림 등과 같은 사용자 행동에 반응하기
   - 네트워크를 통해 원격 서버에 요청을 보내거나, 파일 다운로드, 업로드하기(AJAX나 COMET과 같은 기술 사용)
   - 쿠키를 가져오거나 설정하기. 사용자에게 질문을 건네거나 메시지 보여주기
   - 클라이언트 측에 데이터 저장하기(로컬 스토리지)

2. **서버의 자바스크립트**
   - 보안을 위해서 브라우저에서 자바스크립트 기능에 제약 걸어 놓음
   - 서버에서는 브라우저에서 제약을 걸어 놓은 작업 가능

자바스크립트 - 강점

1. HTML/CSS와 완전히 통합할 수 있음
2. 간단한 일은 간단하게 처리할 수 있게 해줌
3. 모든 주요 브라우저에서 지원하고, 기본 언어로 사용됨
4. 서버나 모바일 앱을 만들 수 있음

코어 자바스크립트

NEXT X LIKELION

코어 자바스크립트

실행 환경에 독립적인 자바스크립트 문법

자바스크립트는 자바스크립트 엔진이 있는 환경이라면 어디서든 실행 가능

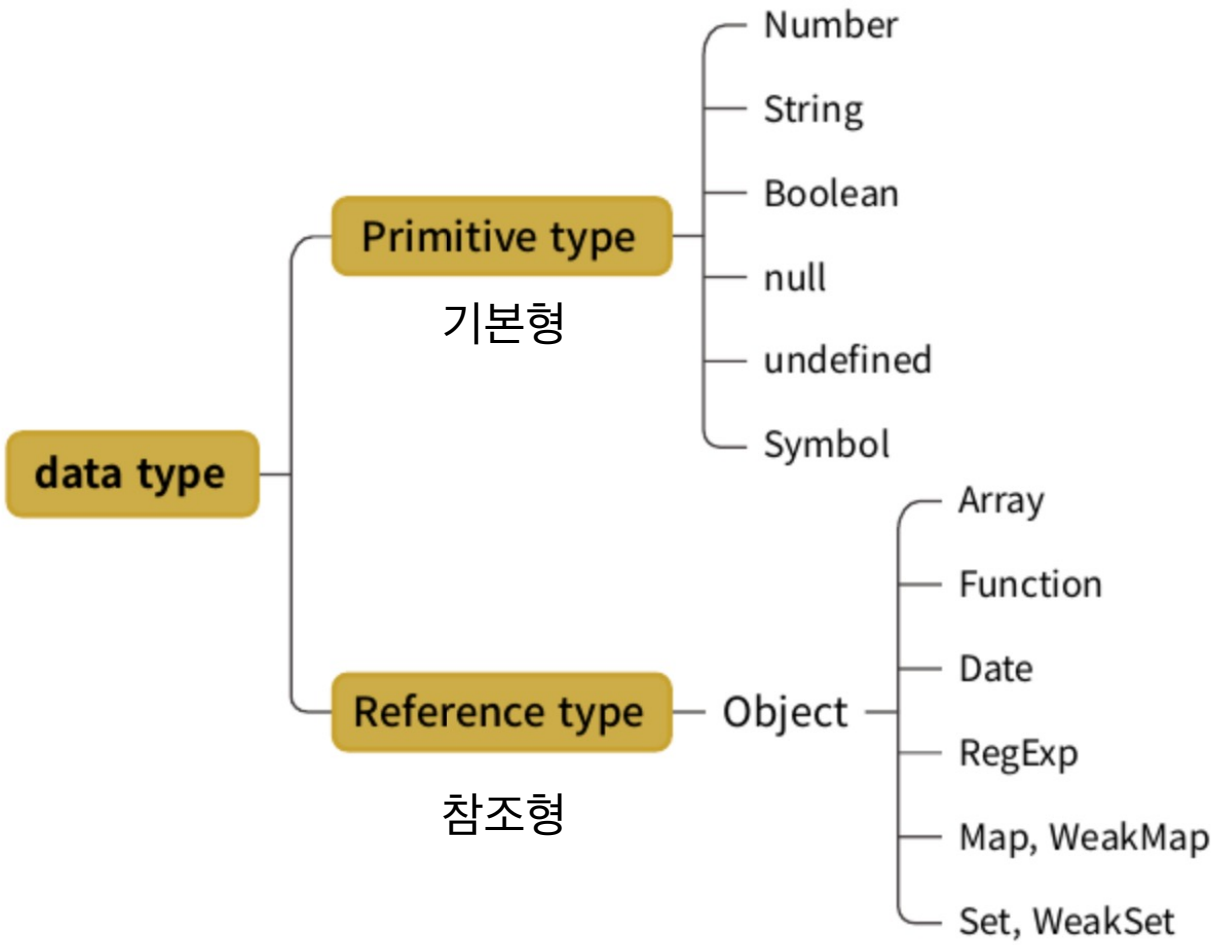
공통적으로 쓰이는 코어 자바스크립트 문법을 제대로 알아야 함!

데이터 타입(자료형)

자료형	상세 설명
숫자형(Number)	정수, 소수점 숫자 등을 포함한 숫자를 나타낼 때 사용
문자형(String)	빈 문자열 or 문자들로 이뤄진 문자열을 나타낼 때 사용 * 단일 문자를 나타내는 별도의 자료형은 없음.
불린형(Boolean)	true, false 를 나타낼 때 사용
null	알 수 없는 값을 나타냄
undefined	할당되지 않은 값을 나타냄
객체형(Object)	복잡한 데이터 구조를 표현할 때 사용할 때 사용

+ bigint, 심볼형, Function, Array 등등…

데이터 타입(자료형)


data type
Primitive type
기본형
Number
String
Boolean
null
undefined
Symbol
Reference type
참조형
Object
Array
Function
Date
RegExp
Map, WeakMap
Set, WeakSet

데이터 타입(자료형)

1. **기본형**
   - 문자열이든 숫자든 한 가지만 표현할 수 있음
   - 할당이나 연산 시 값이 담긴 주솟값을 바로 복제
   - 불변성(immutability)

2. **참조형**
   - 할당이나 연산 시 값이 담긴 주솟값들로 이루어진 묶음을 가리키는 주솟값을 복제
   - 데이터 자체가 아닌 내부 프로퍼티를 변경할 때 가변

```
const object = {a: 1, b: 2}
undefined
object = {a: 1, b:2, c:3}
▸ Uncaught TypeError: Assignment to constant variable.
    at <anonymous>:1:8
object.a = 2
2
console.log(object)
▸ {a: 2, b: 2}
```

형 변환

- **함수와 연산자에 전달되는 값은 대부분 적절한 자료형으로 자동 변환됨**

  Ex) alert가 전달받은 값의 타입과 관계없이 문자열로 자동 변환해서 보여줌,

  수학 관련 연산자가 전달받은 값을 숫자로 변환함

```
const a = '1';
const b = 2;
console.log(a / b);
```

```
0.5
```

- **의도를 가지고 명시적으로 변환할 수도 있음**
  - 문자형으로 변환: String(value)
  - 숫자형으로 변환: Number(value)
  - 불린형으로 변환: Boolean(value)

전달받은 값	형 변환 후
`undefined`	`NaN`
`null`	`0`
`true / false`	`1 / 0`
`string`	전달받은 문자열을 "그대로" 읽되, 처음과 끝의 공백을 무시합니다. 문자열이 비어있다면 `0` 이 되고, 오류 발생 시 `NaN` 이 됩니다.

전달받은 값	형 변환 후
`0`, `null`, `undefined`, `NaN`, `""`	`false`
그 외의 값	`true`

https://ko.javascript.info/type-conversions

변수와 상수

var vs let, const

	재 선언	재 할당
var	O	O
let	X	O
const	X	X

오래된 변수, 상수 선언 키워드

모던한 변수 선언 키워드

모던한 상수 선언 키워드

변수와 상수 – 명명 규칙

1. **변수, 함수**
   - 변수명에는 오직 문자, 숫자, $, _ 만 들어갈 수 있음
   - 첫 글자가 숫자일 수 없음
   - 보통 camelCase를 사용함 (첫 단어를 제외한 각 단어의 첫 글자를 대문자로 작성)
2. **상수**
   - 기억하기 힘든 값을 보통 상수로 사용함
   - 보통 대문자 SNAKE_CASE 사용함
3. **기타**
   - CSS class: kebab-case
   - 파일명: kebab-case

컨벤션이라 규칙을 정하기 나름이지만 보통 이렇게 씁니다!

변수와 상수 - 심화

1. 호이스팅
   - 인터프리터가 변수와 함수의 메모리 공간을 선언 전에 미리 할당하는 것
   - var로 선언한 변수의 경우 호이스팅시 undefined로 변수를 초기화함.
   - 반면 let과 const로 선언한 변수의 경우 호이스팅시 변수를 초기화하지 않음
   - https://developer.mozilla.org/ko/docs/Glossary/Hoisting

```
getCatName("뽀또");

function getCatName(name) {
  console.log("제 고양이의 이름은 " + name + "입니다");
}
```

```
제 고양이의 이름은 뽀또입니다
```

```
getCatName("야옹이");

const getCatName = (name) => {
  console.log(`제 고양이 이름은 ${name} 입니다`);
};
```

```
▶ Uncaught ReferenceError: getCatName is not defined
    at <anonymous>:1:1
```

```
console.log(name);

var name = "John";
```

```
undefined
```

```
console.log(name);

const name = "John";
```

```
▶ Uncaught ReferenceError: name is not defined
    at <anonymous>:1:13
```

변수와 상수 - 심화

2. 스코프

- 변수에 접근할 수 있는 범위
- 밖에선 안을 볼 수 없고, 안에서는 밖을 볼 수 있음
- 전역 스코프와 지역 스코프(함수 스코프, 블록 스코프)
- var는 함수 또는 전역 스코프만 형성
- let, const는 블록 스코프도 형성
- https://learnjs.vlpt.us/useful/08-scope.html

변수와 상수 - 심화

호이스팅, 스코프를 잘 모르더라도 결론은

변수와 함수를 가급적 코드 상단부에서 선언하자

var 대신 let 과 const를 쓰자

연산자 – 비교 연산자

[기본 수학 연산자]

사용 상황	연산자
크다/작다	< , >
이상/이하	>= , <=
같음(동등)	== (등호 2개)
다름(부등)	!=

[일치 연산자(===)]

JS는 약타입 언어,

```
console.log( 0 == false ); // true
```

피연산자의 묵시적 형변환이 일어나기 때문에 나타나는 현상

=== 를 통해 type의 일치 여부까지 검사한다.

```
console.log( 0 === false ); // false
```

연산자 – 삼항 연산자

숫자 0, 빈 문자열"", null, undefined, NaN은 불린형으로 변환 시 모두 false가 됩니다.

이런 값들은 '**falsy**(거짓 같은)' 값이라고 부릅니다.

이 외의 값은 불린형으로 변환시 true가 되므로 '**truthy**(참 같은)' 값이라고 부릅니다.

```
let result = condition ? value1 : value2;
```

평가 대상인 condition이 truthy라면 value1이, 그렇지 않으면 value2가 반환

연산자 – 삼항 연산자

언제 if문을 쓰고 언제 삼항 연산자를 쓰는 거죠?

가독성이 좋은 방법을 선택하면 됩니다!

보통 조건에 따라 반환값이 다를 때 삼항 연산자를,

조건에 따라 표현식이 실행되어야 할 경우는 if문을 씁니다.

그리고 중첩 삼항 연산자는 웬만하면 쓰지 맙시다…!

(코드 짠 사람도 못 알아봄)

```
let message =
age < 3
? "아기야 안녕?"
: age < 18
? "안녕!"
: age < 100
? "환영합니다!"
: "나이가 아주 많으시거나, 나이가 아닌 값을 입력 하셨군요!";
```

연산자 – 논리 연산자

|| (OR)

- 인수 중 하나라도 true이면 true 반환
- **첫 번째 truthy를 찾음**
  - 왼쪽에서부터 피연산자를 평가함
  - 각 피연산자를 불린형으로 변환함
  - 그 값이 true이면 **연산을 멈추고 해당 피연산자의 변환 전 원래 값을 반환**
  - 피연산자가 모두 false로 평가되는 경우 마지막 피연산자를 반환
- 참고: nullish 병합 연산자 '??'

  https://ko.javascript.info/nullish-coalescing-operator

연산자 – 논리 연산자

&& (AND)

- 인수가 모두 true일 때 true 반환
- **첫 번째 falsy를 찾음**
  - 왼쪽에서부터 피연산자를 평가함
  - 각 피연산자를 불린형으로 변환함
  - 그 값이 false이면 **연산을 멈추고 해당 피연산자의 변환 전 원래 값을 반환**
  - 피연산자가 모두 true로 평가되는 경우 마지막 피연산자를 반환
- &&가 ||보다 우선순위가 높음

연산자 – 논리 연산자

! (NOT)

- 인수를 하나만 받음
- **피연산자를 불린형으로 변환함**
- **변환된 값의 역을 반환함**
- 두 번 연달아 사용하면 Boolean을 사용한 것과 같은 결과를 도출함
- &&, || 보다 !이 우선순위가 가장 높음

연산자 – 논리 연산자

```
const a = ""; // falsy
const b = " "; // truthy
const c = 0; // falsy
const d = null; // falsy
const e = []; // truthy

console.log(a || b || c || d || e); // " "
console.log(a || c || d); // null

console.log(a && b && c && d && e); // ""
console.log(b && e); // []

console.log(!a); // true
console.log(!!a); // false
```

퀴즈!

```
!(b && !c) || !d && e
```

반복문

- for
- for in
- for of
- while
- do … while
- forEach, map, reduce 등등

조건문

- If 문
- switch case
- 삼항 연산자 () ? () : ();
- 짧은 조건문 -> 단축 평가
  - a || b → a 또는 b 둘 중 하나라도 참일 경우 실행
  - a && b → a 와 b 모두 참이어야 실행

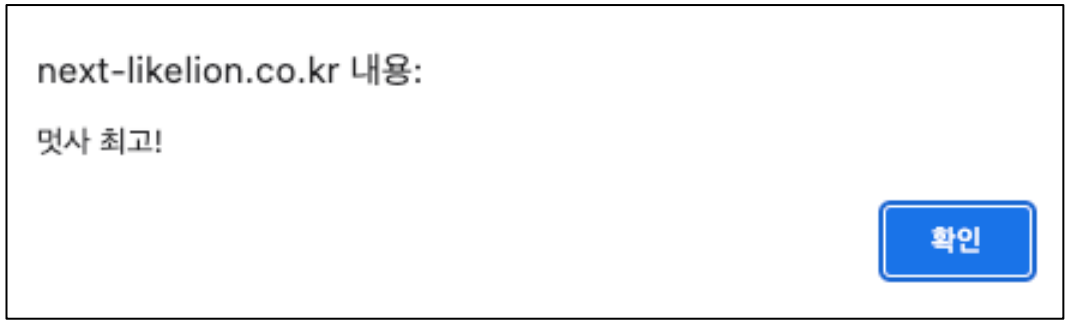
가독성!!!

alert, prompt, confirm

1. Alert

- 확인 버튼을 누를 때까지 창이 떠 있고 다른 행동을 할 수 없음

```
alert("멋사 최고!");
```



alert, prompt, confirm

2. Prompt

- 프롬프트 대화 상자에 사용자가 원하는 값을 입력하고 확인을 누를 수 있음
- 입력하지 않거나, 취소를 누르면 null return 하고 대화상자 빠져나감
- 두번째 매개변수는 선택값

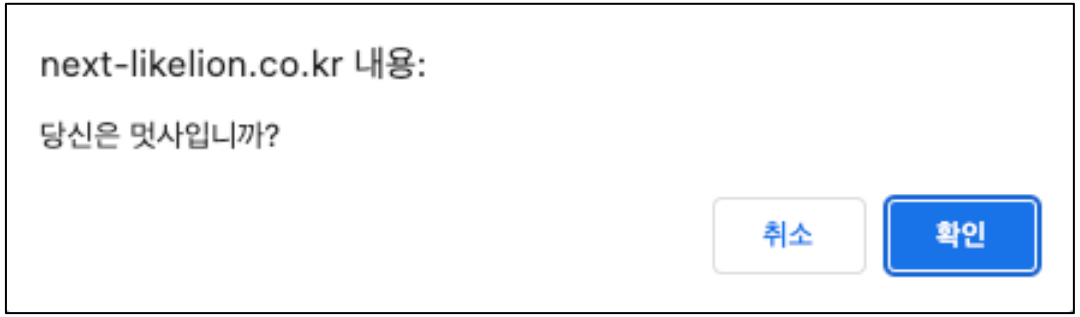
```
const age = prompt("연세가 어떻게 되시죠?", 20);
alert(`당신은 ${age}살 입니다.`);
```

alert, prompt, confirm

3. Confirm

- 매개변수로 받은 질문과 확인, 취소 버튼이 있음
- 사용자가 확인을 누르면 true, 그 외는 false return

```
const isNext = confirm("당신은 멋사입니까?");
alert(isNext);
```



alert, prompt, confirm

모달 창: 메시지가 있는 작은 창, 페이지의 나머지 부분과 상호작용이 불가능함

1. 위 함수들은 모두 모달 창을 띄워주는데, 모달 창이 떠 있는 동안은 스크립트의 실행이 일시 중단됨

   -> 사용자가 창을 닫기 전까진 나머지 페이지와 상호 작용이 불가능함
2. 모달 창의 위치는 브라우저가 결정하고, 모달 창의 모양은 브라우저마다 다름

   -> 개발자가 창의 모양을 수정할 수 없음

실습 – 간이 로그인 구현하기

프롬프트(prompt) 대화상자를 이용해 간이 로그인 창을 구현해보세요.

사용자가 ”likelion"를 입력하면 비밀번호를 물어보는 프롬프트 대화상자를 띄워주세요.

이때 아무런 입력도 하지 않거나 Esc를 누르면 "취소되었습니다."라는 메시지를 보여주세요. 틀린 비밀번호를 입력했다면 "인증에 실패하였습니다."라는 메시지를 보여주세요.

비밀번호 확인 절차는 다음과 같습니다.

- 맞는 비밀번호 ”220704"를 입력했다면 "환영합니다!"라는 메시지를 보여주세요.
- 틀린 비밀번호를 입력했다면 "인증에 실패했습니다."라는 메시지를 보여주고 다시 비밀번호를 입력받으세요.
- 빈 문자열을 입력하거나 입력을 취소했다면 "취소되었습니다."라는 메시지를 보여주세요.

*** 꼭 사용할 것: alert, prompt, 논리 연산자**

Object

- Python의 dict와 같음
- 참조에 의해(by reference) 저장되고 복사된다는 특징.

```
// 객체를 선언할 때

let user = {
    name: "Lion",
    age: 22
};


// 참조할 때 (1)점 표기법, (2)대괄호 표기법

user.name

user["name"]


// 동적으로 추가 및 수정 가능

user.name = "Tiger"

user["name"] = "Tiger"
```

없는 key에 접근했을 때

1. 점 표기법 -> 에러
2. 대괄호 표기법 -> undefined

Array

- Python의 list와 같음

- [대표적인 메서드들]
- arr.push() - Array 맨 끝에 요소 추가
- arr.pop() - Array 맨 끝 요소 제거
- arr.shift() - Array 맨 앞 요소 제거
- arr.unshift() - Array 맨 앞에 요소 추가

- [그 외에도]
- arr.slice([start], [end])
- arr.splice()

```
// push 메서드 활용 예시

let meotjaengi = [];

meotjaengi.push("!"); // Array "meotjaengi"에 !가 추가됨
meotjaengi.append("?"); // Error (Python과 다릅니다!)


// slice 메서드 활용 예시

let meotjaengi2 = ["L", "I", "O", "N"];

console.log(meotjaengi2.slice(1,3)); // ['I', 'O']

// (인덱스가 1인 요소(start)부터 3인 요소 이전까지(end - 1)의
"SubArray(하위 배열)" 형태로 복사)
```

Array

다양한 메서드들 잘 활용해보자!
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array

```
const list = ["a", "b", "c", "d"];

// forEach() 메서드는 주어진 함수를 배열 요소 각각에 대해 실행
list.forEach((item, index) => console.log({ item, index }));

// map() 배열 내의 모든 요소 각각에 대하여 주어진 함수를 호출한 결과를 모아 새로운 배열을 반환
const mappedList = list.map((item, index) => {
return `${index}: ${item}`;
});

console.log(mappedList); // [ '0: a', '1: b', '2: c', '3: d' ]

// filter() 메서드는 주어진 함수의 테스트를 통과하는 모든 요소를 모아 새로운 배열로 반환
const filteredList = list.filter((item) => item === "a" || item === "c");

console.log(filteredList); // [ 'a', 'c' ]

// includes() 메서드는 배열이 특정 요소를 포함하고 있는지 판별
console.log(list.includes("a")); // true
console.log(list.includes("e")); // false
```

find(), findIndex(), reduce(), sort() 등등…

Optional Chaining

?.의 앞의 평가 대상이 undefined나 null이면 평가를 멈추고 undefined를 반환

```
let user;

console.log(user.id); // error

// 방법 1: && 연산자 사용
console.log(user && user.id); // undefined

// 방법 2: optional chaining 사용
console.log(user?.id); // undefined
```

https://ko.javascript.info/optional-chaining

구조분해할당 = 비구조화할당 = destructuring

Object나 Array를 변수로 분해할 수 있는 문법

```
const [a, b] = ["넥스트", "멋사"];

console.log(a); // 넥스트
console.log(b); // 멋사

const { id, name } = { id: 1, name: "춘식" };
console.log(id); // 1
console.log(name); // 춘식
```

https://ko.javascript.info/destructuring-assignment

Spread와 rest

1. Spread

   - 객체나 배열을 펼칠 수 있음

2. Rest

   - 구조분해할당과 함께 사용됨
   - 객체, 배열, 함수의 파라미터에서 사용 가능
   - 마지막에 남아있는 인수를 모으는 역할

https://ko.javascript.info/rest-parameters-spread

Function

1. 함수 선언
2. 함수 표현식
   - 익명 함수, 유명 함수
   - 즉시 실행 함수
3. 화살표 함수
4. 콜백 함수
   - 특정 함수의 인자로 들어가서 나중에 호출되는 함수

```
// 함수 선언
function printName(name) {
  console.log(name);
}

// 함수 표현식 - 익명함수
const printName = function (name) {
  console.log(name);
};

// 함수 표현식 - 유명함수
const printName = function print(name) {
  console.log(name);
};

// 함수 표현식 - 즉시 실행 함수
(function (name) {
  console.log(name);
})();

// 화살표 함수
const printName = (name) => console.log(name);

// 콜백 함수
document.getElementById("foo").addEventListener("click", () =>
alert("bar"));
```

브라우저 환경의 자바스크립트

NEXT X LIKELION

DOM

Document Object Model

문서 객체 모델(The Document Object Model, 이하 DOM) 은 HTML, XML 문서의 프로그래밍 interface 이다.

DOM은 문서의 구조화된 표현(structured representation)을 제공하며 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공하여 그들이 문서 구조, 스타일, 내용 등을 변경할 수 있게 돕는다.

DOM 은 nodes와 objects로 문서를 표현한다.

이들은 웹 페이지를 스크립트 또는 프로그래밍 언어들에서 사용될 수 있게 연결시켜주는 역할을 담당한다.

브라우저 렌더링 순서

- html을 다운로드 → dom tree
- link태그를 만나면 css 다운로드 → cssom tree
- script 만나면 해석이 끝날 때까지 렌더링 스탑
- img와 같은 외부리소스 파싱
- dom과 cssom합쳐서 render tree 생성
- layout(reflow): 배치, 배치가 바뀌지 않으면 돔이 바뀌어도 생략
- paint(repaint): 그리기, 배치가 바뀌어도 필수
- render

<script>태그

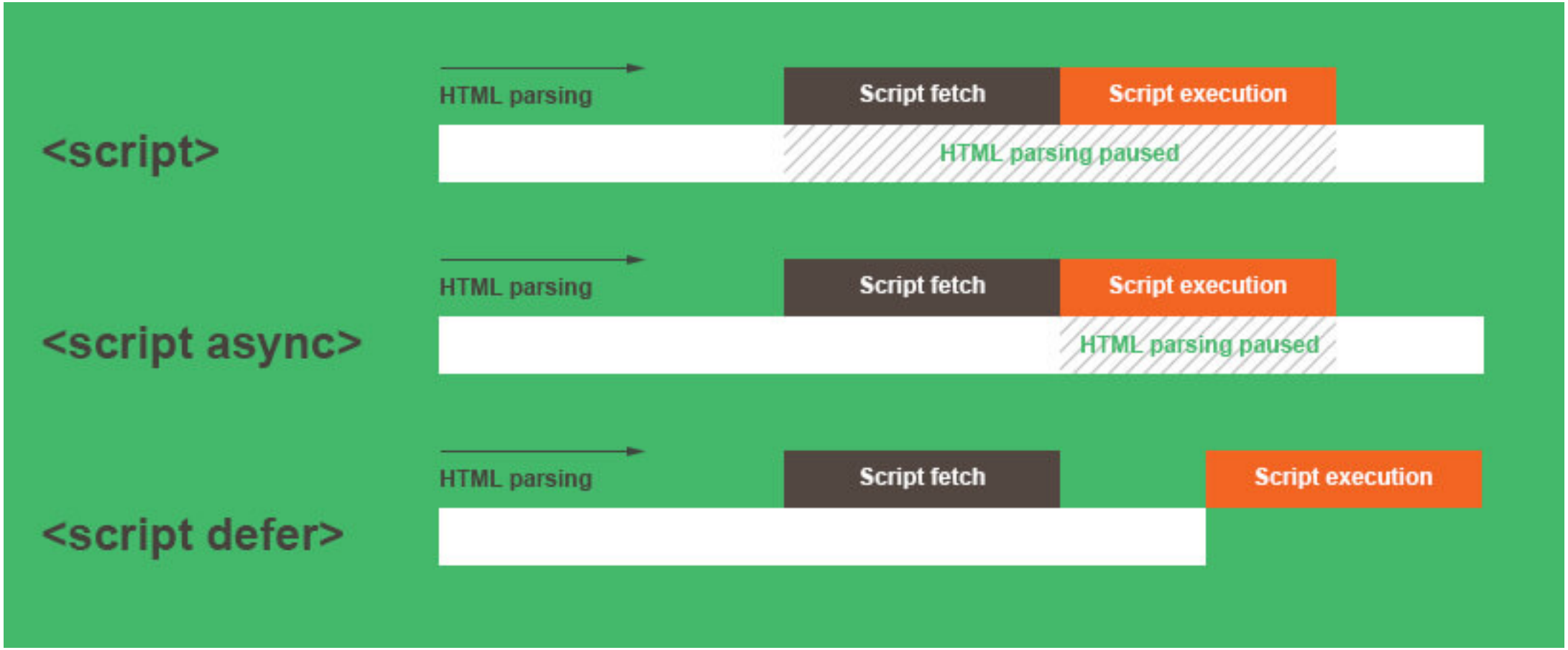
- Html 파싱 중에 script를 만나면 script를 다운받고 실행하는 동안 파싱 중단됨
- Head 태그 안에 또는 body 태그 중간에 일반 script 태그가 들어가면 좋지 않은 사용자 경험 & 아직 존재하지 않은 dom 접근 에러 등의 예상치 못한 문제 발생 가능
- 보통은 body 태그 하단에 들어감

<script>태그

- **async**
  - 다운로드 중에 HTML 파싱을 막지 않지만 다운로드가 완료되면 즉시 실행하고 실행하는 동안 브라우저는 HTML 파싱을 멈춤
  - 독립적인 스크립트에 혹은 실행 순서가 중요하지 않은 경우에 적용
- **defer**
  - 다운로드 중에 html 파싱을 막지 않고, </html>을 만났을 때 실행
  - 외부 스크립트에서만 유효 -> src 없으면 무시
  - DOM 전체가 필요한 스크립트나 실행 순서가 중요한 경우에 적용

https://ko.javascript.info/script-async-defer

<script>태그


<script>
HTML parsing
Script fetch
Script execution
HTML parsing paused
<script async>
HTML parsing
Script fetch
Script execution
HTML parsing paused
<script defer>
HTML parsing
Script fetch
Script execution

이벤트 버블링

- 한 요소에 이벤트가 발생하면, 이 요소에 할당된 핸들러가 동작하고, 이어서 부모 요소의 핸들러가 동작합니다.
- 가장 최상단의 조상 요소를 만날 때까지 이 과정이 반복되면서 요소 각각에 할당된 핸들러가 동작합니다.
- 중단하기: event.stopPropagation()

https://ko.javascript.info/bubbling-and-capturing

이벤트 위임

- 비슷한 방식으로 여러 요소를 다뤄야 할 때 사용됩니다.
- 이벤트 위임을 사용하면 요소마다 핸들러를 할당하지 않고, 요소의 공통 조상에 이벤트 핸들러를 단 하나만 할당해도 여러 요소를 한꺼번에 다룰 수 있습니다.
- 알고리즘
  - 컨테이너에 하나의 핸들러를 할당합니다.
  - 핸들러의 event.target을 사용해 이벤트가 발생한 요소가 어디인지 알아냅니다.
  - 원하는 요소에서 이벤트가 발생했다고 확인되면 이벤트를 핸들링합니다.
- event.target.dataset에 대해 찾아보자

https://ko.javascript.info/event-delegation

여름 방학을 알차게 보내고 싶다면

NEXT X LIKELION

자바스크립트

기본 문법

1. MDN web docs
   https://developer.mozilla.org/ko/docs/Web/JavaScript

2. 모던 자바스크립트 튜토리얼
   https://ko.javascript.info/

3. 드림코딩 자바스크립트 무료 강의
   https://youtube.com/playlist?list=PLv2d7VI9OotTVOL4QmPfvJWPJvkmv6h-2

3. 제로초 렛츠기릿 자바스크립트 무료 강의
   https://inf.run/DQBu

응용 심화

1. 블랙커피 Vanilla JS 카페메뉴 앱 만들기
   https://www.udemy.com/share/105zH43@ujQu89HOpLRQo6w815i1q98EAE1JjM4oRziCON4ie2Gfk17Zxgfh-nj1cEFGGv08SA==/

2. 클린코드 자바스크립트
   https://www.udemy.com/share/105zfE3@e0cmGItu8WxX-TnNpAxTaEi19eHLm9X-C7H9LlcpTbyITBs64QAIpQNL52Ad6d8-EQ==/

React JS

SPA(Single Page Application)을 만들 수 있는 자바스크립트 라이브러리

1. 리액트 공식문서
   https://ko.reactjs.org/

2. 벨로퍼트와 함께하는 모던 리액트(유료 강의도 있음)
   https://react.vlpt.us/

3. 제로초 웹게임을 만들며 배우는 React(무료)
   https://inf.run/Jnzp

4. 코딩애플 리액트(유료) - 아주 쉽고 재밌게 맛보고 싶은 사람 추천
   https://codingapple.com/course/react-basic/

NEXT X LIKELION 양효령