강의에서 사용되는 기초 JavaScript

JavaScript란?

웹 개발의 필수 프로그래밍 언어

동적이고 인터랙티브한 웹 페이지를 만드는 데 사용

특징

- 브라우저에서 실행 (Client-Side)
- Node.js로 서버에서도 실행 가능
- HTML, CSS와 함께 웹의 3대 요소

```
console.log("Hello, JavaScript!");
```

https://www.programiz.com/javascript/online-compiler/

JavaScript 변수와 데이터 타입

키워드	특징	예제
var	함수 범위, 중복 선언 가능	var x = 10;
let	블록 범위, 중복 선언 불가	let y = 20;
const	블록 범위, 값 재할당 불가	const z = 30;

타입	설명	예제
Number	숫자 (정수, 실수 포함)	let age = 25;
String	문자열	let name = "John";
Boolean	참(true), 거짓(false)	let isAdult = true;
Null	빈 값 또는 "아무것도 없음"	let data = null;
Undefined	값이 할당되지 않은 상태	let x;

타입	설명	예제
Object	키-값 쌍의 집합	let person = { name: "John", age: 25 };
Array	값의 순서가 있는 목록	let colors = ["red", "blue", "green"];
Function	코드의 재사용 가능 블록	function greet() { console.log("Hi!"); }

```
let userName = "Alice"; // String
const userAge = 25; // Number
let isMember = true; // Boolean
let favoriteColor = null; // Null
let hobby; // Undefined

// 객체 타입
let user = { name: "Alice", age: 25 }; // Object
let numbers = [1, 2, 3, 4]; // Array
function greet() {
return "Welcome to the site!";
}

// Alert로 값 출력
alert("User Name: " + userName); // String
alert("User Age: " + userAge); // Number
alert("Is a Member: " + isMember); // Boolean
alert("Favorite Color: " + favoriteColor); // Null
alert("Hobby: " + hobby); // Undefined

// 객체 및 배열 출력
alert("User Info: " + JSON.stringify(user)); // Object 출력
alert("Numbers: " + numbers.join(", ")); // Array 출력

// 함수 호출 후 결과 출력
alert(greet()); // Function 실행 결과 출력
```

https://www.programiz.com/javascript/online-compiler/

JavaScript 조건문

조건문의 종류

if-else

- 특정 조건이 참(true)일 때 코드 실행

switch

- 여러 조건을 처리할 때 유용

```
let score = 85;

if (score >= 90) {
console.log("A 학점");
} else if (score >= 80) {
console.log("B 학점");
} else {
console.log("C 학점");
}
```

https://www.programiz.com/javascript/online-compiler/

JavaScript map

map 함수와 객체 배열:

- 배열의 각 객체를 변환하여 새로운 배열 반환
- 원본 배열은 변경되지 않음

```
const people = [
{ name: "John", age: 25 },
{ name: "Jane", age: 30 },
{ name: "Tom", age: 35 }
];
const names = people.map(person => person.name);
console.log(names); // ["John", "Jane", "Tom"]
```

https://www.programiz.com/javascript/online-compiler/

JavaScript 함수

함수란?

- 특정 작업을 수행하는 코드 블록
- 재사용 가능, 코드 구조화에 도움

함수 선언 방법:

- 함수 선언식
- 함수 표현식
- 화살표 함수

```
// 함수 선언식
function greet(name) {
return `Hello, ${name}!`;
}
// 함수 표현식
const add = function(a, b) {
return a + b;
};
// 화살표 함수
const square = x => x * x;
```

https://www.programiz.com/javascript/online-compiler/

JavaScript 비동기(1)

비동기 처리란?

예를 들어.. 피자 배달

- 피자를 주문한 후 배달을 기다리는 동안 고객은 다른 일을 할 수 있음. (ex TV 보기)
- JavaScript의 **비동기 처리**는 이런 상황과 비슷하다 볼 수 있음.
  - 어떤 작업(ex 서버 요청)을 실행한 후, 그 작업이 끝나길 기다리지 않고 다른 작업을 수행함.

Promise란?

예를 들어.. 피자 주문의 배달 수행

- **Promise**는 피자 가게가 배달을 "약속"하는 것과 같음.
  - "30분 내로 피자가 도착합니다!" (약속 상태: pending)
  - 피자가 제시간에 도착하면 약속이 완료됩니다. (약속 상태: fulfilled)
  - 배달이 지연되거나 취소되면 약속이 깨집니다. (약속 상태: rejected)

```
// 피자 배달 Promise
const orderPizza = new Promise((resolve, reject) => {
    console.log("피자 주문 완료. 배달 중입니다...");
    setTimeout(() => {
      const success = true; // 배달 성공 여부
      if (success) {
        resolve("피자가 도착했습니다! 🍕");
      } else {
        reject("배달 실패! 😢");
      }
    }, 3000); // 3초 후 결과
  });

  // Promise 사용
  orderPizza
    .then((message) => console.log(message)) // 성공 시 출력
    .catch((error) => console.error(error)) // 실패 시 출력
    .finally(() => console.log("피자 주문 프로세스 완료."));
```

https://www.programiz.com/javascript/online-compiler/

JavaScript 비동기(2)

async/await란?

•비유: "주문이 끝날 때까지 기다리기"

- Promise를 사용하면 약속의 결과를 then으로 처리. 하지만 async/await는 "주문이 완료될 때까지 기다린 후" 결과를 처리하는 방식을 제공.
- async/await는 Promise를 더 간단하고 읽기 쉽게 다룰 수 있음.

Promise와 async/await 비유

1.Promise:

1. 피자를 주문한 후, 알림(전화 또는 메시지)을 받으면 작업을 처리함.
2. 여러 알림을 처리하려면 알림 체인을 관리해야 함. (then 체인)

2.async/await:

1. 피자를 주문한 후 배달원을 기다리며 바로 결과를 확인함. 다른 작업과 병렬적으로 수행할 필요가 없을 때 적합.

```
// 비동기 함수 선언
async function orderPizzaAsync() {
    try {
      console.log("피자 주문 완료. 배달 중입니다...");
      const result = await new Promise((resolve, reject) => {
        setTimeout(() => {
          const success = true;
          if (success) {
            resolve("피자가 도착했습니다! 🍕");
          } else {
            reject("배달 실패! 😢");
          }
        }, 3000);
      });
      console.log(result); // 성공 메시지 출력
    } catch (error) {
      console.error(error); // 실패 메시지 출력
    } finally {
      console.log("피자 주문 프로세스 완료.");
    }
  }
 orderPizzaAsync();
```

https://www.programiz.com/javascript/online-compiler/