**JAVAScript day1**

코딩은 컴퓨터와 대한하는 방법이다.

자바스크립트는 컴퓨터에게 일을 시키기 위한 언어이다.

그중에서 console.log()는 “이 값을 화면에 보여줘” 라고 컴퓨터에게 알려주는 가장 기본적인 방법이다.

**콘솔(console)이란?**

콘솔은 컴퓨터와 대화를 나누는 창이다. 우리가 코드를 입력하면 그 코드에 따라 어떤 동작을 하거나 결과를 보여준다. 이처럼 코드를 직접 입력하고 실행 결과를 바로 확인할 수 있는 창이 바로 콘솔이다.

개발자는 콘솔을 이용해 코드가 제대로 작동하는지 확인하고, 중간 결과를 출력하거나, 오류를 추적한다. 즉, 콘솔은 **코드를 실험해보고, 컴퓨터의 반응을 직접 볼 수 있는 소통 도구**이다.

이 콘솔은 다양한 환경에서 사용할 수 있지만, 우리가 가장 빠르게 접할 수 있는 콘솔은 **웹브라우저의 개발자 도구에 있는 콘솔 탭**이다. 개발자 도구의 콘솔탭은 코드의 실행 결과나 에러를 간편하게 확인할 수 있는 곳이다.

**console.log()**

console.log() 는 괄호 안에 있는 코드를 평가해 콘솔에 출력해 달라는 명령어이다. JAVAScript의 모든 데이터와 표현식은 console.log()를 통해 콘솔에 출력할 수 있다.

**텍스트, 소프트웨어, 멀티미디어 소프트웨어, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

console.log(1) // 출력값: 1

console.log(2) // 출력값: 2

console.log(3) // 출력값: 3

console.log()는 자동개행이다. 즉, 한 줄씩 출려되며 한 줄에 여러값을 출력하고 싶다면 값과 값 사이에 ,를 넣어주면 된다.

텍스트, 스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.괄호 안에 하나의 값을 전달할 수도 있지만 1+1과 같이 표현식을 입력하여 결과를 출력할 수 있다.

Console.log(1+1) // 출력값 2

console.log(10+5) // 출력값 15

console.log(10-5) // 출력값 5

또한 아래와 같이 따옴표로 묶인 텍스트(string)도 출력할 수 있다.

console.log(“Hello, world!”); // 출력값 : Hello, World!

console.log(“JaVaScript는 재밌어!”) // 출력값 JaVaScript는 재밌어!

**console.log()는 어떤 상황에 사용하나요?**

console.log()는 주로 아래와 같은 상황에 사용된다.

* **코드가 제대로 작동하는지 확인할 때**

함수의 실행 결과나 변수에 어떤 값이 들어 있는지 확인할 때

* **에러가 발생한 위치나 원인을 찾을 때**

특정 지점까지 코드가 실행되는지 확인하거나, 값이 예상과 다르게 나오는 부분을 추적할 때 유용하다.

* **디버깅할 때**

복잡한 로직 안에서 어떤 조건이 충족되었는지, 루프가 몇 번 도는지를 확인하는 데에 활용된다.

* **사용자 입력값이나 외부 데이터 확인할 때**

사용자가 입력한 값이나 서버에서 받아온 데이터가 정상적으로 들어왔는지를 확인할 수 있다.

즉 console.log()는 개발 중 **코드의 흐름과 상태를 눈으로 확인하는 가장 간단하고 강력한 도구**이다.

**변수란?**

컴퓨터와 함께 작업하려면 **데이터를 저장하고 꺼내 쓰는 방법**이 필요하다.

변수가 바로 그 역할을 하며 **변수는 값을 기억해 두기 위한 이름표**이며 **값을 저장해 놓고 필요할 때 그 이름으로 꺼내쓸 수 있는 도구**이다.

왜 변수가 필요한가?

대부분의 JaVaScript 프로그램은 사용자나 서버로부터 받은 정보를 처리하는 방식으로 동작한다. 예를 들어, 계산기 앱은 사용자가 입력한 숫자와 계산 결과를 저장하고, 온라인 쇼핑몰은 상품 목록이나 장바구니 정보를 기억해야 한다. 이런 다양한 데이터를 저장하고 관리하기 위해 변수가 필요하다.

JaVaScript에서는 이렇게 변하는 **데이터를 담아두는 공간**으로 변수를 만들고 거기에 이름을 붙인다.

**변수 선언**

변수를 사용하려면, 먼저 변수를 만들어야 하는데 이를 변수 선언이라 한다.

let 이라는 키워드를 사용해 선언한다.

let userName; // userName 이라는 이름을 가진 변수를 선언

이렇게 하면 userName이라는 이름표만 만들어진 상태이며 아직 값은 정해지지 않았다.

**변수에 값 할당하기**

변수를 선언한 뒤에는 그 변수에 값을 저장하는 작업이 필요하다. 이때 할당 연산자 =를 사용하며 이 과정을 값을 할당한다고 표현한다.

userName = “OZ”; // 변수 userName에 문자열 “OZ”를 할당

즉 userName 이라는 이름표가 붙은 공간에 “OZ”라는 값을 넣는다 라는 의미이다. 변수 선언과 값을 할당하는 과정을 한 줄로 작성 할수도 있다.

let userName = “OZ”; // 선언과 할당을 한번에

이렇게 하면 변수 선언과 동시에 값을 저장하는 셈이다.

let userId = 1; //변수 userId를 선언하고 숫자 1을 할당하기

let userName = “OZ”; // 변수 userName을 선언하고 문자열 “OZ”를 할당

**console.log()로 변수 출력하기**

변수에 할당된 값은 변수명으로 접근하여 사용할 수 있다.

let greeting = “안녕하세요!”;

console.log(greeting). // “안녕하세요!”

먼저 greeting 변수를 선언하고 “안녕하세요!”라는 값을 할당한다. 그다음 console.log 명령문에 greeting 변수를 넣어 출력하면 콘솔창에 “안녕하세요!”가 표시된다.

**변수 재할당**

let 키워드로 선언한 변수는 값을 재할당 할 수 있다. 재할당이란 변수에 할당된 기존 값을 새로운 값으로 바꿔 넣는 것을 의미한다.

let count =1;

count = 2;

console.log(count); // 2

위 코드에서 count 변수에 처음엔 1이 저장되었지만 그 다음줄에서 2로 값을 바꾸었기에 콘솔에 2가 출력된다.

반면 다른 변수 선언 키워드인 **const로 선언한 변수는 값을 한번만 할당할 수 있고 이후에는 재할당이 불가능하다**.

이처럼 **변하지 않는 값을 저장하는 변수를 상수(constant)**라고 한다.

const number = 10;

number =20; // uncaught TypeError

**변수명 규칙**

변수를 선언할 때 다음과 같은 이름 규칙을 준수해야 한다.

* 특수문자를 제외한 문자,숫자 언더스코어(\_), 달러기호($)를 포함할 수 있다.
* 변수 이름은 숫자로 시작할 수 없다.
* 변수 이름으로 키워드를 사용할 수 없다.

키워드란?

JaVaScript 프로그래밍 언어에서 어떤 역할이나 기능이 정해진 특별한 단어.

키워드는 변수 선언 키워드인 var, let, const, asyne,await,catch, class, for, null, this, try, switch 등 많다.

**네이밍 컨벤션**

네이밍 컨벤션이란 코드의 가독성과 유지 보수성을 높이기 위한 코드 작성 규칙.

협업 과정에서 다른 개발자가 코드를 이해하기 쉽게 만들고, 프로젝트의 전반적인 품질을 향상시키기 위해 일관된 네이밍 규칙을 따라야 한다.

1. **카멜케이스(camelCase)**

JaVaScript에서 변수, 객체의 속성, 함수는 모두 카멜케이스(camelCase)를 사용한다. 첫 번째 단어는 소문자로 시작하고, 이후의 각 단어를 대문자로 시작한 형태가 낙타의 등과 비슷하다고 하여 카멜케이스라고 한다.

ex). myVariable, calculateTotalWidth

1. **파스칼 케이스(PascalCase)**

모든 단어의 시작을 대문자로 시작하는 파스칼 케이스는 클래스와 생성자를 사용할 때만 제한적으로 사용되는 네이밍 컨벤션이다.

ex) Product, UserProfile

**JAVAScript day2**

**number 타임 값을 사용하여 간단한 연산의 예시를 들 수 있다.(+.-,\*,/,%)**

**산술 연산자**

덧셈,곱셈,뺄셈,나눗셈과 같은 연산이 존재하는데 이러한 연산을 도와주는 연산자를 산술 연산자라고 한다.

덧셈 연산자 +

뺄셈 연산자 –

곱셈 연산자\*

나눗셈 연산자 /

나머지 연산자 %

**덧셈 연산자**

let num1 = 15;

let num2 = 5;

console.log(num1 + num2) // 출력 결과 20

덧셈 연산자 + 는 숫자끼리는 덧셈, 문자열과 함께 쓰이면 연결(concatenation)을 수행한다.

ex)

console.log(10 + ‘5’) // 출력 결과 105

10과 5가 문자열로 결합되어 105가 출력됨

**뺄셈 연산자**

뺄셈 연산자 – 는 첫번째 숫자에서 두 번째 숫자를 뺀다.

let num1 = 15;

let num2 = 5;

console.log(num1 – num2) // 출력 결과 10

**나눗셈 연산자**

나눗셈 연산자 / 는 첫 번째 숫자를 두 번째 숫자로 나눈다.

let num1 = 15;

let num2 = 5;

console.log(num1 / num2) // 출력 결과 3

**나머지 연산자**

나머지 연산자(%)는 첫 번째 숫자를 두 번째 숫자로 나눈 나머지를 구한다.

let num1 = 15;

let num2 = 4;

console.log(num1 % num2). // 출력 결과 3

**산술 연산자 요약표**

텍스트, 번호, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**복합 대입 연산자**

**산술 연산과 변수 대입을 동시에 할 수 있는 축약 표현.**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

let x = 10;

x += 5; // 15

x \*= 2; // 30

console.log(x); // 출력 결과 30

**증감 연산자**

변수의 값을 **1씩 증가하거나** 감소시키고 싶을 때는 증감 연산자를 사용할 수 있다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

let count = 5;

count++; // 1증가

console.log(count); // 6

count--; // 1감소

console.log(count). // 5

**++x VS x++의 차이**

두 표현 모두 값을 1증가시키지만, **값을 반환하는 시점**이 다르다.

* ++x : 값을 먼저 1 증가시키고 그 결과를 반환한다.
* x++ : 값을 먼저 반환한 뒤, 나중에 1 증가시킨다.

let x = 5;

console.log(++x); // 6 (x를 먼저 증가 -> 그 값을 출력)

console.log(x++); // 6

x=5;

console.log(x++); // 5 (x를 먼저 출력 -> 그다음 증가)

console.log(x); // 6

이 차이는 **값을 바로 사용하는 상황**에서만 중요하며, **단순히 값을 증가시킬 때는 둘 다 같은 결과**는 낸다.

**연산자 우선순위 정리**

console.log(50+ 10 / 5+2); // 50 + 2 + 2 -> 출력 결과 54

console.log((50+10) / (8+2); // 60 / 10 -> 출력 결과 6

* 연산자끼리 우선순위가 다르면 우선순위 높은 것 먼저 계산
* 우선순위가 같으면 결합 방향에 따라 계산 순서가 결정됨
* 명확한 계산 순서를 원하면 괄호()를 사용한다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

ex)

console.log(3+4\*2); // 3+ (4\*2) -> 11

console.log((3+4) \* 2. // (3+4) \* 2. -> 14

let a = 2;

a \*\*= 3; -> a= a \*\* 3 -> 8

**실전 팁**

헷갈리면 괄호로 묶기 : 우선순위보다 가독성이 중요할 때가 많다.

연산자 우선순위는 자주 쓰다 보면 익숙해지지만, 처음엔 의식적으로 괄호를 써서 흐름을 명확히 해라.

**Math 객체의 다양한 메서드(Math,random(),math.floor(),math.ceil(),math.pow()등)를 활용하여 난수 생성, 올림/내림, 거듭제곱 계산과 같은 수학적 연산의 사용 예시를 들 수 있다.**

**math 객체는 숫자 계산을 도와주는 javascript의 내장 객체이다.**

내장 객체란 javascript 안에 이미 만들어져 있어서 우리가 따로 만들지 않아도 바로 사용할 수 있는 도구이다.

Math 객체 안에는 무작위 숫자를 만들거나, 숫자를 올림/내림하거나, 제곱을 계산하는 등 숫자와 관련된 다양한 기능들이 들어있으며 이런 기능을 메서드라 부른다.

**math.random(), math.floor(), math.pow()** 처럼 (.)을 찍어 사용하는 방식으로 호출한다.

이처럼 javascript는 math 뿐만 아니라 유용한 객체들을 기본으로 제공하며, 우리는 그것들을 불러 쓰기만 하면 된다.

**Math 객체**

**Math 객체는 숫자 계산**을 위한 기능들이 미리 정리된 **Javascipt의 기본 도구**이다.

javascript에는 우리가 직접 만들지 않아도 사용할 수 있는 내장 객체들이 있다. 이 객체들은 프로그램을 만들 때 자주 쓰이는 기능을 담고 있어, 별도의 설정 없이 언제든 사용할 수 있다.

**Math 객체**는 그 중 하나로, 숫자와 관련된 다양한 기능들을 포함하고 있다. 복잡한 수학 계산도 Math 객체의 **메서드**를 활용하면 아주 간단하게 처리할 수 있다.

**객체** : javascript에서 객체는 여러 값을 하나로 묶어 놓은 덩어리라고 볼 수 있다. 객체에는 다양한 정보와 기능이 담길 수 있는데 그 구성요소를 속성(또는 프로퍼티)라고 부른다. 각 속성은 **이름(키)**과 **값(value)**으로 이루어져 있다. 객체는 다양한 기능이나 데이터를 묶어놓은 구조이다.

**Math 메서드**

**메서드**는 객체가 가지고 있는 기능(함수)이다. Math 객체의 메서드는 모두 Math.메서드이름() 형태로 사용하며, 숫자를 계산해주는 역할을 한다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

난수 : 예측할 수 없는 무작위 숫자

Math.random()은 0보다 크거나 같고, 1보다 작은 소수값을 반환한다.

ex) 0.238192, 0.91472 처럼 매번 실행할 때마다 다른값이 나온다.

**Math.random()**

**Math.random()**은 0이상 1 미만의 숫자를 무작위로 반환한다. 실행할 때마다 매번 **다른 소수값**이 출력된다.

console.log(Math..random()) // 0.5668689883160865

console.log(Math.random()) // 0.8819052071000923

console.log(Math.random()). // 0.0781936370960119

Math.random() 이 반환하는 값은 항상 0이상 1미만이기 때문에 범위를 바꾸고 싶다면 계산이 필요하다. 예를 들어 0이상 10미만의 숫자를 만들고 싶다면 아래처럼 10을 곱하면 된다.

console.log(Math.random() \* 10) // 7.522055818079993

**Math floor(x)**

**Math.floor(x)**는 x보다 작거나 같은 수 중에서 가장 큰 정수를 반환한다.

수학에서 내림과 같은 방식이다.

console.log(Math.floor(5.95)); // 5

console.log(Math.floor(45.05)); // 45

console.log(math.floor(-1.6)); // -2

앞서 사용한 Math.random()의 결과를 정수로 출력하고 싶다면 Math.floor()와 함께 사용하면 된다.

console.log(Math.floor(Math.random() \* 10)); // 2

이렇게 하면 0이상 10 미만의 정수를 랜덤으로 출력하게 된다.

Math.random()으로 생성된 실수에 10을 곱한 후, Math.floor()로 내림처리하여 정수를 얻을 수 있다.

**주사위 만들기**

Math.random()과 Math.floor()를 이용하여 1부터 6까지의 정수를 무작위로 출력하는 주사위 함수를 만들어보자

// 1부터 6까지의 정수를 무작위로 반환하는 주사위 함수

function rollDice(){

return Math.floor(Math.random() \* 6) + 1;

}

console.log(rollDice());

console.log(rollDice());

console.log(rollDice());

* Math.random() -> 0이상 1미만의 소수값 생성
* Math.random() \* 6 -> 0이상 6미만의 소수값으로 범위 확장
* Math.floor(…) -> 소수점 아래 버림(0,1,2,3,4,5 중 하나)
* +1 -> 1부터 6까지 정수값 완성

이 구조는 랜덤 정수 범위 만들기에 자주 사용되며 Math.floor(Math.random()\* (최댓값 – 최솟값 + 1)) + 최솟값 형태로 일반화할 수 있다.

**Math.ceil(x)**

Math.ceil(x)은 x보다 크거나 같은 수 중에서 가장 작은 정수를 반환한다.

수학에서 올림과 같은 방식

condole.log(Math.ceil(5.95)); // 6

console.log(Math.ceil(45.05)); // 46

console.log(Math.ceil(-1.6)); // -1

**Math.pow(x,y)**

Math.pow(x,y)는 x의 y 제곱을 반환한다.

console.log(Math.pow(2,3)); // 8

console.log(Math.pow(2, -1)); // 0.5

**Math.abs(x)**

Math.abe(x)는 x의 절대값을 반환한다. x가 음수이면 -x를 반환하고 양수라면 x를 반환한다. 따라서 결과는 항상 양수 혹은 0이다.

console.log(Math.abs(-1)); // 1

console.log(Math.abe(3)); // 3

**이외에 다양한 Math 객체**

숫자 계산에서 Math 객체는 매우 자주 사용되며, 위에 나온 메서드들은 실무에서도 자주 활용된다. 필요한 계산이 있다면 Math에서 찾을 수 있을 가능성이 높다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.