Typescript#2 함수

**서론 :**

어느 덧 타입스크립트의 마지막 수업입니다. 오늘은 타입스크립트의 함수 정의에 대해서 알아보도록 하겠습니다. 클래스와 더불어서 함수에 매개변수를 정의하고 return 하는 값을 지정하는 것은 개발자에게 매우 중요한 일입니다.

**함수 작성법 :**

자바스크립트에는 두가지 함수 타입이 존재합니다. 바로 기명 함수와 익명 함수입니다. 당연히 둘의 차이는 함수의 이름이 존재 유무입니다. 물론 다들 아시겠지만, 익명 함수와 기명 함수의 예시를 보며 빠르게 진행 하겠습니다.

//기명함수

function add(x, y){

return x + y

}

//무기명 함수

var add = function(x,y){

return x + y

}

두 함수 모두 함수 선언식으로 작성하였지만, 무기명 함수의 경우 함수의 이름이 존재하지 않아, 그 자체로 사용이 불가능 하기에 add변수에 무기명 함수를 할당하여 사용하였습니다.

**함수 매개변수 타입 작성 :**

우리는 첫 시간에 매개변수에 타입을 주는 방법에 대하여, 더 알아보았지만 다시 한번 알아보며, 조금 더 나아가 유니언 타입의 매개변수를 사용 하는 방법에 대해서 알아보도록 하겠습니다.

./function.ts

function add(x : number, y : number) : number{

return x + y;

}

함수의 매개변수에 타입을 지정하는 방법은 다음과 같이 param : type , param : type 으로 작성하면 컴파일 전 전달인자의 타입을 체크 합니다.

함수가 return하는 값의 타입을 지정 할 때는 아시는 바와 같이 괄호 연산자 오른쪽에 : type을 작성하여 해당 함수의 return 값을 명시 할 수 있습니다.

매개변수의 타입을 지정할 때 다형성을 고려 한다면, any type을 사용 해야 할 지 모릅니다. Moment.add(time, format)이나 가상의 함수 drive(vehicle)같은 함수들이 해당합니다. 이런 경우 any를 사용하면 속은 편하겠지만, typescript를 사용하는 이유가 없어집니다. 다형성을 고려하여, 어떤 매개변수의 타입을 여럿 지정 할 수 있다면 어떨까요 ? 이를 위해 유니언이라는 것을 타입스크립트에서 지원합니다.

우리가 만든 add 함수에 x 혹은 y매개 변수에 type을 string까지 추가 하도록 하겠습니다. 예시도 같이 만들 도록 하겠습니다.

Add-button이라는 id를 가진 버튼을 클릭시, input-1과 input-2의 value를 가져와 이를 합쳐 alert함수로 전달하는 이벤트를 하나 작성하도록 하겠습니다.

Document객체를 사용 하려면 먼저 tsconfig파일을 열어 stricNullChecks 항목을 false로 변경해줍니다.

그리고는 addEventListener를 통하여, “input-1”과 “input-2” 에서 value를 가져와 add함수에 전달합니다.

document.getElementById("add-button").addEventListener("click",()=>{

const a = (<HTMLInputElement>document.getElementById("input-1")).value;

const b = (<HTMLInputElement>document.getElementById("input-2")).value;

alert(add(a,b))

})

function add(x : number | string, y : number | string) : number{

return Number(x) + Number(y);

}

만약 add함수가 number타입의 매개변수만 받는다면, 어떨 까요? Input으로 받은 value값을 전달하기 전에 Number타입으로 변환하여야 합니다. 만약 add함수를 자주 사용하게 된다면, input의 값을 Number로 작성하지 않아 실수도 자주 발생될 수 있고, 수 많은 string 값을 Number타입으로 변경하여, 변수에 할당해 코드가 지저분해 질 수 있습니다. 이런 union 타입은 함수를 좀 더 유연하고 간결하게 적용 할 수 있는 힘이 있습니다 .

**선택적 매개변수 :**

만약 지정된 매개변수에 인자를 전달하지 않으면 어떻게 될까요 ? undefined 가 해당 매개변수에 할당되게 됩니다. 함수는 약 3개 이상의 매개변수를 가지게 된다면, 매우 복잡해집니다. 그래서 보통 2개의 매개변수를 가지는게 일반적이며, 다양한 매개변수를 전달하는 경우, 객체를 전달하는 것이 이상적입니다. 물론 add와 같은 함수는 전달 인자의 길이가 길어 arguments를 이용할 수 도 있습니다.

Base number와 exponent(지수)를 전달 받는 함수를 작성 하도록 하겠습니다.

./function.ts

function power(base : number, exponent : number) : number{

let index = 1;

function \_\_power(\_\_base : number, \_\_exponent : number) : number{

if(\_\_exponent === index){

return \_\_base;

}else{

index++;

return \_\_power(base \* \_\_base,\_\_exponent)

}

}

return exponent ? \_\_power(base,exponent) : 0;

}

클로저를 이용한, power 재귀 함수를 만들어 봤습니다. 이를 제대로 사용하는 방법은 power(10,2)와 같이 전달 인자를 제대로 넣는 것이지요. 그렇기에 ts에서는 매개변수 수에 맞추어, 전달인자의 수를 맞추지 않고 호출 하는 경우 다음과 에러를 발생시킵니다.

power(10) //오류 : 전달인자가 부족하다.

power(10,20,30)//오류 : 전달인자가 많다.

power(10,20); //정상

보통 option들을 매개변수로 할당하는 생성자 함수나, 메서드들은 매개변수를 null check 하여, 해당 매개변수의 옵션에 따른 default value를 할당 합니다. 우리의 경우는 power 함수는 exponent매개변수가 만약 옵션으로 전달 하지 않은 경우, 이를 0으로 처리하여 2^0으로 처리하여, 0을 return하는 방법이 존재합니다. 만약 exponent를 유니언 타입으로 undefined를 처리하여도 전달인자로 undefined를 직접 전달하지 않는다면 매개변수 부족으로 에러를 발생 시킵니다. 이를 위해, 매개변수를 선택적으로 지정할 수 있는 선택적 매개변수 옵션이 존재합니다.

선택적 매개변수를 지정하는 방법은 매개변수 명 우측에 “?” 를 달아주는 것으로 가능합니다.

function power(base : number, exponent? : number) : number{

…

}

console.log(power(10)) //정상

// power(10,20,30)//오류 : 전달인자가 많다.

console.log(power(10,20)); //정상

이렇게 되면 power함수를 호출 할 때 exponent에 인자를 전달하지 않아도, 컴파일 할 수 있게 허용 합니다. 그러나 위와 같이 단순히 매개변수를 함수 내에서 undefined 처리하여 사용하는 경우, 많은 오류를 범할 수 있는 위험이 존재합니다. 또한 매개변수가 undefined 가 아닌 default value로 할당되어 사용 된다면, 코드내에서 or 연산자를 사용하여 변수에 할당하거나, 조건문을 사용해야 합니다. 이와 같은 경우에는 코드를 지저분하게 만들 수 있습니다. 이를 위해서 ES6에서는 기본 초기화 매개변수라는 방법을 제공하며, 타입스크립트에서도 이를 지원합니다. 선택적 매개변수에 기본 값을 할당하는 방법은 다음과 같습니다.

function power(base : number, exponent = 0) : number{

…

return exponent ? \_\_power(base,exponent) : 0;

}

console.log(power(10)) //정상 : 전달인자가 부족하다.

// power(10,20,30)//오류 : 전달인자가 많다.

console.log(power(10,20)); //정상

만약 exponent 매개변수에 인자를 할당하지 않는다면, exponent 매개변수는 0으로 기본 할당 될 것 입니다. 이는 컴파일 뿐만 아니라 런타임에도 유효한 옵션입니다.

기본 초기화 매개변수는 지정된 값의 타입에 따라 매개변수의 타입이 지정됩니다. 즉.

// exponent = 0 === exponent : number

이와 같이 되어 만약 number타입이 아닌 다른 타입의 값으로 인자를 전달 할 시 타입 오류가 발생합니다.

console.log(power(10,"2")) //오류 : 타입 오류

// power(10,20,30)//오류 : 전달인자가 많다.

console.log(power(10,20)); //정상

**나머지 매개변수 :**

Add 함수를 조금 더 고쳐 보겠습니다. ES5이전에는 이름을 지정해주지 않은 매개변수에 대하여, 대응하기 위해서 arguments 배열을 함수 내에서 사용하였습니다.

function add() {

let result = 0;

for(let i =0; i < arguments.length; i++){

result += arguments[i];

}

return result;

}

console.log(add(10,20,30,40))

ES5에서는 나머지 매개변수를 지원하게 되면서, arguments를 대체 할 수 있게 되었습니다. 나머지 매개변수를 사용하는 방법은 다음과 같습니다.

function add(x : number | string, ...rest : number[]) : number{

}

필수 매개변수나 선택적 매개변수 끝에, …을 붙인 매개변수를 선언 하는 것입니다. Rest매개변수는 이름을 지정한 매개변수를 제외한 나머지 매개변수를 의미 합니다.

이를 이용하여, add함수를 좀 더 유연하게 작성 하도록 하겠습니다.

function add(x : number | string, ...rest : number[]) : number{

let result = Number(x);

for(let i =0; i < rest.length; i++){

result += rest[i];

}

return result;

}

이로써 필수 매개변수를 제외한 매개변수에 유연하게 대처하는 add함수를 만들 수 있게 되었습니다.

**~~This :~~**

~~This는 함수가 호출 되는 시기에 결정되는 변수입니다. 많은 이들이 객체를 사용할 때 this가 가리키는 객체가 혼돈 스러워 많은 고충을 느낍니다.~~

#4 this로 수업을 제작 할 예정