지난 시간에 이은 퀴즈

1. 콜바이 밸류란 무엇인가 //함수에 전달한 변수의 값만 복사하여 파라메터에 할당하는 메커니즘
2. 콜바이 레퍼런스란 무엇인가 //함수에 전달 변수의 메모리 값을 참조하여 파라메터에 할당하는 메커니즘 -> 전달인자의 값을 변환 할 수 있다.
3. 이 둘의 차이는 무엇인가?
4. 익명함수는 무엇인가 ?
5. 콜바이 밸류에 해당하는 5가지 원시 데이터타입은 ?
6. A 변수를 두개의 파라미터를 가k진 sum함수를 통해 값을 변경하라.
7. 전역변수란 ? 지역변수란 ?
8. 배열을 파라미터로 받아 모든 버킷의 값에 10씩 더 해주는 addTen함수를 만들어라

금일할 내용 컨텍스트와 스코프입니다. 이 두 단어는 단순히 자바스크립트에 국한된 용어들이 아니기 때문에 오늘 내용 기억하시다가 나중에 이 두개 때문에 작업에 차질이 생기면 구글링 하셔서 보시길 바라겠습니다.

자바스크립트에서 스코프란 현재 위치해 있는 함수나, 객체, 배열 내부에서 어떤 변수들에 접근 할 수 있는지를 정의합니다. 자바스크립트의 스코프를 분류해보면 , 함수가 정의될 때 참조하는 위치를 결정하는 정적스코프와 동적스코프, 블록 내부와 외부 중 참조 위치를 결정하는 전역스코프와 지역스코프로 나뉩니다.

먼저 전역 스코프란

전역 스코프는 변수가 함수 바깥이나 {} 바깥에서 선언되었다면, 전역 스코프에 해당하게 됩니다. ( 일반적인 전역변수의 내용과 비슷합니다. )

const name = "leejogngho";

function sayMyName(){

console.log(name);

}

sayMyName();

console.log(name)

이런 함수를 만들어 실행하게 되면 쎄마넴 함수는 name변수를 찾아 블록 바깥을 찾게 됩니다. 전역변수, 지역변수를 이해한 여러분들 수준에서는 너무 쉬운 내용입니다.

그러나 전역 스코프를 이용하여 변수를 선언하는 경우 큰 문제점이 있습니다. 프로그래밍에서 아주 안 좋은 평가를 받을 정도로요. 사실 전역 변수가 나쁜 것은 아닙니다. 프로그래밍을 하다보면 반드시 써야합니다. 하지만 나쁜 것은 전역 스코프를 남용하는 것입니다. 예제를 하나 만들어 보도록 하겠습니다.

let name = "Irena";

let age = 25;

function hello(){

console.log(name);

}

function getBirtyYear(){

console.log(new Date().getFullYear() - age);

}

2개의 함수를 만들었는데, 이 두 함수의 가장 큰 문제 점은 함수가 호출하는 스코프에 대단히 의존적이라는 것입니다. 그 말인 즉 슨 저 name,age변수는 어떤 함수든 , 프로그램 어디에서든 상관없이 의도적으로든 실수 던지 바꿀 수 있습니다. 그러므로 지금 저 프로그램은 프로그램의 다른 부분에서 name,과 age를 정확히 사용한다고 가정하고 있는 것입니다.

그렇다면 이를 해결하는 방법이 무엇일까요 ? 네 여러분들이 아는 파라메터를 이용해서 정확한 곳에 정확한 방법으로 호출을 유도하는 방법입니다.

function hello(*name*){

console.log(name);

}

function getBirtyYear(*age*){

console.log(new Date().getFullYear() - age);

}

사실 프로그램이 저렇게나 단순하다면 문제가 생길 소지는 전혀 없습니다. 그러나 프로그램이 수만줄, 수천 줄이되면, 모든 스코프를 기억하고 관리 할 수 없게 된다면 전역 스코프에 의존하지 않는 것이 정말 중요해집니다.

다음으로는 이 전역스코프와 엮어 우리가 var 문법을 쓰지 않는 이유입니다.

이것은 es6문법이 생기기 전 2015년 이전으로 돌아갑니다. Es6문법 전에는 변수를 선언하는 방법이 지금처럼 let 과 const 를 이용한 방식이 아닌 var 하나만 있었습니다. 이 var는 치명적인 문제가 있었는데, var로 변수를 선언하고 두 번째로 같은 변수를 선언하게 된다면 두번 째 변수가 첫 번째 변수를 덮어 쓰게 됩니다.

var cat = "cute";

var cat = "awsome";

이 var 문법은 글로벌 스코프를 가지고 있습니다. 그래서 같은 블럭 내부에서 재 선언 된 경우 기존에 있던 변수의 값을 두 번째 선언한 내용의 값으로 변경됩니다. 더 심각한 문제는 이 문제가 배포 후 발생 할 수 있다는 점 입니다. 만약 몇 만줄이나 되는 경우에 어떤 프로그래머가 var로 선언한 변수를 같은 스코프내(다른 파일)에서 변수를 실수로 재 선언 했습니다. 그리고 간단하게 테스트를 한 경우 별 문제 없이 테스트가 통과 되었습니다. 그러나 이 var로 재 선언된 변수가 고약한 유저들의 손에 변형이 되어 특정한 버그를 일으킬 수도 있습니다. 그리고 웹 특성상 런타임 환경에서는 오류를 찾기 어렵기 때문에 이 전역스코프로 설정된 var 변수하나가 대재앙을 일으킬 수 있습니다.

그러나 es6문법에서 생긴 let과 const는 같은 스코프내에서 변수가 재 선언된 경우 오류메세지를 발생하게 됩니다. 만약 앞서 말한 프로그램의 let과 const로 변수를 설정했다면 같은 오류를 범하지는 않았을 것입니다.

다음으로는 지역 스코프 입니다.

이 지역스코프는 우리가 지역변수를 통해서 알아 봤 듯이 특정 부분에서만 사용이 가능한 변수입니다. 자바스크립트에서는 크게 두 가지의 지역 스코프가 존재 합니다.

첫번째로 함수 스코프입니다.

여러분이 만약 함수 내부에서 변수를 선언한다면, 그 변수는 선언한 함수 내부에서만 사용이 가능합니다.

function helloWolrd(){

let string = "hello world";

console.log(string); //hello world

}

console.log(string) //undefined

이 함수 내부에서 설정한 변수는 외부에서는 작동하지 못하고 , 함수 내부에서만 작동하게 됩니다. 그렇다면 다음과 같은 경우에서는 어떻게 될까요 ?

let s` = "hi";

var dummy = "cool";

function helloWolrd(){

let string = "hello world";

var dummy = "boo";

console.log(string); //hello wolrd

}

helloWolrd();

console.log(dummy) cool

놀랍게도 오류를 일으키지 않습니다. 그 이유는 선언된 string과 dummy는 함수내부에서 선언되었기 때문에 이 두 스코프는 함수 스코프를 가지게 됩니다. 이 말인 즉슨 이 함수에서 선언되었기 때문에 이 두 변수는 이 함수 내부에서만 유효하게 되는 것입니다. 그렇기 때문에 헬로월드 함수 바깥에서 전역변수들이 영향을 받지 않고, 오류를 일으키지 않게 되는 것입니다.

지역스코프의 두번 째 스코프는 바로 블럭 스코프 입니다. 함수 스코프와 매우 똑같지만 함수에서만이 아닌 {} 블럭 내부에서만 유효하게 되는 스코프입니다. Let과 cosnt가 이 블럭스코프의 특징을 가지고 있습니다.

let string = "hi";

var dummy = "cool";

function helloWolrd(){

let string = "hello world";

var dummy = "boo";

console.log(string);

}

helloWolrd();

console.log(dummy)

{

let string = "hi";

const dummy = "who"

}

자바스크립트에서 블럭을 사용하는 곳들은 어디가 있었죠? 한번 이야기 해봅시다.

1. If
2. For
3. While
4. Switch
5. 함수
6. 등

많은 예약어들에서 블럭을 사용하는데, 이외에도 그냥 전역에서 블럭을 사용해줄 수 있습니다. 만약 블럭을 전역에서 사용하는 경우 의미는 전역스코프를 사용하지 않고 블럭스코프를 사용하겠다는 의미가 됩니다.

다음은 변수 숨김과 스코프 중첩을 알아보도록 하겠습니다.

function first(){

function second(){

function third(){

function four(){

}

}

}

}

만약 다음과 같은 상황이 존재 한다면 최 하단에 있는 블럭은 그 위에있는 블럭들을 공유하게 됩니다. 이를 스코프 중첩이라고 합니다.

function first(){

let a = 10;

function second(){

function third(){

let a = 20;

function four(){

console.log(a);

}

}

}

}

변수 a가 설정되어 있는 경우 이 변수는 함수 내부에서 선언되었기 때문에 함수 스코프를 가지게 됩니다. 그러나 아래에 있는 하위 함수들 또한 같은 first함수 내부에서 선언 되었기 때문에 같은 해당 스코프를 공유하게 됩니다. 이때 이 four함수 에서 재미난 사실을 알 수 있는데, four함수 내부에는 a 변수가 존재 하지 않습니다. 이럴 때 자바스크립트는 해당 변수를 찾을 때까지 자신의 상위 스코프를 모두 뒤져 해당 변수가 있는지 확인하고 없다면 undefined를 반환합니다.

이 때 변수는 스코프를 거슬러 올라가다 가장 가까운 스코프에서 변수명이 일치하는 변수를 찾으면 탐색을 중단하고 해당 변수를 사용하게 됩니다. 상위에 같은 이름의 변수가 있더라도 말이죠. 이를 변수 숨기기라고 합니다.