WEB UI BASIC

# JavaScript 2탄

NHN NEXT 우재우

본 자료는 NHN NEXT 윤지수 교수님의 자료를 바탕으로 제작했습니다.

# 본격적으로 JS 시작합니다~

# Fasten your seatbelt!!

# 오늘은 함수에 대해 알아봅시다

# 펑크숀(function)ㅋ

# 함수를 선언하는 법

```
function  functionName (argument1, argument2 …) {
   …statements…
   (return returnValue;)
}
```

JavaScript에서 함수는 argument 타입을 입력하지 않아요!

그리고 return 타입도 입력할 필요가 없습니다.

return;이 있으면 return 하는 거죠 ㅎㅎㅎ

# 함수를 써봅시다!

```
function printName(name) {
   console.log( “ My name is  “  + name);
}

printName( “ Smith ” );
```

```
My name is Smith
```

쉽죠?

# 이렇게도 할 수 있어요!

```
var functionName = function (argument1, argument2 …) {
   …statements…
   (return returnValue;)
}
```

JavaScript에서 함수는 argument 타입을 입력하지 않아요!

그리고 return 타입도 입력할 필요가 없습니다.

return;이 있으면 return 하는 거죠 ㅎㅎㅎ

# 다음 함수를 바꿔봅시다!

앞에서 작성한 printName 함수를 변수 선언 함수로 바꿔봅시다!

```
function printName(name) {
   console.log( “ My name is  “  + name);
}
```

==> ???

# 뭐가 다른걸까요?

```
function addNum(num) {
   return num++;
}

addNum(3);
```

```
var addNum = function(num) {
   return num++;
}

addNum(3);
```

함수 이름을 변수처럼 쓸 수 있다는 것은 어떤 의미일까요?

함수 이름(addNum)은 단지 함수를 가리키는 이름일 뿐이란거죠!

# 그래서 함수를...

그래서 함수를 인자로도 쓸 수 있고, 리턴값으로 쓸 수도 있죠!!!

자바스크립트는 정말 엄청난 ~~변태같은~~ 언어 아닌가요?

아래 코드를 차근차근 뜯어봅시다~

```
var newFunc = function(arg) {return ++arg;};
function addNum(num, func) {
   value = func(num);
   return function(){return ++value;};
}
resultFunc = addNum(3, newFunc);
resultFunc();
```

# JS, 너... 너는!!!

# 하...함..수형 언...어?!?!

JavaScript는 함수형 언어입니다!

엄밀히 따지면 명령형, 함수형, 객체지향형 언어입니다.

한 마디로 짬뽕이죠 ㅋㅋㅋㅋㅋ

JavaScript는 C나 JAVA처럼 상태도 관리하지만

앞선 예시와 같이 함수를 가지고 놀 수도 있어서

함수형 언어로도 볼 수 있는 것이죠.

그냥 다재다능한 친구 정도로 해둡시다 ㅎㅎㅎ

내가 그의 이름을 불러 주기 전에는

그는 다만

하나의 몸짓에 지나지 않았건만

너는 왜...

# (익명)즉시실행함수!

함수를 호출하지 않아도 스스로 실행되는 함수를 말해요~

```
(function() {
   console.log( “ HELLO!!! ” );
})();
```

또는

```
(function() {
   console.log( “ HELLO!!! ” );
}());
```

```
(function() {
  console.log( “ HELLO!!! ” );
})();
```

요게 포인트죠!

# 즉시실행함수는 왜 쓰는거죠?

보통 함수를 선언하면 인터프리터가 스크립트를 로딩 시점에
함수를 변수이름에 할당합니다. 즉시실행함수를 사용하면 함수를
따로 저장하지 않고 인터프리터가 runtime에 해석되고 실행됩니다.
그래서 성능향상에 도움이 됩니다.
또 즉시실행함수 범위 내에 변수를 전역으로 선언하지 않고
외부에서의 접근을 제한할 수 있어서 코드 충돌을 줄일 수 있습니다.

# 잠시 쉬는 시간!!!

# SCOPE!

이번에는 scope에 대해 이야기하려 합니다.

scope는 해당 변수가 유효한 범위를 뜻하는데요,

scope는 크게 global scope와 local scope가 있죠?

JavaScript는 특이한 scope를 가지고 있어요.

차근차근 하나씩 살펴봅시다.

# Function Scope

JavaScript는 function 단위로 scope가 만들어져요!

```
function printName() {
   var smith = “ Smith ”;
}
printName();
console.log(smith);
```

이건 그냥 당연하게 받아들이면 되죠??

# 만약 var가 없다면??

```
function printName() {
    smith = “ Smith ”;
}

printName();
console.log(smith);
```

헐!!!

# var의 중요성

```
var value = 100;

function printNum() {
    value = 0;
}

printNum();
console.log(value);

value = ???
```

var를 붙이지 않으면

function 안에 있더라도

전역변수가 됩니다ㅠ

Scope로 인한 충격과 공포는

아직 끝나지 않았다!

# for {} 무쓸모

```
function test() {
   for (var i = 0; i < 10; i++) {
      continue;
   }
   console.log(i);
}

test();
```

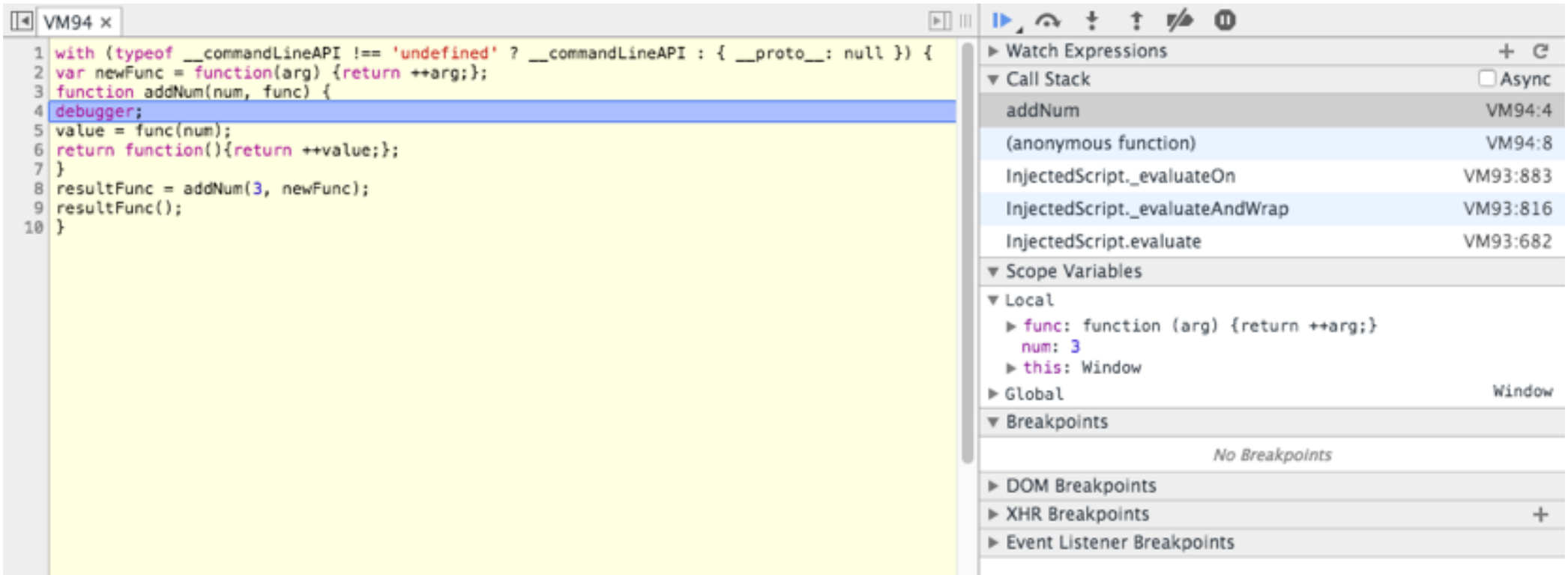
for문 쓰고나서

index(i) 쓸 때는 주의해야겠죠?

# 디버그(Debug)

```
var newFunc = function(arg) {return ++arg;};
function addNum(num, func) {
   debugger;
   value = func(num);
   return function(){return ++value;};
}
resultFunc = addNum(3, newFunc);
resultFunc();
```

# 디버그(Debug)




디버그 창을 살펴봅시다!!!

변수와 함수에 대해 다뤘으니

다음 시간에는 JS를 이용해서

DOM을 조작하고, Event를 추가합시다!

다음 주까지 숙제는

윤지수 교수님이 만든 자료 중에

Object와 Array에 관한 자료를

혼자 읽고 공부하는 것입니다!

https://github.com/WooJaeWoo/WEB_UI_BASIC 에 가시면 자료가 다 있어요!