

S H O U T · O U R · P A S S I O N · T O G E T H E R

inno SOPT

1차 세미나

SHOUT OUR PASSION TOGETHER
SOPT

01

Javascript 소개

- 언어의 특징

02

Javascript 문법 - 1

- 자료형
- 함수
- JSON
- 연산자

03

Javascript 문법 - 2

- 변수 범위
- 호이스팅
- 코드 작성

04

AWS 회원가입

01

Javascript 소개

Javascript의 특징(Scripting Language)

- 가볍고 손쉽게 작성이 가능한 프로그래밍 언어
- HTML 내에 코드를 삽입하여 사용
- 거의 모든 브라우저에서 실행 가능
- 이벤트 중심 프로그래밍
- Node.js 의 등장으로 서버 사이드 개발 가능
- 클래스는 지원하지 않지만 객체 지향 프로그래밍 가능
- 모든 객체는 프로토타입을 가짐
- 일급객체와 클로저의 특성으로 함수 지향 프로그래밍 가능
- Javascript의 표준이 ECMAScript이고 현재는 ECMAScript6(줄여서 ES6)가 표준

02

Javascript 문법 - 1

- Javascript는 변수 타입 표시를 하지 않고, 값이 할당되는 과정에서 자동으로 자료형이 결정
=> 그래서 같은 변수에 여러 자료형의 값을 대입할 수 있다
- String, Number, Boolean, undefined, null, Object
- ES6에서 Symbol이 추가됨 : primitive type 이기 때문에 new로 생성 X
- 자료형 var, let, const로 표시 -> Scope의 차이
- 기본타입 : Number, String, Boolean, undefined, null + Symbol
- 참조타입 : Array, Function (Object)

Number

- 다른 언어들처럼 여러 타입(int, short, long, float) 등이 있지 않음
- 정수값과 실수값 구분하지 않음
- 모든 숫자를 실수로 표현(64bit의 floating point type으로 저장)
- 비트연산도 가능. But 느림

String

- 2byte의 값들이 연속적으로 나열된 것
- 0기반의 인덱싱 사용(다른 언어와 동일)
- 문자 하나를 표현하는 문자형은 제공하지 않음(길이가 1인 문자열)
- “ (작은 따옴표) , ” (큰 따옴표)
- 여러 문자열을 ‘+’ 를 이용해 연결할 수 있음
(+가 addition 과 concatenation 으로 사용되기 때문에 주의하여 사용해야함)
- 문자열을 수정하는 모든 메소드는 새로운 문자열을 반환

Boolean

- true, false 중 하나의 값을 가짐.
- 비교의 결과로 생성
- false : 0, "", undefined, null, NaN(Not A Number)

null, undefined

- null은 '객체가 아님'을 뜻하는 특수한 값
- undefined는 '값이 없음'을 나타내는 값 - 값 자체가 없음, 초기화 되어있지 않거나 존재하지 않는 값에 접근할 때
- 시스템 레벨에서는 undefined, 일반적인 프로그램 레벨에서는 null을 사용

Object

- 속성(property) : 키(key) - 값(value)의 쌍으로 이루어짐
- 속성 끼리는 쉼표(,)로 구분
- 키는 문자열만 가능. 따옴표 있어도, 없어도 가능

Array

- [] 로 감싼다. 값들이 순서대로 나열.
- 배열의 원소에는 다른 데이터 타입들이 들어갈 수 있음.

Function

- 다음 페이지에!

일급 객체

- Javascript는 함수형 프로그래밍 언어(Functional Programming)
- C : Imperative Programming / Java : Object-Oriented Programming
- Javascript에서는 객체가 일급 객체(First Class Object)이다 => 함수도 객체! => 함수가 일급 객체!!
- 익명 함수(Anonymous Function) : 함수의 이름이 없는 함수
- 고차 함수(Higher-Order Function) : 함수를 인자로 받거나 반환할 수 있는 함수

```
console.log('*** 변수나 데이터 구조안에 담을 수 있다. ***');
var func1 = function() {
  console.log(1);
}
func1();    // 1

console.log('*** 파라미터로 전달할 수 있다. ***');
var func2_1 = function() {
  return 2;
}
var func2_2 = function(value) {
  console.log(value);
}
func2_2(func2_1());    // 2

console.log('*** 반환값(return value)으로 사용할 수 있다. ***');
var func3 = function() {
  return function() {
    console.log(3);
  }
}
func3()();    // 3

console.log('*** 할당에 사용된 이름과 관계없이 고유한 구별이 가능하다. ***');
var func4 = function func44() {
  console.log(4);
}
func4();    // 4

console.log('*** 동적으로 프로퍼티 할당이 가능하다. ***');
var func5 = function() {
  console.log(5);
}
func5.property = '55';
console.log(func5.property);    // 5
```

일급 객체 의 조건

- 변수나 데이터 구조안에 담을 수 있다.
- 다른 함수의 파라미터로 전달할 수 있다.
- 반환값(return value)으로 사용할 수 있다.
- 할당에 사용된 이름과 관계없이 고유한 구별이 가능하다.
- 동적으로 프로퍼티 할당이 가능하다.

함수의 생성 방법

- 함수 선언문을 사용한 생성
- 함수 표현식을 사용한 생성
- 생성자 함수를 사용한 생성

```
function func1(n) {  
  console.log('func1 : ' + n);  
}
```

함수 선언문을 이용한 생성

- 코드 블록 자체는 실행 가능 코드가 아님
- 함수명이 반드시 정의되어야 함
- 일반적인 함수 정의와 같음
- 매개변수의 타입 표시하지 않음

```
var func2 = function(n) {  
  console.log('func2 : ' + n);  
}
```

```
var func4 = function func44() {  
  console.log(4);  
}  
func4();    // 4
```

함수 표현식을 이용한 생성

- 함수 리터럴(표현식)로 특정 변수에 할당되거나 즉시 실행 가능한 코드 블록
- 일급 객체이므로 변수에 할당 가능
- 함수 이름은 선택 사항. 함수 표현식에서 사용된 함수 이름이 외부에서 접근할 수 없음

생성자 함수를 이용한 생성

- 함수가 일급 객체이기 때문에 객체 생성 방식과 비슷
- new 키워드로 객체를 생성할 수 있는 함수
- prototype 을 사용하여 한 번만 메소드를 생성할 수 있음

JSON(JavaScript Object Notation)

- 속성 - 값의 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷
- 일반적으로 서버에서 클라이언트로 데이터를 보낼 때 사용하는 포맷
- 자바스크립트에서 파생되어 자바스크립트의 구문 형식을 따르지만 언어 독립형 데이터 포맷
- 공식 인터넷 미디어 타입 : application/json

기본 자료형

- Number
- String : 항상 큰 따옴표로 묶어야함.
- Boolean : true / false
- Array : 0 이상의 임의의 종류의 값으로 이루어진 순서가 있는 리스트. 대괄호
- Object : 순서가 없는 이름 - 값 의 쌍으로 이루어진 집합. 이름(키)은 문자열. 중괄호
- null

연산자의 종류

- 보통의 연산자는 다른 언어와 유사
- 일반적이지 않은 Javascript 의 연산자에 대해 설명
- 동등 연산자(Equality) vs 일치 연산자(Identity)
- + 연산자
- / 연산자
- typeof 연산자

관계 연산자

- 두 피연산자의 관계를 검사하여, 관계가 성립하면 true, 아니면 false를 반환
- 항상 Boolean 값을 리턴

동등 연산자 vs 일치 연산자

- ==, != (동등 연산자, Equality) : 다른 타입일 경우 형변환(묵시적 형변환)을 한 후에 값을 비교함. 타입이 달라도 동등할 수 있음.
- ===, !== (일치 연산자, Identity) : 형변환을 하지 않고 현재 상태로 값을 비교함. 타입이 다르면 일치하지 않는다.

+ 연산자

- Number + Number => 더하기 연산 수행
- String + String , Number + String , String + Number => 문자열 연결 연산 수행
- 여러 숫자, 문자열을 결합 시 연산자가 실행된 순서에 따라 연산 결과가 바뀜

/ 연산자

- console.log(5 / 3) = ?

typeof 연산자

- 피연산자의 타입을 String 형태로 반환하는 연산자

03

Javascript 문법 - 2

변수 범위(Variable Scope) : 변수가 존재하는 컨텍스트(함수가 실행되거나 변수를 참조할 때의 환경).

어디에서 변수가 접근할 수 있는지, 그 컨텍스트에서 변수에 접근할 수 있는지를 명시적으로 나타냄.

- 지역 변수(함수 수준 범위) : 함수 내에 정의된 변수는 지역 범위를 가지며, 해당 함수와 내부 함수에서만 접근이 가능.

지역변수는 함수내에서 전역변수보다 높은 우선순위를 가짐.

- 전역 변수 : 함수의 외부에서 선언된 모든 변수는 전역 범위(global scope)를 가짐.

전역 컨텍스트(scope)는 window 객체를 가리킴.

```

var var1 = 'var1';
var var2 = 'var2';
var var1 = 'var3';           // 재선언 가능
function a() {
  var var1 = "var1_2";
  var2 = 'var2_2';           // 전역 변수
  console.log(var1);         // 지역 변수
}
a();                          // var1_2
console.log(var1);           // var3
console.log(var2);           // var2_2

let let1 = "let1";
function b() {
  let let1 = "let1_2";
  let let2 = "let2";
  let let2 = "let3";         // 재선언 불가(Syntax Error)
  console.log(let1+' / '+let2);
}
b();                          // let1_2 / let2
console.log(let1);           // let1
console.log(let2);           // Reference Error

const constvalue = 10;
console.log(constvalue);

const constvalue = 20;       // Syntax Error
console.log(constvalue);

constvalue = 20;             // 재할당 불가(Type Error)
console.log(constvalue);

const constvalue2;          // Syntax Error

```

var / let / const

- var : function-scoped. 변수 재선언 가능.
- let : block-scoped. 변수 재선언 불가능. 변수 재할당 가능
- const : block-scoped. 변수 재선언 불가능. 변수 재할당 불가능

호이스팅 : 변수의 정의가 그 범위에 따라 선언과 할당이 분리되는 것을 의미.

- 함수내에서 정의되었을 경우 : 선언이 함수의 최상위
- 함수 바깥에서 정의되었을 경우 : 전역 컨텍스트의 최상위로 변경
- 변수의 선언이 초기화나 할당시에 발생하는 것이 아니라, 최상위로 호이스트 됨
- 함수 선언문 방식만 호이스팅이 제대로 이루어짐

함수 선언은 변수 선언을 덮어씀

But, 변수에 값이 할당될 경우 반대로 변수가 함수 선언을 덮어씀

```
var test;

function test() {
  console.log('test');
}

console.log(typeof test);    // function

var test = 'test';

function test() {
  console.log('test');
}

console.log(typeof test);    // string
```

- 변수는 camelCase로
- 세미콜론은 항상 붙여준다.
- 변수 선언은 스코프 상단에
- 동치 연산자(==, !=) 보다는 일치 연산자(===, !==)를 사용
- 중괄호 '{ ' 위치는 선언문과 같은 줄에

K & R 방식

```
if (expression) {  
    statement  
}
```

BSD 방식

```
if (expression)  
{  
    statement  
}
```


04

AWS 회원가입

04 AWS 회원가입

01
AWS 회원가입

SHOUT OUR PASSION TOGETHER
SOPT

Create an AWS account

Email address

Password

Confirm password

AWS account name ⓘ

Continue

[Sign in to an existing AWS account](#)

© 2018 Amazon Web Services, Inc. or its affiliates.
All rights reserved.

[Privacy Policy](#) | [Terms of Use](#)



Contact Information

All fields are
required.

Please select the account type and complete the fields below with your contact details.

Account type ⓘ

☐ Professional

☒ Personal

Full name

Lee Sangeun

Phone number

01062114771

Country/Region

Korea, Republic of

Address

Street, P.O. Box, Company Name, c/o

Apartment, suite, unit, building, floor, etc.

City

State / Province or region

Postal code

☐ Check here to indicate that you have read and agree to the terms of the [AWS Customer Agreement](#)

Create Account and Continue

<https://aws.amazon.com/ko/>에 입력하여 회원가입을 진행합니다.

주소는 영어로 입력해주세요.
네이버에 주소 영어 변환하면
간편합니다.

결제 정보

결제 정보를 입력해야 자격 증명을 확인할 수 있습니다. 사용량이 **AWS 프리 티어 한도**를 초과하지 않는 한 요금을 청구하지 않습니다. 자세한 정보는 [FAQ](#)를 참조하십시오.

신용/직불 카드 번호

카드 만료일

03 2018

카드 소유자 이름

청구지 주소

☒ 내 연락처 주소 사용

☐ 새 주소 사용

보안 전송



전화 번호 확인

AWS에서 자동화 시스템을 사용하여 즉시 전화를 겁니다. 메시지가 나오면 전화 키패드로 AWS 웹 사이트에서 받은 4자리 숫자를 입력하십시오.

전화 번호 제공

아래 정보를 입력하고 "지금 전화하기" 버튼을 클릭하십시오.

국가/리전 코드

대한민국 (+82)

전화번호

01062114771

내선 번호

보안 확인

34ay8f

위에 보이는 문자를 입력하십시오.

지금 전화하기

지원 플랜 선택

AWS는 귀하의 요구를 충족할 수 있는 선별된 지원 플랜을 제공합니다. 귀하의 AWS 사용량에 맞는 지원 플랜을 선택하십시오. [자세히 알아보기](#)



기본 플랜

무료

- 모든 계정에 포함됨
- 포럼 및 리소스에 대한 상시 셀프 서비스 액세스
- 보안 및 성능 개선을 돕는 모범 사례 확인
- 상태 확인 및 알림에 대한 액세스



개발자 플랜

월 \$29부터

- 이온 채택, 테스트 및 개발을 위해 사용됨
- 업무 시간 중 AWS Support로의 이메일 액세스
- 하나의 기본 연락처로 무제한의 지원 사례를 열 수 있음
- 비 프로덕션 시스템에 대한 12시간 이내 응답



비즈니스 플랜

월 \$100부터

- 프로덕션 워크로드 및 비즈니스 크리티컬 의존성에 사용됨
- 채팅, 전화 및 이메일을 통한 AWS Support로의 상시 액세스
- 무제한의 연락처로 무제한의 지원 사례를 열 수 있음
- 프로덕션 시스템에 대한 1시간 이내 응답

엔터프라이즈 수준의 지원이 필요하신가요?

AWS에서 비즈니스와 미션 크리티컬 워크로드 실행(월 15,000 USD부터 시작됨)에 관한 추가 정보는 계정 관리자에게 문의하십시오. [자세히 알아보기](#)

기본 플랜을 선택하세요.

S H O U T · O U R · P A S S I O N · T O G E T H E R

inno SOPT

총회가기

SHOUT OUR PASSION TOGETHER
SOPT