SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

JavaScript

SAMSUNG SW ACADEMY FOR YOUTH

Web - FrontEnd

JavaScript

- JavaScript
- 기본문법
- 객체
- 함수

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

JavaScript

자바스크립트란?

- 프로토타입 기반 객체 생성을 지원하는 동적 스크립트 언어
- 웹 브라우저에서 주로 사용, Node.js를 이용하여 콘솔 환경에서 사용
- 웹 브라우저의 UI를 제어하기 위해 만들어진 프로그래밍 언어
- 자바와 기본 구문이 비슷하다. (C언어의 기본 구문을 바탕)
- 브랜든 아이크 개발(1995)
- Mocha → LiveScript → JavaScript

자바스크립트

- ECMA International : 정보 통신에 대한 표준을 제정하는 비영리 표준화 기구
- ECMAScript : ECMA-262 기술 규격에 의해 정의된 범용 스크립트 언어
- ECMAScript6 (ES6) : ECMA에서 제안하는 6번째 표준 명세 (2015년 발표)

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

기본 문법

기본 문법

HTML 자바스크립트 사용

- <script> </script> 태그를 사용
- 문서 내의 위치의 제약이 없다.

```
<html>
<head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
    <script>console.log('head');</script>
</head>
<body>
    <script>console.log('body');</script>
</body>
</html>
```

기본 문법

외부스크립트 참조하기

- .js 확장자를 가진 파일을 생성
- html 문서에서 <script src="외부파일의 위치"></script>

```
console.log('hello');
```

outer.js

```
<body>
<h1>외부 스크립트 파일 참조</h1>
<script src="outer.js">console.log("실행되지 않는 부분임");</script>
<script>
    console.log("파일 호출과 별도의 태그를 만들어서 실행");
</script>
</body>
```

html

기본 문법

- 주석 (Comment)
  - // 한 줄 주석
  - /* */ 여러 줄 주석

```
// 한줄 주석
/*
    여러줄 주석
*/
```

기본 문법

- 변수 (Variable)
  - 자바스크립트의 변수 타입은 가리키는 값에 대한 타입을 나타낸다.
  - var, let, const 키워드를 이용해서 변수를 선언
  - var를 이용한 변수의 선언일 경우 중복 선언이 가능
  - undefined 는 변수에 아무 값도 없어서 타입을 알 수 없는 경우를 말한다.
  - 동적 타입 : 대입되는 값에 따라서 용도가 변경되는 방식
  - 문자, $, _ 로 시작, 대소문자 구분, 예약어 사용 x

https://www.w3schools.com/js/js_reserved.asp

기본 문법

- var
  - 재 선언 가능, 재 할당 가능
  - ES6 이전에 변수 선언 시 사용
  - 호이스팅 (Hoisting) 특성이 있음.
  - 함수 스코프

```
var id = "hong";
console.log(id);
var id = "kim";
console.log(id);
```

https://developer.mozilla.org/en-US/docs/Glossary/Hoisting

기본 문법

- let
  - 재 선언 불가, 재 할당 가능
  - 블록 스코프

```
let id = "hong";
console.log(id);
let id = "kim";
console.log(id);
```

기본 문법

- const
  - 재 선언 불가, 재 할당 불가
  - 블록 스코프
  - 대문자 SNAKE_CASE 사용
  - 선언 시 값을 할당해야 함.
  - 상수로 사용

```
const id = "hong";
console.log(id);
id = "kim";  // error
console.log(id);
```

- undefined
  - 변수에 값이 대입되지 않은 상태

```
var name;
console.log(name);
```

기본 문법

- 데이터 타입 (Data Type)
  - 기본 데이터 타입 (Primitive Type)
    - String, Number, Boolean, null, undefined
  - 객체 타입 (Reference Type)
    - Object - function, array 등

*es6 추가된 타입 : Symbol(변경 불가능한 기본타입)

```
var num1 = 10; // number
var num2 = 10.2;
var msg = "hi"; // string
var bool = true; // boolean
var nullVal = null; // null
var unVal; // undefined
var obj = {}; // object
var obj2 = new Object(); //object
```

기본 문법

- typeof - 변수의 자료형 검사
  - typeof 데이터
  - typeof (데이터)
  - typeof 의 결과는 문자열 반환
  - null의 데이터 타입은 null이 아닌 object (설계 실수)
  - function 은 기능을 가진 객체

*es6 추가된 타입 : Symbol(변경 불가능한 기본타입)

```
console.log(typeof ("test")); // string
console.log(typeof (10)); // number
console.log(typeof (true)); // boolean
console.log(typeof (null)); // object
console.log(typeof (a)); // undefined
console.log(typeof ({})); // object
console.log(typeof (function () {})); // function
console.log(typeof Symbol()) // symbol
```

기본 문법

- 동적 데이터 타입 - 다양한 값의 대입이 가능

```
var val = 10;
console.log(val, typeof (val)); // 10 'number'
val = "hello";
console.log(val, typeof (val)); // hello string
val = true;
console.log(val, typeof (val)); // true 'boolean'
```

기본 문법

- 숫자형 (Number)
  - 정수와 실수로 나누어 구분하지 않음. (부동소수점 형식)
  - 일반적인 숫자 외 특수 숫자 포함 (Infinity, NaN …)
  - e 를 활용하여 거듭제곱 표현 가능

https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/NaN

기본 문법

- 문자열 (String)
  - “ ” 로 감싼다.
  - ‘ ’ 로 감싼다.
  - `` (backtick) 으로 감싼다. → Template Literal (ES6)
    여러 줄 입력이 가능 - 공백, 줄 넘김 유지
    문자열 내 ${변수명}을 이용하여 변수와 문자열을 결합
  - UTF-16 형식

```
let msg = "자바스크립트 문자열";
msg = '자바스크립트 문자열';
msg = `자바스크립트 문자열`;

let name = "홍길동";
msg = `나의 이름은 "${name}"입니다.`;

let msg2 = `저의
이름은
홍길동 입니다.`;
```

기본 문법

- 문자열 (String) 연산
  - 문자열과 숫자 타입의 + 연산 → 문자열
  - 문자열과 숫자 타입의 + 연산 이외 → 숫자

```
console.log(1 + "20"); // 120
console.log("1" + "20"); // 120
console.log("1" + 20); // 120
console.log("100" - 8); // 92
console.log("100" * 8); // 800
```

기본 문법

자바스크립트 false

- 아래의 5가지 값은 false로 인식, 나머지 값은 true로 인식
- null
- undefined
- 0
- '' (빈문자열)
- NaN

```
console.log("!!0", !!0);
console.log("!!''", !!'');
console.log("!!null", !!null);
console.log("!!undefined", !!undefined);
console.log("!!NaN", !!NaN);
var id;
if (id) {
    console.log("id가 값이 있는 경우임...");
}
else {
    console.log("id가 값이 없는 경우임...");
}
```

https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Boolean

기본 문법

- 연산자 (Operator)
  - + (덧셈), 단 항 사용시 Number() 와 동일한 역할
  - - (뺄셈)
  - * (곱셈)
  - / (나눗셈)
  - % (나머지)
  - ** (거듭제곱)
  - = (할당 연산자)
  - +=, -=, *=, … (복합 할당 연산자)
  - ++, -- (증감 연산자)

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

기본 문법

- 일치 연산자
  - 값과 타입이 일치하는지 체크
  - === , !==

```
var i = 100;
var j = "100";

console.log("i == j", i == j);  // true
console.log("i === j", i === j);  // false
```

https://262.ecma-international.org/5.1/#sec-11.9.3

기본 문법

- 제어문 (Java 유사)
  - 조건문 (Condition)
    - if
    - switch
  - 반복문 (Loop)
    - for
    - while
    - do-while

기본 문법

- 배열 (Array)
  - 배열의 생성 : [] 또는 Array() 활용
  - 배열의 크기는 동적으로 변경된다.
  - 크기가 지정되어 있지 않은 경우에도 데이터의 입력 가능
  - 배열의 길이는 가장 큰 인덱스 _ 1 한 값이다.
  - 배열은 여러가지의 데이터 타입을 하나의 배열에 입력할 수 있다.
  - push 함수를 이용하여 데이터 추가 가능

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

기본 문법

배열 (Array)

```
var arr1 = [];
var arr2 = new Array();

arr1[0] = 10;
arr1[2] = 30;
console.log(arr1[0], arr1[1], arr1[2]); // 10 undefined 30
console.log(arr1.length); // 3

arr1[3] = "문자열";
arr1[4] = {};
arr1[5] = [1, 2, 3];
arr1[6] = true;
arr1.push("추가");
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

객체 - object

객체(Object)

- 객체 (Object)
  - 객체는 문자열로 이름을 붙인 값들의 집합체이다. (Key : Value)
  - 객체에 저장하는 값을 프로퍼티(Property) 라고 한다.
  - 객체는 prototype 이라는 특별한 프로퍼티를 가지고 있다.

https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Object

객체(Object)

- 객체 (Object) 만들기
  - 객체 리터럴 이용 : { }
  - Object 생성자 이용 : new Object()
  - 생성자 함수 이용

```
let member1 = {}
let member2 = new Object();
function Member() {}
let member3 = new Member()
```

객체(Object)

- 객체 (Object) 생성 시 프로퍼티 추가

```
let member1 = {id: "shy", email: "ssafy@a.com"}
function Member(id, email) {
    this.id = id;
    this.email = email;
}
let member2 = new Member("shy", "ssafy@a.com");
```

객체(Object)

객체 (Object) 프로퍼티

- .(dot) 또는 [] 를 이용하여 프로퍼티의 조회 및 변경을 처리한다.

```
let student = {
  name: "김싸피",
  age: 20,
  hobby: ["공부", "숙면"],
  "favorite singer": "아이유",
};

console.log(student.name);
console.log(student[age]); // 에러
console.log(student.hobby);
console.log(student["favorite singer"]);
```

```
var member = {};
member ["id"] = "ssafy";
member.name = "싸피";
```

객체 (Object) 프로퍼티 - 추가 / 수정 / 삭제

```
var member = {"id": "hong", "email": "hong@a.com"};

// 동적인 프로퍼티 추가

member.name = "홍길동";

console.log(member);
```

```
let member = {id: "shy", email: "ssafy@a.com"}
member["id"] = "ssafy";
member.email = "ssafy@ssafy.com";
```

```
let member = { id: "shy", email: "ssafy@a.com" };
delete member.id;
console.log(member);
```

객체(Object)

- 객체 변수에는 주소가 저장되어 공유 가능

```
let member1 = {id: "hong", email: "hong@a.com"}
let member2 = member1;
member2.id = "kang";

console.log(member1.id); // kang
console.log(member2.id); // kang
```

객체(Object)

- 함수안에서의 this는 함수를 호출한 객체

```
var m1 = {name: "홍길동"};
var m2 = {name: "배수지"};
function msg () {
    console.log(this);
    console.log(this.name + "님이 입장함...");
}
m1.msg = msg;
m2.msg = msg;
m1.msg();
m2.msg();
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

함수 - function

함수

함수 특징

- 자바스크립트에서 함수는 객체 타입으로 값처럼 사용이 가능하다.
- 함수를 변수에 대입하거나 매개변수로 넘길 수 있다.
- 배열의 요소에 넣거나 객체의 프로퍼티로 설정이 가능하다.
- 매개변수의 개수가 일치하지 않아도 호출이 가능하다.
- JavaScript의 함수는 일급 객체(First-class citizen)에 해당
  - 변수에 할당 가능
  - 함수의 매개변수로 전달 가능
  - 함수의 반환 값으로 사용가능

https://developer.mozilla.org/ko/docs/Glossary/First-class_Function

함수 만들기

- 함수 선언식

function 함수명() { 함수 내용 }

- 함수 표현식

let 함수명 = function() { 함수 내용 }

함수

- 함수 선언식 (function declaration)
  - 함수의 이름과 함께 정의하는 방식
  - 함수의 이름
  - 매개 변수
  - 내용
  - 호이스팅 됨

```
function func( ) {
    console.log('선언식');
}
func( );
```

- 함수 표현식 (function expression)
  - 익명함수로 정의가능
  - 매개 변수
  - 내용

```
let func = function ( ) {
    console.log('표현식');
};
func( );
```

선언식 vs 표현식

- 선언식 함수는 호이스팅의 영향을 받아 함수 선언 이전에 호출이 가능하다.
- 표현식 함수는 선언 이전에 호출이 불가능하다.

```
func( );

function func( ) {

    console.log('선언식');

}
```

호출가능

```
func( );

let func = function ( ) {

    console.log('표현식');

};
```

호출불가능

함수의 리턴

- 함수의 실행 결과로 함수를 반환할 수 있다.
- 함수가 특별한 값을 리턴 하지 않은 경우 undefined가 반환된다.

```
function func( ) {
    return function (num1, num2) {
        return num1 + num2;
    }
}
function func2( ) { }
```

```
let callFn = func( );
let result = callFn(100, 200);
console.log(result);


console.log( func2( ) );  // undefined 출력
```

함수의 호출

- 정의된 함수를 호출 시 함수를 값으로 넘길 수 있다.

```
function func( callFn ) {
    callFn('hello');
}
function fn( msg ) {
    console.log(msg);
 }
```

```
func( fn );
```

함수 매개변수

- 함수는 호출 시 매개변수의 영향을 받지 않는다.
- arguments 라는 함수 내부의 프로퍼티를 이용하여 매개변수의 처리가 가능하다.
- 자바스크립트의 함수는 오버로딩 개념을 지원하지 않는다.
- 기본 인자 (default arguments)를 사용할 수 있다.

함수 매개변수

```
function fn1(num) {
   console.log( " fn1 ", num)
}
```

```
fn1( );  // undefined
fn1(100);  // 100
fn1(100, 100);  // 100
```

함수 매개변수

```
function fn( ) {
    console.log(arguments.length);
    for (let i = 0; i < arguments.length; i++) {
        console.log(arguments[i]);
    }
}
```

```
fn(1);
1
1
fn(1, 10, 100);
3
1
10
100
```

함수 매개변수

```
function fn() {
    console.log(1);
}
function fn() {
    console.log(2);
}
function fn(num) {
    console.log(num);
}
```

```
fn();
// undefined
fn(1);
1
```

화살표 함수 (Arrow Function)

- ES6에서 추가된 개념
- 함수를 심플하게 정의할 수 있도록 해준다.
- 형태
  (매개변수) => { 명령어 }
- 작성순서

1. function 키워드 삭제
2. ( ) 안에 함수가 사용할 파라미터 이름 작성
3. 화살표 (=>) 를 붙인다.
4. { } 를 작성하고 블록 안에 함수가 실행할 코드 작성

화살표 함수 (Arrow Function)

```
function func() {
    console.log("함수선언식 func");
}
func();
```

```
var func = () => {
    console.log("화살표 func");
};
func();
```

화살표 함수 (Arrow Function)

- 매개변수가 하나일 경우 ( ) 를 생략할 수 있다.

```
function func(num) {
    console.log("func", num);
};
func(1);
```

```
var func = (num) => {
    console.log( " 화살표 func", num);
};
func(2);

var func = num => {
    console.log( " 화살표 func", num);
};
func(3);
```

화살표 함수 (Arrow Function)

- 실행 문장이 하나일 경우 { } 을 생략할 수 있다.

```
function func(num) {

    console.log(num * num);

}
```

```
var func = (num) => console.log(num * num);
```

화살표 함수 (Arrow Function)

- 실행되는 하나의 문장이 return 문일 경우 return 키워드를 생략해야 한다.

```
function func(num) {
    return num * num;
}
```

```
var func = (num) => return num * num;  (X)

var func = (num) => num * num;  (O)
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미