SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Vue.js

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

SAMSUNG SW ACADEMY FOR YOUTH

목차

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(ECMAScript 2015)

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(var, let, const)

삼성 청년 SW 아카데미

|| var VS let VS const.

✓ 선언 및 할당.

키워드	구분	선언위치	재선언
var	변수	전역 스코프(함수레벨)	가능
let	변수	해당 스코프(블록레벨)	불가능
const	상수	해당 스코프(블록레벨)	불가능

✓ var : 같은 이름으로 선언 및 재할당 가능.

```
// var 예약어는 똑같은 변수를 재선언할 수 있다.
var v = 10;
console.log(v);
var v = 20;
console.log(v);
```

|| var VS let VS const.

✓ 선언 및 할당.

키워드	구분	선언위치	재선언
var	변수	전역 스코프(함수레벨)	가능
let	변수	해당 스코프(블록레벨)	불가능
const	상수	해당 스코프(블록레벨)	불가능

✓ let : 같은 이름으로 재선언은 불가능하고 재할당 가능.

```
// let 예약어는 한번 선언한 똑같은 변수를 재선언할 수 없다.
let a = 10;
let a = 2; // Cannot redeclare block-scoped variable 'a'
```

✓ const : 값 변경 불가능.

```
// const 예약어는 한번 할당한 값을 변경할 수 없다.
const b = 20;
b = 30; // Uncaught TypeError: Assignment to constant variable.
```

var VS let VS const.

✓ 선언 및 할당.

```
// 객체 { } 또는 배열 [ ]로 선언했을 경우 객체의 속성과 배열의 요소는 변경 가능.
const user = {
  name: "안효인",
  age: 25,
};
console.log(user.name); // 안효인
user.name = "홍길동";
console.log(user.name); // 홍길동

const num = [];
console.log(num); // []
num.push(10);
console.log(num); // [10]
```

Scope.

✓ 대부분의 프로그래밍 언어는 Block-level scope이나 JavaScript는 Function-level scope임.

함수 레벨 스코프 (Function-level scope).

함수 내에서 선언된 변수는 함수 내에서만 유효하며 함수 외부에서는 참조 불가.
함수 내부에서 선언한 변수는 지역 변수이며 함수 외부에서 선언한 변수는 모두 전역변수이다.

블록 레벨 스코프 (Block-level scope).

모든 코드 블록(함수, 제어문, 조건문, try/catch문 등) 내에서 선언된 변수는 코드 블록 내에서만 유효하며 블록 외부에서는 참조 불가능. 코드 블록 내부에서 선언한 변수는 지역 변수.

```
function test() {
    var x = 10;
    console.log(x); // 10
}
test();
console.log(x); // Uncaught ReferenceError: x is not defined
```

Scope.

- ✓ var.

```
// var의 유효범위 : 함수레벨 스코프(function-level scope)
var x = 100;
function print() {
  var x = 200;
  console.log(x); // 200
  {
    var x = 300;
    console.log(x); // 300
    x = 400;
  }
  console.log(x); // 400
}
print();
console.log(x); // 100
```

Scope.

✓ let.

```
// let의 유효범위 : 블록레벨 스코프(block-level scope)
let y = 100;
function print() {
  let y = 200;
  console.log(y); // 200
  {
    let y = 300;
    console.log(y); // 300
    y = 400;
  }
  console.log(y); // 200
}
print();
console.log(y); // 100
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(Property Shorthand)

삼성 청년 SW 아카데미

Property Shorthand (단축 속성명).

✓ 객체를 정의할 때 객체의 key값과 value에 할당할 변수명이 같을 경우 value를 생략할 수 있다.

```
// ES6 이전
const id = "ssafy",
  name = "안효인",
  age = 30;

const user = {
  id: id,
  name: name,
  age: age,
};
console.log(user);
```

```
{ id: 'ssafy', name: '안효인', age: 30 }
```

Property Shorthand (단축 속성명).

✓ 객체를 정의할 때 객체의 key값과 value에 할당할 변수명이 같을 경우 value를 생략할 수 있다.

```
// ES6 이후
const id = "ssafy",
  name = "안효인",
  age = 30;

const user = {
  id,
  name,
  age,
};
console.log(user);
```

```
{ id: 'ssafy', name: '안효인', age: 30 }
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(Concise Method)

삼성 청년 SW 아카데미

Concise Method (간결한 메소드).

✓ 객체 내에서 좀 더 간결하게 메소드를 사용.

```
// ES6 이전
const id = "ssafy",
  name = "안효인",
  age = 30;

const user = {
  id: id,
  name: name,
  age: age,
  info: function () {
    console.log("user info : " + this.id + " " + this.name + " " + this.age);
  },
};
user.info();
```

Concise Method (간결한 메소드).

✓ 객체 내에서 좀 더 간결하게 메소드를 사용.

```
// ES6 이후
const id = "ssafy",
  name = "안효인",
  age = 30;

const user = {
  id,
  name,
  age,
  info() {
    console.log("user info : " + this.id + " " + this.name + " " + this.age);
  },
};
user.info();
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6 (Destructuring Assignment)

삼성 청년 SW 아카데미

Destructuring Assignment (구조 분해 할당).

✓ 배열이나 객체에 입력된 값을 개별적인 변수에 할당하는 간편한 방식 제공.

```
// 배열
const areas = ["광주", "구미", "서울", "대전", "부울경"];

// ES6 이전
{
  const a1 = areas[0];
  const a2 = areas[1];
  const a3 = areas[2];
  const a4 = areas[3];
  const a5 = areas[4];

  console.log(a1, a2, a3, a4, a5);
}

// ES6 이후
{
  const [a1, a2, a3, a4, a5] = areas;
  console.log(a1, a2, a3, a4, a5);
}
```

ES6 (ECMAScript 2015)

Destructuring Assignment (구조 분해 할당).

✓ 배열이나 객체에 입력된 값을 개별적인 변수에 할당하는 간편한 방식 제공.

```
// 객체
const user = {
  id: "ssafy",
  name: "안효인",
  age: 30,
};

// ES6 이전
{
  let id = user.id;
  let name = user.name;
  let age = user.age;
  console.log(id, name, age);
}
```

Destructuring Assignment (구조 분해 할당).

✓ 배열이나 객체에 입력된 값을 개별적인 변수에 할당하는 간편한 방식 제공.

```
// ES6 이후
// 객체의 property와 변수명이 같을 경우.
{
  let { id, name, age } = user;
  console.log(id, name, age);
}

// 변수명을 객체의 property명과 다르게 만들 경우.
{
  let { id: user_id, name: user_name, age: user_age } = user;
  console.log(user_id, user_name, user_age);
}
```

Destructuring Assignment (구조 분해 할당).

✓ 배열이나 객체에 입력된 값을 개별적인 변수에 할당하는 간편한 방식 제공.

```
// 객체를 return 하는 함수.
function getUser() {
  return {
    id: "ssafy",
    name: "안효인",
    age: 30,
  };
}
```

```
// ES6 이전
{
  let user = getUser();
  let id = user.id;
  let name = user.name;
  let age = user.age;
  console.log(id, name, age);
}

// ES6 이후
{
  let { id, name, age } = getUser();
  console.log(id, name, age);
}
```

Destructuring Assignment (구조 분해 할당).

✓ 배열이나 객체에 입력된 값을 개별적인 변수에 할당하는 간편한 방식 제공.

```
// ES6 이전
function showUserInfo1(user) {
console.log("showUserInfo1 call");
let id = user.id;
let naem = user.name;
let age = user.age;
console.log("아이디 : " + id);
console.log("이름 : " + name);
console.log("나이 : " + age);
}
showUserInfo1(user);
```

```
// ES6 이후
function showUserInfo2({ id, name, age }) {
console.log("showUserInfo2 call");
console.log("아이디 : " + id);
console.log("이름 : " + name);
console.log("나이 : " + age);
}
showUserInfo2(user);
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(Spread Syntax)

삼성 청년 SW 아카데미

Spread Syntax (전개 구문).

- ✓ Spread operator는 반복 가능한(iterable) 객체에 적용할 수 있는 문법.
- ✓ 배열이나 문자열 등을 풀어서 하나 하나로 전개 시킬 수 있다.

```
const user1 = { id: "ssafy1" };
const user2 = { id: "ssafy2" };

const arr = [user1, user2];

// array copy
const copyArr = [...arr];
console.log(arr, copyArr);
console.log(arr === copyArr);
```

```
// 주의 : spread operator의 경우 값 복사가
아닌 주소를 가져오기 때문에 값을 바꿀 경우
모두 변경.
user1.id = "ssafy8";
console.log(arr, copyArr);

// 배열 복사 + 새로운 객체 추가.
const addArr = [...arr, { id: "ssafy3" }];
console.log(arr, addArr);
```

Spread Syntax (전개 구문).

- ✓ Spread operator는 반복 가능한(iterable) 객체에 적용할 수 있는 문법.
- ✓ 배열이나 문자열 등을 풀어서 하나 하나로 전개 시킬 수 있다.

```
const team1 = ["서울", "광주"];
const team2 = ["대전", "부울경", "구미"];

// 배열 복사
const team3 = [...team1];
console.log(team3);

// 배열의 추가 요소로 넣기
const team4 = ["서울", ...team2, "광주"];
console.log(team4);

// 두 배열 이어 붙이기.
const team = [...team1, ...team2];
console.log(team);
```

Spread Syntax (전개 구문).

- ✓ Spread operator는 반복 가능한(iterable) 객체에 적용할 수 있는 문법.
- ✓ 배열이나 문자열 등을 풀어서 하나 하나로 전개 시킬 수 있다.

```
// 객체 복사
const copyUser = { ...user1 };
// const copyUser = user1;
console.log(copyUser);
copyUser.id = "ssafy100";
console.log(user1, copyUser);
console.log(user1 === copyUser);

// 객체 복사 + 새로운 property 추가.
const copyUser2 = {
    ...user1, name: "안효인"
};
console.log(copyUser2);
```

```
// 주의 : spread operator의 경우 값 복사가 /
/ object merge (병합)
const u1 = { id1: "ssafy1" };
const u2 = { id2: "ssafy2" };
const u = { ...u1, ...u2 };
console.log(u);

// 주의점 : key가 같은 object를 병합할 경우
마지막 obj의 값이 설정.
const user = { ...user1, ...user2 };
console.log(user);
```

Spread Syntax (전개 구문).

- ✓ Spread operator는 반복 가능한(iterable) 객체에 적용할 수 있는 문법.
- ✓ 배열이나 문자열 등을 풀어서 하나 하나로 전개 시킬 수 있다.

```
// 함수 호출 시 인자로 전달.
const num = [1, 3, 5, 7];

function plus(x, y, z) {
  return x + y + z;
}

const result = plus(...num);
console.log(result);
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(Default Parameter)

삼성 청년 SW 아카데미

ES6 (ECMAScript 2015)

Default Parameter.

✓ Default parameter는 함수에 전달된 파라미터가 undefined이거나 전달되지 않았을 경우, 설정한 값으로 초기화 됨.

```
// ES6 이전
function print1(msg) {
  console.log(msg);
}
print1("hello ssafy");
print1();

// ES6 이후
function print2(msg = "안녕하세요 싸피") {
  console.log(msg);
}
print2("hello ssafy");
print2();
```

Default Parameter.

✓ Default parameter는 함수에 전달된 파라미터가 undefined이거나 전달되지 않았을 경우, 설정한 값으로 초기화 됨.

```
function getUserId(userId = "ssafy9") {
  return userId;
}

console.log(getUserId());
console.log(getUserId(undefined));
console.log(getUserId(null));
console.log(getUserId("troment"));

// default parameter각 reference type일 경우 호출될 때마다 새로운 객체를 생성함.
function appendArr(val, array = []) {
  array.push(val);
  return array;
}

console.log(appendArr(10)); // [10]
console.log(appendArr(20)); // [20] or [10, 20] ???
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(Template String)

삼성 청년 SW 아카데미

ES6 (ECMAScript 2015)

Template String.

- ✓ 문자열에 변수를 포함시킬 때 좀 더 직관적이고 편하게 사용하기 위한 기능.
- ✓ 변수를 넣고자 하는 부분에 ${ var } 키워드를 사용해 변수를 넣어 줌.
- ✓ 문자열 사용시 큰따옴표(") 대신 백틱(`)을 사용.
- ✓ 백틱 내부의 줄 바꿈은 그대로 적용.

```
// Template String(`, 백틱 사용)

const id = "ssafy",
name = "안효인",
age = 30;
// ES6 이전
console.log(name + "(" + id + ")님의 나이는 " + age + "입니다.");
// ES6 이후
console.log(`${name}(${id})님의


나이는 ${age}입니다.`);
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(Arrow Function)

삼성 청년 SW 아카데미

Arrow Function (화살표 함수).

- ✓ 기존 함수의 선언 `function name(param) { }` 의 형식을 축약하여 사용.
- ✓ 함수의 이름이 없는 익명 함수이므로 변수에 담아서 사용.

```
const name = (param) => { };
```

- 매개변수에 따른 설정.

```
// 매개변수가 없을 경우.
() => {};
// 매개변수가 한 개 있을 경우. (param)의 소괄호 생략 가능.
(param) => {};
// 매개변수가 여러 개 있을 경우. (param1, param2)의 소괄호 생략 불가능.
(param1, param2) => {};
```

Arrow Function (화살표 함수).

- function body에 따른 설정.

```
(x) => {
  return x + 10;
};
// body의 내용이 한 줄일 경우 {} 생략 가능하고 자동으로 결과값이 return된다. 위와 동일.
(x) => x + 10;

// object return시 () 사용.
() => {
  return { id: "ssafy" };
};
() => ({ id: "ssafy" });

// // body가 여러 줄일 경우 {}, return 생략 불가.
(x) => {
  const y = x + 100;
  return y;
};
```

ES6 (ECMAScript 2015)

Arrow Function (화살표 함수).

```
// ES6 이전
function func1() {
  return 100;
}
let result = func1();
console.log(result);

// ES6 이후
// const func2 = () => {
// return 100;
// };
const func2 = () => 100;
result = func2();
console.log(result);
```

```
// ES6 이전
function func3(x) {
  return x + 100;
}
result = func3(50);
console.log(result);

// ES6 이후
const func4 = (x) => x + 100;
result = func4(50);
console.log(result);
```

ES6 (ECMAScript 2015)

Arrow Function (화살표 함수).

```
// ES6 이전
function func5(x, y) {
  return x + y;
}
result = func5(50, 80);
console.log(result);

// ES6 이후
const func6 = (x, y) => x + y;
result = func6(50, 80);
console.log(result);
```

```
// ES6 이전
function func7() {
  return {
    id: "ssafy",
  };
}
result = func7();
console.log(result.id);

// ES6 이후
const func8 = () => ({ id: "ssafy" });
result = func8();
console.log(result.id);
```

ES6 (ECMAScript 2015)

Arrow Function (화살표 함수).

```
// ES6 이전
function func9(x) {
  const y = x * 3;
  return y;
}
result = func9(10);
console.log(result);

// ES6 이후
const func10 = (x) => {
const y = x * 3;
  return y;
};
result = func10(10);
console.log(result);
```

```
// Arrow Function에서는
// this가 바인딩 되지 않음.
const user = {
   id: "ssafy8",
   age: 20,
// info() {
//   console.log(this.id, this);
// },
// this 사용 불가.
   info: () => console.log(this.id, this),
};
user.info();
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

이어서..

삼성 청년 SW 아카데미

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ES6
(Module)

삼성 청년 SW 아카데미

module.

- ✓ project의 규모가 커지면서 코드를 여러 파일과 폴더로 나누어 작성하고 파일간에 효율적으로 불러올 수 있도록 해주는 시스템이 필요해짐.

```
<script type="module" src="app.js"></script>
```

- ✓ 구형 브라우저의 경우 module을 지원하지 않는 문제 있음 ➔ webpack, parcel등의 모듈 번들러를 통해 변환과정을 거친 뒤 사용.

- ✓ module?
  - ES2015 모듈은 기본적으로 JavaScript 코드를 담고 있는 파일.
  - Import 또는 export 구문을 사용.
  - 브라우저의 지원 여부 확인 : https://developer.mozilla.org/ko/docs/web/javascript/guide/modules

module.

✓ **module system.**

- CommonJS (NodeJS) : 웹 브라우저 밖에서도 동작될 수 있는 모듈 규칙 설립.
- AMD (Asynchronous Module Definition) : 비동기적으로 모듈을 로딩.
- ESM (ECMAScript Module, ECMA215, es6) : 자바스크립트 자체 모듈.

module.

- ✓ module scope.
  - 모듈의 가장 바깥쪽에서 선언된 이름은 전역 스코프가 아니라 모듈 스코프에서 선언됨.
    - 모듈 스코프에 선언된 이름은 export 해주지 않으면 해당 모듈 내부에서만 접근 가능.
    - 따라서 여러 모듈의 가장 바깥쪽에서 같은 이름으로 변수, 함수, 클래스를 선언하더라도, 서로 다른 스코프에서 선언되기 때문에 이름의 충돌이 발생하지 않음.
  - 01.var-scope.js

```
const userId = "ssafy";

// var-scope.js가 module로 사용되고 있다면 'undefined'가 출력 됨.
console.log(window.userId);
```

  - 01.main.html

```
<script type="module" src="./01.var-scope.js"></script>
```

  - result

```
undefined
```

module.

✓ export & import.

- module scope에 선언된 이름은 `export` 구문을 통해 다른 파일에서 사용가능.
  - 변수, 함수, 클래스 export 가능.
  - 개별 export, 한번에 export, export default

```
export const name = "안효인";
```

```
export { name, age, info, Student };
```

```
export default function () { }
```

- export 된 이름을 다른 파일에서 `import` 구문을 통해 가져온 뒤 사용가능.

```
import { export한 리스트 } from 'module 파일'
```

module.

- ✓ export & import.
  - export를 이용한 개별 변수, 함수, 클래스를 export.
  - 이름을 선언하는 구문 앞에 export를 붙이면 선언과 동시에 export 가능.
  - 02.declare.js

```
export const name = "안효인";
export const age = 30;

export function info() {
  return `이름 : ${name}, 나이 : ${age}`;
}

export class Student {}
```

02.main.html

```
<script type="module">
import { name, age, info, Student } from "./02.declare.js";

console.log(name, age);

const result = info();
console.log(result);
</script>
```

```
안효인 30
이름 : 안효인, 나이 : 30
```

module.

- ✓ export & import.
  - export를 이용한 변수, 함수, 클래스를 한번에 export.
  - 파일의 마지막에 export를 선언하여 한번에 export 가능.
  - 03.declare.js

```
const name = "안효인";
const age = 30;

function info() {
  return `이름 : ${name}, 나이 : ${age}`;
}

class Student {}

export { name, age, info, Student };
```

03.main.html

```
<script type="module">
import { name, age, info, Student } from "./03.declare.js";

console.log(name, age);

const result = info();
console.log(result);
</script>
```

```
안효인 30
이름 : 안효인, 나이 : 30
```

module.

- ✓ export & import.
  - export default를 통해 모듈을 대표하는 하나의 값을 지정하고 모듈을 가져올 때 { }를 사용하지 않아도 됨.
  - import 시 이름을 다르게 가능.
  - 04.declare.js

```
export const name = "안효인";
export const age = 30;

export default function info() {
  return `이름 : ${name}, 나이 : ${age}`;
}

export class Student {}
```

04.main.html

```
<script type="module">
import info2, { name, age, Student } from "./04.declare.js";

console.log(name, age);

const result = info2();
console.log(result);
</script>
```

```
안효인 30
이름 : 안효인, 나이 : 30
```

module.

- ✓ export & import.
  - export default : 객체형식으로 하게 되면 한번에 import 가능.

- 05.declare.js

```
export default {
  name: "안효인",
  age: 30,

  info() {
    return `이름 : ${this.name}, 나이 : ${this.age}`;
  },
};
```

05.main.html

```
<script type="module">
import student from "./05.declare.js";

console.log(student.name, student.age);

const result = student.info();
console.log(result);
</script>
```

```
안효인 30
이름 : 안효인, 나이 : 30
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미