SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Vue.js 핵심정리

ES6 일부 기능

- let, const
- property shorthand, concise method
- for ~ of
- template string

ES6 - let

let

- 중복선언 불가능
- ECMA6 부터 선언된 블럭에서만 사용할 수 있는 블록변수(let)가 제공됨
- 호이스팅 대상에서 제외

ES6 - let

```
var msg = 1;
if (msg === 1) {
  var msg = 2;
  console.log(msg);    2
}
console.log(msg);      2
```

```
var msg = "1";
if (msg == "1") {
  let msg = "2";       호이스팅 X
  console.log(msg);    2
}
console.log(msg);      1
```

ES6 - const

const

- 상수로 사용
- 선언시 값을 할당해야 함
- 호이스팅 대상에서 제외
- let 키워드와 값을 변경할 수 없다는 것만 제외하고 동일

ES6 - const

```
const msg1 = '상수';
msg1 = '상수2';    에러
console.log(msg1);
```

```
const msg2;   에러
msg2 = "선언시 값을 할당해야 한다.";
console.log(msg2);
```

ES6 - property shorthand

```
const id = 'ssafy',
name = ' 싸피 ' ,
age = 3;
const member = {
  id: id,
  name: name,
  age: age
};
```

```
const id = 'ssafy',
name = ' 싸피 ' ,
age = 3;
const member = {
  id,
  name,
  age
};
```

ES6 - concise method

```
...
const member = {
  id: id,
  name: name,
  age: age,
  info: function () {
     console.log('info');
  }
};
```

```
...
const member = {
  id,
  name,
  age,
  info() {
     console.log('info');
  }
};
```

ES6 – for ~ of

```
const data = [1, 10, 100];
for (let element of data) {
  console.log(element);
}
```

```
1
10
100
```

ES6 – template string

- ``(백틱) 사용
- jsp의 el과 문법 형식이 동일
- 문자열과 변수의 결합시 편리함
- multiline 지원

```
const name = "홍길동";
const age = 22;
console.log(name + "의 나이는 " + age + "세 입니다.");
console.log(`${name}의 나이는 ${age}세 입니다.`);
```

arrow function

function(param) { 코드 } 형태를 축약

(param) => { **코드** }

함수 이름이 없는 익명함수이므로 사용시 변수에 담아서 사용

arrow function

```
(param) => { 코드 };

param => { 코드 };

( ) => { 코드 };

( ) => 코드;

( ) => ( { key: value } );
```

arrow function

```
const func = function ( ) {
    console.log("익명 func");
}
```

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

arrow function

```
const func = function (num) {
    console.log("익명 func");
}
```

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

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

arrow function

```
const func = function (num1, num2) {
    console.log("익명 func");
}
```

```
const func = ( num1, num2 ) => {
    console.log("화살표 func");
};

// 불가능
const func = num1, num2 => {
    console.log("화살표 func");
};
```

arrow function

```
const func = function (num) {
  return num * num;
};
console.log(func(1));    // 1
```

```
const func = (num) => {
  return num * num;
};
console.log(func(1));  // 1

const func = (num) => num * num;
console.log(func(1));  // 1
```

ES6 - Destructuring

객체(배열, 객체)에 입력된 값을 개별적인 변수에 할당하는 간편 방식 제공

ES6 - Destructuring

```
const member = {
    id: 'aaa',
    name: 'bbb',
    age: 22,
};
```

```
let id = member.id;
let name = member.name;
let age = member.age;

ES6
let {id, name, age} = member;
```

ES6 - Destructuring

```
function getMember() {
    return {
        id: 'aaa',
        name: 'bbb',
        age: 22,
    };
}
```

```
let member = getMember();
let id = member.id;
let name = member.name;
let age = member.age;


ES6
let {id, name, age} =  getMember();
```

ES6 - Destructuring

```
let member = {
    id: 'aaa',
    name: 'bbb',
    age: 22,
};
```

```
function showMember(member) {
  console.log("아이디 :", member.id);
  console.log("이름 :", member.name);
  console.log("나이 :", member.age);
}
```

```
function showMember({ id, name, age }) {
  console.log("아이디 :", id);
  console.log("이름 :", name);
  console.log("나이 :", age);
}
```

ES6 - SpreadSyntax

전개 구문

```
const user1 = { id: 'ssafy1' };
const user2 = { id: 'ssafy2' };
const array = [user1, user2];
```

배열복사

```
const arrayCopy = [...array];
console.log(array, arrayCopy);
```

주의 주소값복사가 일어난다

ES6 - SpreadSyntax

전개 구문

```
const user1 = { id: 'ssafy1' };
const user2 = { id: 'ssafy2' };
const array = [user1, user2];
```

```
add
const arrayAdd = [...array, { id: 'ssafy3' }];
console.log(arrayAdd);

array concat
const team1 = ['대전', '광주'];
const team2 = ['구미', '서울'];
const team = [...team1, ...team2];
```

ES6 - SpreadSyntax

전개 구문

```
const user1 = { id: 'ssafy1' };
const user2 = { id: 'ssafy2' };
const array = [user1, user2];
```

```
object copy
  const user3 = { ...user1 };
  console.log(user3);

obejct merge(병합)
  const u1 = { id1: 'ssafy1' };
  const u2 = { id2: 'ssafy2' };
  const u = { ...u1, ...u2 };
```

ES6 - DefaultParameter

기본 인자값 설정

```
function print (msg = 'inital msg') {
  console.log(msg);
}
```

ES6 - Promise

자바스크립트 비동기 처리객체

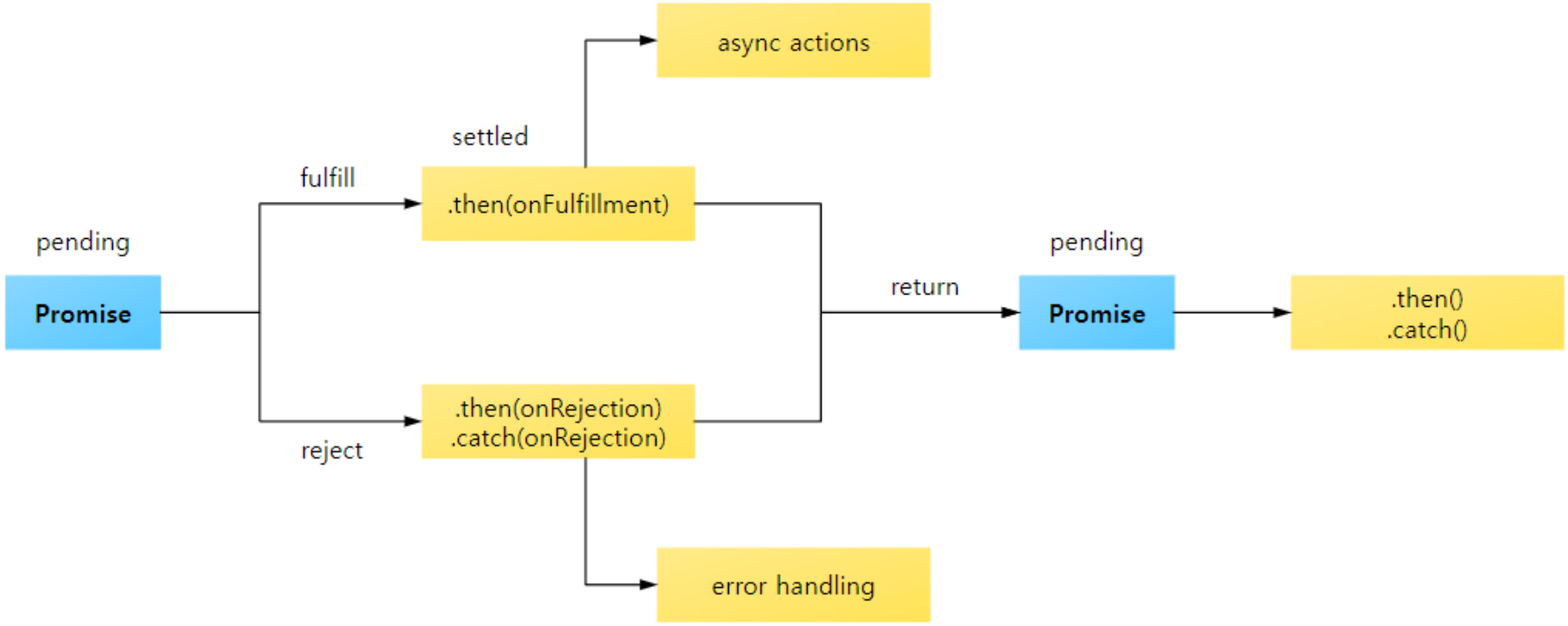
ES6 - Promise

자바스크립트 비동기 처리객체

처리상태

- Pending(대기) : 비동기 처리 로직이 아직 완료되지 않은 상태
- Fulfilled(이행) : 비동기 처리가 완료되어 프로미스가 결과 값을 반환해준 상태
- Rejected(실패) : 비동기 처리가 실패하거나 오류가 발생한 상태

ES6 - Promise



프로미스 처리 흐름 - 출처 : MDN

ES6 - Promise

자바스크립트 비동기 처리객체

처리상태

- Pending(대기) : 비동기 처리 로직이 아직 완료되지 않은 상태
- Fulfilled(이행) : 비동기 처리가 완료되어 프로미스가 결과 값을 반환해준 상태
- Rejected(실패) : 비동기 처리가 실패하거나 오류가 발생한 상태

ES6 - Promise

```
const authorize = function (b) {
  return new Promise(function (resolve, reject) {
    console.log("aaaaa");
    if (b) {resolve("인증됨");
    } else {reject("인증안됨");}});}
//프로미스 실행
authorize(false)
  .then(function (result) {console.log(result); // result   인증됨})
  .catch(function (err) {console.log(err);// err    인증안됨});
```

ES6 – async , await

향상된 비동기 처리

async 함수는 프라미스를 반환하고, 프라미스가 아닌 것은 프라미스로 변환하여 반환

```
promise = new Promise((resolve, reject) => {
                setTimeout(() => {resolve('끝2~~~~~~~');}, 3000);});

 let result = await promise;
  console.log(result);
  console.log("end2.....");
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미