[JavaScript]

📌 유튜브 코딩앙마 자바스크립트 기초 강좌, 중급 강좌 정리

▼ 변수

- 변수는 문자와 숫자, $와 _만 사용
- 첫 글자는 숫자가 될 수 없음
- 예약어는 사용할 수 없음 ex) class
- 상수(const)는 가급적 대문자로 사용
- 변수명은 읽기 쉽고 이해할 수 있게 선언

```
let grade = "apple";

//...1000 lines

grade = "banana";
```

변할 수 있는 값은 `let`으로 선언

```
const PI = 3.14;

const SPEED_LIMIT = 50;

const BIRTH_DAY = '1999-02-24'
```

변하지 않는 값은 `const`로 선언

 모든 변수를 `const`로 선언 후 값을 바꿔야 할 때 `let`으로 바꾸면 좋다 !

▼ `var`, `let`, `const`

- `var` : 함수 스코프(function-scoped) - 1) 선언 및 초기화 2) 할당
- `let` : 블록 스코프(block-scoped) - 1) 선언 2) 초기화 3) 할당

- `const` : 블록 스코프(block-scoped) - 1) 선언 + 초기화 + 할당
- **Hosting** : 스코프 내부 어디서든 변수 선언은 최상위에 선언된 것처럼 행동
- **Temporal Dead Zone** : TDZ 영역의 변수들은 사용 불가
- `let`과 `const`는 TDZ에 영향을 받아 할당 전에는 사용할 수 없음

```
var name = 'Mike';
console.log(name);   //Mike;

var name = 'Jane';
console.log(name);   //Jane;
```

```
let name = 'Mike';
console.log(name);   //Mike;

let name = 'Jane';
console.log(name);   //Error!
```

> `var`과 `let`의 차이점
> `var`은 한 번 선언되 변수를 다시 선언할 수 있지만 `let`은 불가능

▼ 자료형

- `null` : 존재하지 않는 값
- `undefined` : 변수 선언 후 값을 할당하지 않은 상태
- `NaN` : Not a Number

```
const name = "Mike";
const division = name / 2; // 결과값 - NaN
```

- `typeof` : 변수의 자료형을 알아낼 수 있는 연산자

```
const name = "sumin";
const age = 25;

const a = "나는";
const b = "입니다.";

console.log(a + name + b); // 결과값 - "나는 sumin 입니다."
consoel.log(a + age + "살" + b) // 결과값 - "나는 25살 입니다."
```

> 문자형도 더하기 가능
> 숫자형과 문자형도 더하기 가능

▼ 대화상자

- `alert()` : 알려줌
- `prompt()` : 입력받음
- `confirm()` : 확인받음
- 스크립트가 일시정지되며, 스타일링을 할 수 없다는 단점

```
const name = prompt("이름을 입력하세요.", "홍길동");
alert("환영합니다, " + name + "님");
```

```
const realDelete = confirm("정말 삭제하시겠습니다?");
console.log(realDelete);
// 취소 버튼을 눌렀을 때 값은 false, 확인 버튼 눌렀을 때 값은 true
```

▼ 형변환(Type Conversion)

- `String()` : 문자형으로 변환
- `Number()` : 숫자형으로 변환
- `Boolean()` : 불린형으로 변환

```
Number(true); // 1
Number(false); // 0
Number(undefined); // NaN
Number("문자"); // NaN
```

```
Boolean(0); // false
Boolean(""); // false
Boolean(null); // false
Boolean(NaN); // false
```

자동형변환 ex) “6” / “2” = 3

```
const mathScore = prompt("수학 몇점 ?"); // 90 입력
const engScore = prompt("영어 몇점 ?"); // 80 입력
const result = (mathScore + engScore) / 2; // 결과값 - 4540
```

▼ 연산자(Operators)

- `==` 동등연산자 : 값만 비교
- `===` 일치연산자 : 값과 타입까지 비교

```
const a = 1;
const b = "1";

console.log(a == b); // true
```

```
const a = 1;
const b = "1";

console.log(a === b); // false
```

논리연산자 : AND(`&&`)가 OR(`||`)보다 우선순위가 높음

```
const gender = 'F';
const name = 'Jane';
const isAdult = true;

// 결과값 - 통과
if(gender === 'M' && name === 'Mike' || isAdult) {
  console.log('통과');
} else {
  console.log('탈락')
}


// 결과값 - 탈락
if(gender === 'M' && (name === 'Mike' || isAdult)) {
  console.log('통과');
} else {
  console.log('탈락')
}
```

▼ 반복문

- continue

```
for(let i = 0; i < 10; i++) {
  if(i%2) {
    continue;
  }
console.log(i); // 결과값 - 0 2 4 6 8
}
```

▼ 함수(function)

```
function showError() {
  alert('에러가 발생했습니다. 다시 시도해주세요.';)
}

showError();
```

지역변수(local)와 전역변수(global)

```
fuction sayHello(name) {
  let msg = 'Hello';
  if(name) {
    //msg = `Hello, ${name}`;
    msg += `, ${name}`;
    console.log(msg);
  }
}

sayHello();
sayHello(summin);
// console.log(msg); - 에러
```

```
let msg = 'Hello';

fuction sayHello(name) {
  if(name) {
    msg += `, ${name}`;
    console.log(msg);
  }
}

sayHello();
sayHello(summin);
console.log(msg);
```

💡 매개변수로 받은 값은 복사된 후 함수의 지역변수가 되므로 전체 서비스에서 공통으로 쓰는 변수를 제외하고는 지역변수를 쓰는 것이 좋다 !

함수 선언문과 함수 표현식

```
fuction sayHello() {
  console.log('Hello');
}

sayHello();
```

```
let sayHello = function() {
  console.log('Hello');
}

sayHello();
```

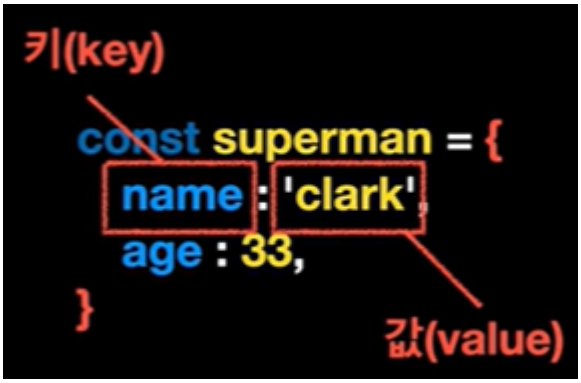
화살표 함수(arrow function)

```
let add = function(num1, num2) {
  return num1 + num2;
}
```

```
let add = (num1, num2) => {
  return  num1 + num2;
}
```

```
cosnt add = function(num1, num2) {
  const result = num1 + num2;
  return result;
}

cosnt add = (num1, num2) => {
  const result = num1 + num2;
  return result;
}

cosnt add = (num1, num2) => {
  return num1 + num2;
}

cosnt add = (num1, num2) => ( num1 + num2; )

cosnt add = (num1, num2) => num1 + num2;
```

▼ 객체(Object)



객체 접근, 추가, 삭제

```
superman.name
superman['age']
```

```
superman.gender = 'male';
superman['hairColor'] = 'black';
```

```
delete superman.hairColor;
```

단축 프로퍼티

프로퍼티 이름과 변수 이름이 동일한 경우 축약하여 사용 가능

```
function makeObject(name, age) {
  return {
    name,
    age,
    hobby : 'football',
  };
}

const Mike = makeObject('Mike', 30);
console.log(Mike);
```

`in` 과 `for … in` 프로퍼티 존재 여부 확인

```
function makeObject(name, age) {
  return {
    name,
    age,
    hobby : 'football',
  };
}

const Mike = makeObject('Mike', 30);

// 결과값 - true
console.log('age' in Mike);
// 결과값 - false
console.log('birthday' in Mike);
```

```
const Mike = {
  name : 'Mike',
  age : 30,
};

// Mike가 가진 key 출력
for(x in Mike) {
  console.log(x);
}

// Mike가 가진 key의 값 출력
for(x in Mike) {
  console.log(Mike[x]);
}
```

method

: 객체 프로퍼티로 할당된 함수

```
const superman = {
  name : 'clark',
  age : 33,
  fly() {
    console.log('날아갑니다.');
  }
};
```

`this`

```
const boy = {
  name : 'Mike',
  sayHello() {
    console.log(`Hello, I'm ${this.name}`);
  }
};

const girl = {
```

```
  name : 'Jane',
  sayHello() {
    console.log(`Hello, I'm ${this.name}`);
  }
};

boy.sayHello();
girl.sayHello();
```

```
let boy = {
  name : 'Mike',
  showName() {
    console.log(boy.name);
  }
};

let man = boy;
boy = null;

man.showName(); // Error
```

```
let boy = {
  name : 'Mike',
  showName() {
    console.log(this.name);
  }
};

let man = boy;
boy = null;

man.showName(); // 결과값 - Mike
```

화살표 함수에서는 `this` 사용 불가하다 !
화살표 함수로 메서드 작성하면 `this`는 window(전역객체)를 가리킨다.

생성자 함수

비슷한 객체를 여러번 만들어야하는 상황이 생길 때 생성자 함수를 사용

```
//생성자 함수
function Item(title, price){  // 첫 글자는 대문자로
  this.title = title;
  this.price = price;
  this.showPrice = function(){
    console.log(`가격은 ${price}원 입니`);
  }
}

let item1 = new Item('인형', 3000);  // new 연산자를 이용하여 호출
let item2 = new Item('가방', 4000);
let item3 = new Item('지갑', 9000);

item3.showPrice();
```

Computed property(계산된 프로퍼티)

```
let a = 'age';
const user = {
  name : 'Mike',
  [a] : 30,   //변수a의 할당된 값
}
```

```
const user = {
  [1+4] : 5,
  ['안녕'+'하세요'] : 'Hi',
}
// 결과값 - {5:5,'안녕하세요':'Hi'}
```

심볼(Symbol)

자바스크립트 중급 강좌(코딩악마) 19분 20초 ~

▼ 배열(Array)

- 순서가 있는 리스트
- 문자 뿐만 아니라 숫자나 객체 함수 등도 포함할 수 있음
- length : 배열의 길이
- 인덱스를 이용하여 배열 수정 가능함
- `push()` : 배열 끝에 추가

```
let days = ['월', '화', '수'];
days.push('목');
console.log(days);
// 결과값 - ['월', '화', '수', '목']
```

- `pop()` : 배열 끝 요소 제거

```
let days = ['월', '화', '수'];
days.pop();
console.log(days);
// 결과값 - ['월', '화']
```

- `unshift()` : 배열 앞에 추가

```
let days = ['월', '화', '수'];
days.unshift('일');
console.log(days);
// 결과값 - ['일', '월', '화', '수']
```

- `shift()` : 배열 앞 요소 제거

```
let days = ['월', '화', '수'];
days.shift();
console.log(days);
// 결과값 - ['화', '수']
```

반복문 `for`과 `for … of`

```
let days = ['월', '화', '수'];

for (let index = 0; index < days.length; index++) {
  console.log(days[index]);
}
```

```
let days = ['월', '화', '수'];

for (let day of days) {
  console.log(day);
}
```