



클라우드보안 과정:

자바스크립트 객체

성결대학교 정보통신공학과

정복래

brjung@sungkyul.ac.kr

Contents

학습목표

- 객체를 생성하고 사용하는 방법을 익힙니다.
- 객체의 속성과 메소드를 이해합니다.
- 생성자 함수와 프로토타입을 이해합니다.

내용

- 객체 기본
- 객체와 반복문
- 속성과 메소드
- 생성자 함수와 프로토타입
- 조금 더 나아가기

1. 객체 기본

■ 배열

코드 6-1

배열

// 배열을 선언합니다.

```
let array = ['사과', '바나나', '망고', '딸기'];
```

표 6-1 배열

인덱스	요소
0	사과
1	바나나
2	망고
3	딸기

- 배열은 요소에 접근할 때 인덱스를 사용하고, 객체는 키 사용

객체 개념

■ 현실 세계는 객체들의 집합

- 사람, 책상, 자동차, TV 등
- 객체는 자신만의 고유한 구성 속성
 - 자동차: <색상:오렌지, 배기량:3000CC, 제조사:한성, 번호:서울1-1>
 - 사람: <이름:이재문, 나이:20, 성별:남, 주소:서울>
 - 은행계좌: <소유자:황기태, 계좌번호:111, 잔액:35000원>



자동차 객체(car)



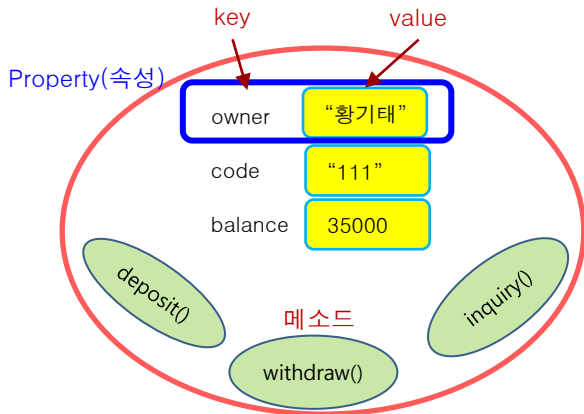
은행 계좌(account)

*발췌 : HTML5+CSS+Javascript 웹프로그래밍, 황기태저, 생능출판사

자바스크립트 객체

■ 자바스크립트 객체 구성

- 여러 개의 프로퍼티(property)와 메소드로 구성
 - 프로퍼티(property) : 객체의 고유한 속성(변수)
 - 메소드(method) : 함수



자바스크립트 객체 account

```
var account = {  
  owner    : "황기태",  
  code     : "111",  
  balance  : 35000,  
  deposit  : function() { ... },  
  withdraw : function() { ... },  
  inquiry  : function() { ... }  
};
```

account 객체를 만드는 자바스크립트 코드

1. 객체 기본

■ 객체

코드 6-2

객체 선언

```
// 객체를 선언합니다.  
let product = {  
  제품명: '7D 건조 망고',  
  유형: '당절임',  
  성분: '망고, 설탕, 메타중아황산나트륨, 치자황색소',  
  원산지: '필리핀'  
};  
  
// 출력합니다.  
console.log(product);
```

실행 결과

```
{  
  '제품명': '7D 건조 망고',  
  '유형': '당절임',  
  '성분': '망고, 설탕, 메타중아황산나트륨, 치자황색소',  
  '원산지': '필리핀'  
}
```

표 6-2 객체

키	속성
제품명	7D 건조 망고
유형	당절임
성분	망고, 설탕, 메타중아황산나트륨, 치자황색소
원산지	필리핀

1. 객체 기본

- 객체 접근

```
product['제품명'] // '7D 건조 망고'  
product['유형']   // '당절임'  
product['성분']   // '망고, 설탕, 메타중아황산나트륨, 치자황색소'  
product['원산지'] // '필리핀'
```

```
product.제품명 // '7D 건조 망고'  
product.유형   // '당절임'  
product.성분   // '망고, 설탕, 메타중아황산나트륨, 치자황색소'  
product.원산지 // '필리핀'
```

표 6-3 예제

속성 이름(키)	값
name	'바나나'
price	1200

1. 객체 기본

- [예제 6-1] 객체 생성. [표 6-3] 예제

코드 6-4

객체 생성

object.js

```
// 객체를 선언합니다.  
let object = {  
  name: '바나나',  
  price: 1200  
};  
  
// 출력합니다.  
console.log(object.name);  
console.log(object.price);
```

실행 결과

바나나
1200

2. 객체와 반복문

- [예제 6-2] 객체와 반복문
 - for in 반복문을 사용해 객체에 반복문을 적용

코드 6-5

객체와 반복문

objectLoop.js

```
// 객체를 선언합니다.  
let object = {  
  name: '바나나',  
  price: 1200  
};  
  
// 출력합니다.  
for (let key in object) {  
  console.log(`${key}: ${object[key]}`);  
}
```

실행 결과

```
name: 바나나  
price: 1200
```

3. 속성과 메소드

- 요소 : 배열 내부에 있는 값 하나하나
- 속성 : 객체 내부에 있는 값 하나하나
- 객체의 다양한 자료형

코드 6-6 다양한 자료형의 객체

```
let object = {  
  number: 273,  
  string: 'RintIanTta',  
  boolean: true,  
  array: [52, 273, 103, 32],  
  method: function () {  
  
  }  
};
```

3. 속성과 메소드

- 메소드 : 객체의 속성 중 자료형이 함수인 속성

코드 6-7

속성과 메소드

```
let object = {  
  name: '바나나',  
  price: 1200,  
  print: function () {  
    console.log(`${this.name}의 가격은 ${this.price}원입니다.`)  
  }  
};
```

3. 속성과 메소드

코드 6-8

메소드 내부에서 this 키워드

```
// 객체를 선언합니다.
let object = {
  name: '바나나',
  price: 1200,
  print: function () {
    console.log(`${this.name}의 가격은 ${this.price}원입니다.`)
  }
};

// 메소드를 호출합니다.
object.print();
```

실행 결과

바나나의 가격은 1200원입니다.

4. 클래스

- 객체 지향 프로그래밍 : 현실의 객체를 모방해서 프로그래밍

■ 개요

- 배열과 객체를 사용하면 여러 개의 데이터를 쉽게 다룰 수 있음

코드 6-9

객체 배열

```
// 상품 목록을 선언합니다.  
let products = [  
  { name: '바나나', price: 1200 },  
  { name: '사과', price: 2000 },  
  { name: '배', price: 3000 },  
  { name: '고구마', price: 700 },  
  { name: '감자', price: 600 },  
  { name: '수박', price: 5000 }  
];
```

4. 클래스

■ 객체에 메소드 추가

코드 6-10

메소드를 가진 객체의 배열

```
// 상품 목록을 선언합니다.
let products = [{
  name: '바나나',
  price: 1200,
  print: function () {
    console.log(`${this.name}의 가격은 ${this.price}원입니다.`)
  }
}, {
  name: '사과',
  price: 2000,
  print: function () {
    console.log(`${this.name}의 가격은 ${this.price}원입니다.`)
  }
}, {
```

4. 클래스

■ 객체에 메소드 추가

```
    name: '배',  
    price: 3000,  
    print: function () {  
        console.log(`${this.name}의 가격은 ${this.price}원입니다.`)  
    }  
}, {  
    name: '고구마',  
    price: 700,  
    print: function () {  
        console.log(`${this.name}의 가격은 ${this.price}원입니다.`)  
    }  
}, {  
    name: '감자',  
    price: 600,  
    print: function () {  
        console.log(`${this.name}의 가격은 ${this.price}원입니다.`)  
    }  
}, {
```

4. 클래스

```
    name: '수박',  
    price: 5000,  
    print: function () {  
        console.log(`${this.name}의 가격은 ${this.price}원입니다.`)  
    }  
  }  
};  
  
// 반복해서 출력합니다.  
for (let product of products) {  
    product.print();  
}
```


4. 클래스

- 함수를 외부로 내보낸 형태 [코드 6-10]

코드 6-11 함수를 외부로 내보낸 형태

```
// 상품 목록을 선언합니다.
let products = [
  { name: '바나나', price: 1200 },
  { name: '사과', price: 2000 },
  { name: '배', price: 3000 },
  { name: '고구마', price: 700 },
  { name: '감자', price: 600 },
  { name: '수박', price: 5000 }
];

// 함수를 선언합니다.
function printProduct(product) {
  console.log(`${product.name}의 가격은 ${product.price}원입니다.`);
}

// 반복해서 출력합니다.
for (let product of products) {
  printProduct(product);
}
```

4. 클래스

■ 클래스 선언과 인스턴스

- **클래스**: 객체지향프로그래밍 스타일을 사용하여 객체를 생성하고 관리하는 템플릿 역할 (ES2015에 처음 도입)
- 대문자로 시작하는 이름 사용

코드 6-12 클래스 선언

```
class Product {  
  
}
```

```
new 클래스_이름()
```

코드 6-13 클래스 선언

```
class Product {  
  
}
```

```
let product = new Product();
```

클래스를 기반으로 객체를 만들어 냅니다.

인스턴스 생성자

4. 클래스

■ 생성자

코드 6-14

클래스의 생성자

```
class Product {  
    constructor (name, price) {  
    }  
}
```

앞에 function 키워드를 붙이지 않으며,
constructor라는 이름으로 만듭니다.

```
let product = new Product("바나나", 1200);
```

생성자가 매개 변수로 name과 price를
받으므로 집어넣어 주었습니다.

4. 클래스

■ 속성

코드 6-15 클래스의 속성 (1)

```
class Product {  
  constructor (name, price) {  
    this.name = name  
    this.price = price  
  }  
}
```

매개 변수로 받은 값을 기반으로
name 속성과 price 속성을 만듭니다.


```
let product = new Product("바나나", 1200);  
console.log(product.name)  
console.log(product.price)
```

인스턴스의 속성에 접근합니다.

4. 클래스

■ 메소드

코드 6-16

클래스의 속성 (2)

```
class Product {
  constructor(name, price) {
    this.name = name
    this.price = price
  }

  print() {
    console.log(`${this.name}의 가격은 ${this.price}원입니다.`)
  }
}

let products = [
  new Product('바나나', 1200),
  new Product('사과', 2000),
  new Product('배', 3000),

  new Product('고구마', 700),
  new Product('감자', 600),
  new Product('수박', 500),
];

// 반복해서 출력합니다.
for (let product of products) {
  product.print();
}
```

