KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

객체

[KB] IT's Your Life

KB 국민은행

배열의 선언

배열의 구성 : 인덱스와 요소

- 배열 요소를 사용하려면 배열 이름 뒤에 인덱스로 접근
  - array[0] ⇨ 사과
  - array[2] ⇨ 망고

- 배열의 인덱스와 요소

인덱스	요소
0	사과
1	바나나
2	망고
3	딸기

객체와 배열

- 배열은 객체를 기반으로 함
- 배열은 요소에 인덱스로 접근/객체는 요소에 키로 접근

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

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

humanobject.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29
    };

    document.write('name = ' + human.name + '<br>');
    document.write('age = ' + human.age + '<br>');
  </script>
</body>
```

```
name = 김상형
age = 29
```

dogobject.html

```
<body>
  <script>
    let dog = {
      type: '치와와',
      weight: 2,
      male: true
    };

    document.write('종류 = ' + dog.type + '<br>');
    document.write('무게 = ' + dog.weight + '<br>');
    document.write('숫컷 = ' + dog.male + '<br>');
  </script>
</body>
```

객체

멤버의 참조

```
객체.멤버        // 예 human.name
객체["멤버"]     // 예 human["name']
```

객체

객체의 사용

- 객체 뒤에 대괄호를 사용하고 키를 표시 → 요소에 접근
  - product['제품명'] ⇨ '7D 건조 망고'
  - product['유형'] ⇨ '당절임'
  - product['성분'] ⇨ '망고, 설탕, 메타중아황산나트륨, 치자황색소'
  - product['원산지'] ⇨ '필리핀'

- 대괄호 외 일반적으로 사용하는 방법
  - product.제품명 ⇨ '7D 건조 망고'
  - product.유형 ⇨ '당절임'
  - product.성분 ⇨ '망고, 설탕, 메타중아황산나트륨, 치자황색소'
  - product.원산지 ⇨ '필리핀'

객체

객체의 사용

- 보조기능 사용
  → 대괄호 외 일반적인 방법을 사용하는 이유 : 보조기능 사용이 가능

- 객체의 키
  → 식별자 또는 문자열 모두 사용 가능

```
var object = {
    'with space': 273,
    'with ~!@#$%^&*()_+': 52
};
```

- 식별자가 아닌 문자를 키로 사용 하려면 대괄호를 사용해야 함

  - object['widht space'] ⇨ 273
  - object['with ~!@#$%^&*()_+'] ⇨ 52

accessmember.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29
    };

    document.write('name = ' + human['name'] + '<br>');
    document.write('age = ' + human['age'] + '<br>');
    document.write('name = ' + human[name] + '<br>');
  </script>
</body>
```

```
name = 김상형
age = 29
name = undefined
```

객체

membername.html

```
<body>
  <script>
    let human = {
      '남들이 뭐라고 부르나': '김상형',
      'how old#': 29
    };

    document.write('name = ' + human['남들이 뭐라고 부르나'] + '<br>');
    document.write('age = ' + human['how old#'] + '<br>');
  </script>
</body>
```

human.남들이 뭐라고 부르나

name = 김상형
age = 29

shortcopy.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29
    };

    let h = human;
    document.write('name = ' + h.name + '<br>');
    document.write('age = ' + h.age + '<br>');
  </script>
</body>
```

```
name = 김상형
age = 29
```

객체

값과 참조

objectref.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29
    };

    let kim = human;
    kim.name = '김태희';

    document.write('name = ' + human.name + '<br>');
    document.write('age = ' + human.age + '<br>');
  </script>
</body>
```

```
name = 김태희
age = 29
```

objectargument.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29
    };

    function changeName(h) {
      h.name = '김태희';
    }
    changeName(human);

    document.write('name = ' + human.name + '<br>');
    document.write('age = ' + human.age + '<br>');
  </script>
</body>
```

```
name = 김태희
age = 29
```

객체

속성

- 요소 : 배열 내부에 있는 값
- 속성 : 객체 내부에 있는 값

- 객체의 속성으로 가질 수 있는 값

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

    }
};
```

메서드

- 객체의 속성 중 함수 자료형인 속성
- 속성과 메서드의 구분
  → 객체 person : name 속성, eat 속성
  → eat 속성은 함수 자료형이므로 eat( ) 메서드라 부름

```
var person = {
    name: '윤인성',
    eat: function (food) { }
};

 // 메서드를 호출합니다.
 person.eat();
```

메서드

- this 키워드
  → 자기 자신이 가진 속성을 출력하고 싶을 때
  → 자신이 가진 속성임을 표시하는 방법

```
let person = {
  name : '홍길동',
  eat : function(food) {
    console.log(this.name + '이 ' + food + '을/를 먹습니다.');
  }
}

person.eat('피자');
```

nestobject.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29,
      address : {
        city : '하남시',
        dong : '덕풍동',
        bunji : 638
      }
    };

    document.write('이름 = ' + human.name + ', 나이 = ' + human.age + '<br>');
    document.write('주소 = ' + human.address.city + ' ' +
      human.address.dong + ' ' +
      human.address.bunji + '<br>');
  </script>
</body>
```

이름 = 김상형, 나이 = 29
주소 = 하남시 덕풍동 638

intro.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29,
      intro: function() {
        document.write('name = ' + this.name + '<br>');
        document.write('age = ' + this.age + '<br>');
      }
    };

    human.intro();
  </script>
</body>
```

```
name = 김상형
age = 29
```

eatsleep.html

```
<body>
  <script>
    let dog = {
      type: '치와와',
      weight: 2,
      male: true,
      eat: function() {
        document.write('할짝 할짝.<br>');
      },
      sleep: function() {
        document.write('쿨~쿨~<br>');
      },
    };

    dog.eat();
    dog.sleep();
  </script>
</body>
```

```
할짝 할짝.
쿨~쿨~
```

동적 속성 편집

- 객체 생성 이후 속성을 추가하거나 제거
  → 동적으로 속성을 추가한다
  → 동적으로 속성을 제거한다

속성 추가

- 빈 객체 생성

```
let student = { };
```

속성 추가

- 동적으로 속성 추가

```
let student = { };
student.이름 = '홍길동';
student.취미 = '악기';
student.특기 = '프로그래밍';
student.장래희망 = '생명공학과';

console.log(student)
{ '이름': '홍길동', '취미': '악기', '특기': '프로그래밍', '장래희망': '생명공학과' }
```

속성 추가

- 동적으로 메서드 추가

```
let student = { };
student.이름 = '홍길동';
student.취미 = '악기';
student.특기 = '프로그래밍';
student.장래희망 = '생명공학과';

student.toString = function() {
  for(let key in this) {
    if(key != 'toString') {
      console.log(key + '\t' + this[key])
    }
  }
}

student.toString();
```

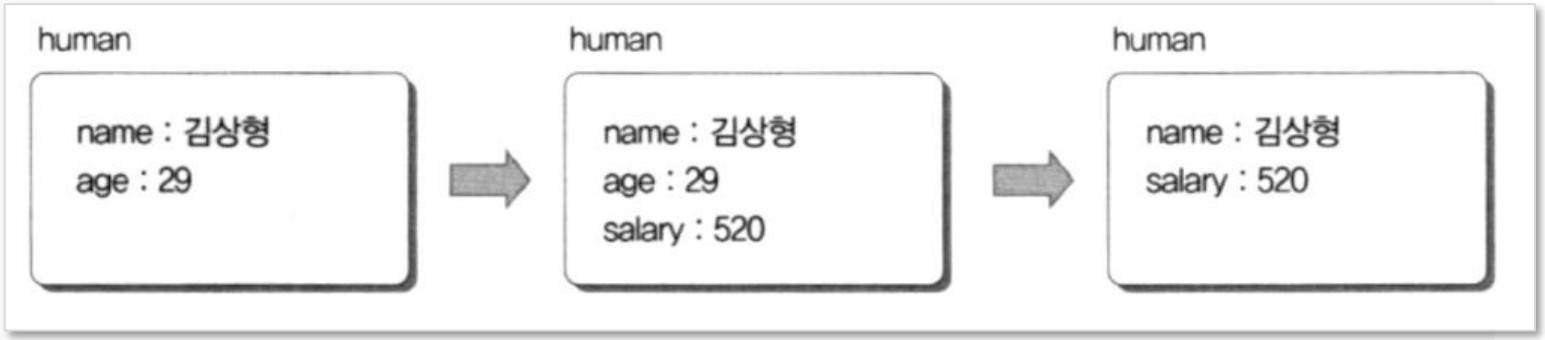
속성 제거 : delete 키워드 사용

- delete 키워드 뒤에 삭제하고자 하는 객체의 속성을 입력

```
let student = { };
student.이름 = '홍길동';
student.취미 = '악기';
student.특기 = '프로그래밍';
student.장래희망 = '생명공학과';

student.toString = function() {
  for(let key in this) {
    if(key != 'toString') {
      console.log(key + '\t' + this[key])
    }
  }
}

delete student.취미
student.toString();
```

editmember.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29
    };

    human.salary = 520;
    delete human.age;

    document.write(human.name + '의 월급 : ' + human.salary + '<br>');
    document.write(human.name + '의 나이 : ' + human.age + '<br>');
  </script>
</body>
```




```
김상형의 월급 : 520
김상형의 나이 : undefined
```

객체와 반복문

- 객체는 단순 for 반복문으로 객체의 속성을 살펴보는것이 불가능
- 객체의 속성을 모두 살펴보려면 for in 반복문 사용

```
let product = {
  name : 'Microsoft Office 2016',
  price : '15,000,000원',
  language : '한국어',
  supportOS : 'Win 32/64',
  subscription : true
}

for(let key in product) {
  console.log(key + ' : ' + product[key]);
}
```

```
name : Microsoft Office 2016
price : 15,000,000원
language : 한국어
supportOS : Win 32/64
subscription : true
```

객체 관련 키워드

- in 키워드

```
var student = {
    이름: '연하진',
    국어: 92, 수학: 98,
    영어: 96, 과학: 98
};
```

in 키워드

- 이름 속성 : true
- 성별 속성 : false

```
let student = {
  이름 : '홍길동',
  국어 : 92,
  수학 : 98,
  영어 : 96,
  과학 : 98
}

console.log('이름' in student)→ true
console.log('성별' in student)→ false
```

inoperator.html

```
<body>
  <script>
    let human = {
      name: '김상형',
      age: 29
    };

    if ('age' in human) {
      document.write('나이 정보가 있습니다.<br>');
    }
    if ('salary' in human) {
      document.write('월급 정보가 있습니다.<br>');
    }
  </script>
</body>
```

나이 정보가 있습니다.

inoperator2.html

```
<body>
  <script>
    let ar = [1, 2, 3];
    delete(ar[1]);

    if (0 in ar) {
      document.write('0번째 요소가 있습니다.<br>');
    }
    if (1 in ar) {
      document.write('1번째 요소가 있습니다.<br>');
    }
  </script>
</body>
```

0번째 요소가 있습니다.

객체와 배열을 사용한 데이터 관리

- 추상화
  → 현실에 존재하는 객체의 필요한 속성을 추출하는 작업
- 학생의 성적 총점과 평균을 계산하는 예제 작성

```
var student0 = { 이름: '윤인성', 국어: 87, 수학: 98, 영어: 88, 과학: 95 };
var student1 = { 이름: '연하진', 국어: 92, 수학: 98, 영어: 96, 과학: 98 };
var student2 = { 이름: '구지연', 국어: 76, 수학: 96, 영어: 94, 과학: 90 };
var student3 = { 이름: '나선주', 국어: 98, 수학: 92, 영어: 96, 과학: 92 };
var student4 = { 이름: '윤아린', 국어: 95, 수학: 98, 영어: 98, 과학: 98 };
var student5 = { 이름: '윤명월', 국어: 64, 수학: 88, 영어: 92, 과학: 92 };
var student6 = { 이름: '김미화', 국어: 82, 수학: 86, 영어: 98, 과학: 88 };
var student7 = { 이름: '김연화', 국어: 88, 수학: 74, 영어: 78, 과학: 92 };
var student8 = { 이름: '박아현', 국어: 97, 수학: 92, 영어: 88, 과학: 95 };
var student9 = { 이름: '서준서', 국어: 45, 수학: 52, 영어: 72, 과학: 78 };
```

객체와 배열을 사용한 데이터 관리

배열에 데이터 추가

```
var students = [];
students.push({ 이름: '윤인성', 국어: 87, 수학: 98, 영어: 88, 과학: 95 });
students.push({ 이름: '연하진', 국어: 92, 수학: 98, 영어: 96, 과학: 98 });
students.push({ 이름: '구지연', 국어: 76, 수학: 96, 영어: 94, 과학: 90 });
students.push({ 이름: '나선주', 국어: 98, 수학: 92, 영어: 96, 과학: 92 });
students.push({ 이름: '윤아린', 국어: 95, 수학: 98, 영어: 98, 과학: 98 });
students.push({ 이름: '윤명월', 국어: 64, 수학: 88, 영어: 92, 과학: 92 });
students.push({ 이름: '김미화', 국어: 82, 수학: 86, 영어: 98, 과학: 88 });
students.push({ 이름: '김연화', 국어: 88, 수학: 74, 영어: 78, 과학: 92 });
students.push({ 이름: '박아현', 국어: 97, 수학: 92, 영어: 88, 과학: 95 });
students.push({ 이름: '서준서', 국어: 45, 수학: 52, 영어: 72, 과학: 78 });
```

객체와 배열을 사용한 데이터 관리

메서드 추가

```
// 모든 students 배열 내의 객체에 메서드를 추가합니다.
for (var i in students) {
    // 총점을 구하는 메서드를 추가합니다.
    students[i].getSum = function () {
        return this.국어 + this.수학 + this.영어 + this.과학;
    };

    // 평균을 구하는 메서드를 추가합니다.
    students[i].getAverage = function () {
        return this.getSum() / 4;
    };
}
```

객체와 배열을 사용한 데이터 관리

학생 성적 출력

```
var output = '이름\t총점\t평균\n';
for (var i in students) {
    with (students[i]) {
        output += 이름 + '\t' + getSum() + '\t' + getAverage() + '\n';
    }
}
```

함수를 사용한 객체 생성

- 객체의 반영
  - 하나씩 만들어 배열에 사용 : 서로 다른 형태의 객체를 배열 안에 넣을 수 있는 장점
  - 개별적 객체를 만드는 것이 객체의 특성을 정확히 반영

함수를 사용한 객체 생성

```
function makeStudent(name, korean, math, english, science) {
  let student = {
    이름 : name,
    국어 : korean,
    수학 : math,
    영어 : english,
    과학 : science,

    getSum : function() {
      return this.국어 + this.수학 + this.영어 + this.과학;
    },

    getAverage : function() {  return this.getSum() / 4;  },

    toString : function() {
      return this.이름 + '\t' + this.getSum() + '\t'
              + this.getAverage();
    }
  };
  return student;
}
```

함수를 사용한 객체 생성

```
let students = [];
students.push(makeStudent('윤인성', 90, 83, 76, 89));
students.push(makeStudent('박찬호', 90, 83, 76, 89));
students.push(makeStudent('류현진', 90, 83, 76, 89));
students.push(makeStudent('이세돌', 90, 83, 76, 89));
students.push(makeStudent('김세진', 90, 83, 76, 89));
students.push(makeStudent('이하나', 90, 83, 76, 89));

let output ='이름\t총점\t평균\n';
for(let i in students) {
  output += students[i].toString()+ '\n';
}

console.log(output);
```

클래스

생성자 함수란?

- new 키워드를 사용해 객체 생성할 수 있는 함수

student 생성자

- student 생성자 함수를 만드는 코드

```
function Student () {

}
```

new 키워드

- new 키워드로 객체 생성

```
function Student () {

}

let student = new Student();
```

this 키워드

- 생성자 함수로 생성될 객체의 속성 지정

```
function Student(name, korean, math, english, science) {
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;
}
let student = new Student('김세진', 90, 83, 76, 89);
```

메서드 생성

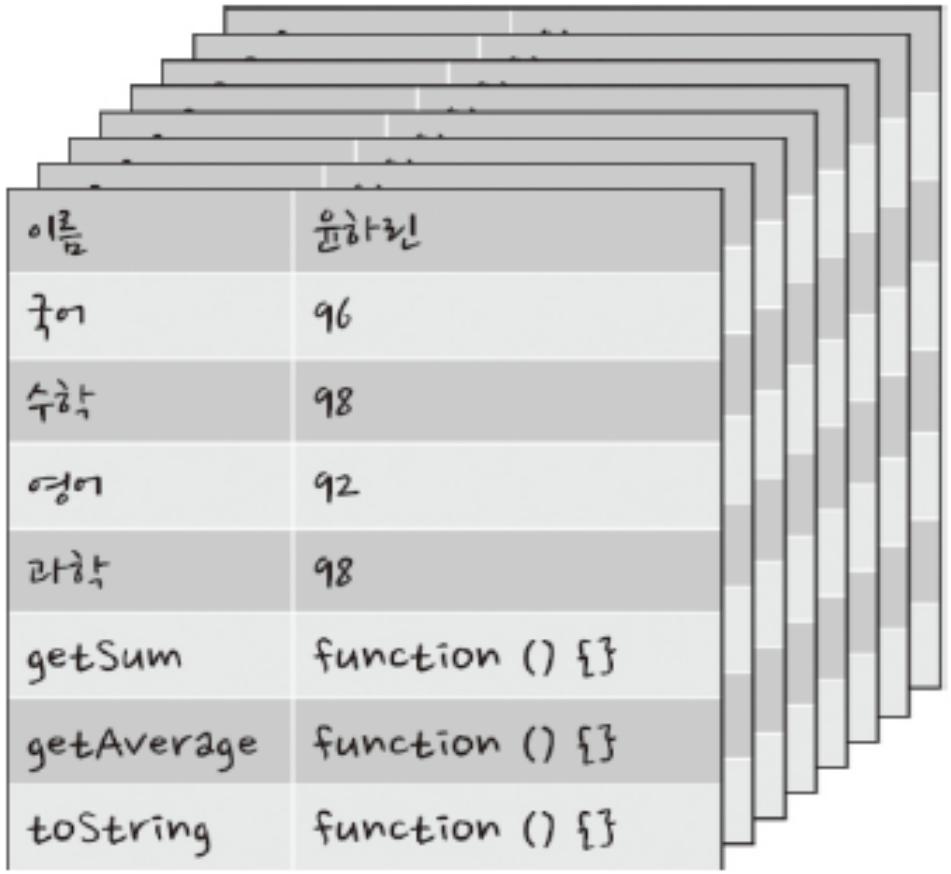
```
function Student(name, korean, math, english, science) {
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;

    this.getSum = function() {
      return this.국어 + this.수학 + this.영어 + this.과학;
    }

    this.getAverage = function() { return this.getSum() / 4;  }

    this.toString = function() {
      return this.이름 + '\t' + this.getSum() + '\t' + this.getAverage();
    }
  };
}
let student = new Student('김세진', 90, 83, 76, 89);
```

생성자 함수를 사용한 객체 배열 생성

```
let students = [];

students.push(new Student('윤인성', 90, 83, 76, 89));
students.push(new Student('박찬호', 90, 83, 76, 89));
students.push(new Student('류현진', 90, 83, 76, 89));
students.push(new Student('이세돌', 90, 83, 76, 89));
students.push(new Student('김세진', 90, 83, 76, 89));
students.push(new Student('이하나', 90, 83, 76, 89));

let output ='이름\t총점\t평균\n';
for(let i in students) {
  output += students[i].toString()+ '\n';
}

console.log(output);
```

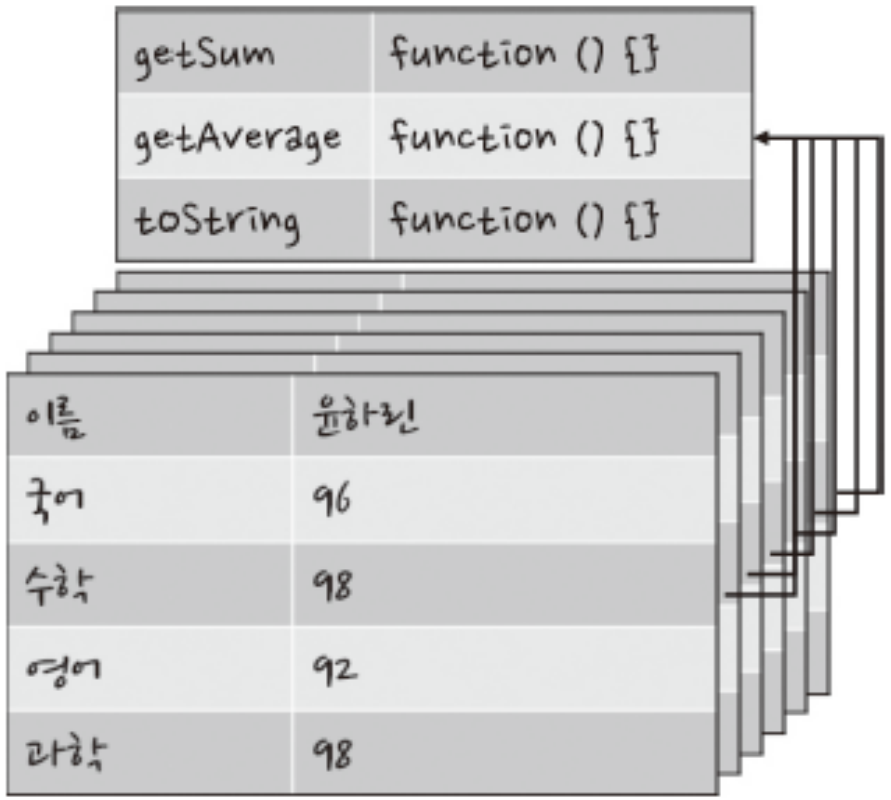
생성자 함수

- 기존의 객체 구조
  → 이름, 국어, 수학,영어, 과학 속성
  → getSum ( ), getAverage ( ),toString ( ) 메서드



메모리에 따른 문제 해결

- 프로토타입
  → 동일한 함수 생성에 따른 비효율적인 메모리 이용을 해결
  → 생성자 함수로 생성된 객체가 공통으로 가지는 공간
- 프로토타입을 사용한 객체 구조



생성자 함수 구성

- 한 개의 메서드로 모든 객체가 사용
- 생성자 함수로 객체를 만들 때 → 생성자 함수 내부에 속성만 넣음

```
function Student(name, korean, math, english, science) {
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;
}
```

프로토타입

- 자바스크립트의 모든 함수는 변수 prototype을 갖음
- prototype은 객체

```
Student.prototype.getSum = function() {
  return this.국어 + this.수학 + this.영어 + this.과학;
}

Student.prototype.getAverage = function() {
  return this.getSum() / 4;
}

Student.prototype.toString = function() {
  return this.이름 + '\t' + this.getSum() + '\t'
            + this.getAverage();
}
```

new 키워드

```
<script>
    // 생성자 함수를 선언합니다.
    function Constructor(value) {
        this.value = value;
    }

    // 변수를 선언합니다.
    var constructor = new Constructor('Hello');

    // 출력합니다.
    alert(constructor.value);
</script>
```

constructor.html

```
<body>
  <script>
    function Human(name, age) {
      this.name = name;
      this.age = age;
      this.intro =  function() {
        document.write('name = ' + this.name + '<br>');
        document.write('age = ' + this.age + '<br>');
      };
    }

    let kim = new Human('김상형', 29);
    let lee = new Human('이승우', 42);
    kim.intro();
    lee.intro();
  </script>
</body>
```

```
name = 김상형
age = 29
name = 이승우
age = 42
```

prototype.html

```
<body>
  <script>
    function Human(name, age) {
      this.name = name;
      this.age = age;
    }
    Human.prototype.intro = function() {
      document.write('name = ' + this.name + '<br>');
      document.write('age = ' + this.age + '<br>');
    };

    let kim = new Human('김상형', 29);
    let lee = new Human('이승우', 40);
    kim.intro();
    lee.intro();
  </script>
</body>
```

```
name = 김상형
age = 29
name = 이승우
age = 42
```

상속이란?

- 기존의 생성자 함수나 객체를 기반으로 새로운 생성자 함수나 객체를 쉽게 만드는 것
- 상속으로 만들어지는 객체는 기존 객체의 특성이 모두 있음
- 상속을 사용하면 이전에 만들었던 객체와 비슷한 객체를 쉽게 만들 수 있음

상속의 예

Rectangle

```
function Rectangle(w, h) {
    let width = w;
    let height = h;

    this.getWidth = function () { return width; };

    this.getHeight = function () { return height; };

    this.setWidth = function (w) {
        width = w;
    };

    this.setHeight = function (h) {
        height = h;
    };
}
```

상속의 예

- 생성자 함수 Square 내부에서 작성한 것
  → (1) base 속성에 생성자 함수 Rectangle을 넣고 실행한 것
  → (2)생성자 함수 Square의 프로토타입에 Rectangle의 프로토 타입을 넣은 것

- (1)을 사용해 Rectangle 객체의 속성을 Square 객체에 추가
- (2)를 사용해 Rectangle 객체의 프로토타입이 가진 속성 또는 메서드를 Square 객체의 프로토타입에 복사

```
Rectangle.prototype.getArea = function () {
    return this.getWidth() * this.getHeight();
};

let rectangle = new Rectangle(5, 7);

console.log('AREA: ' + rectangle.getArea());
```

상속

```
// 생성자 함수를 선언합니다.
function Square(length) {
    this.width = length;
    this.height = length;
}

Square.prototype.getArea = function () {
    return this.getWidth() * this.getHeight();
};
```

- Rectangle에 이미 구현되어 있는 코드임

상속

- Rectangle을 상속받아 Square 정의

```
function Square(length) {
    this.base = Rectangle;
    this.base(length, length);
}

Square.prototype = Rectangle.prototype;
Square.prototype.constructor = Square;

let rectangle = new Rectangle(5, 7);
let square = new Square(5);
console.log(rectangle.getArea() + ' : ' + square.getArea());

console.log(square instanceof Rectangle);
```

instanceof

- 객체가 특정 타입인지 검사

O instanceof T

- 객체 O가 T 생성자로부터 나왔으면 true
- 객체 O가 T의 prototype을 상속했는지 점검

instanceof.html

```
<body>
  <script>
    function Human(name, age) {
      this.name = name;
      this.age = age;
    }

    function Dog(type, weight, male) {
      this.type = type;
      this.weight = weight;
      this.male = male;
    }

    let kim = new Human('김상형', 29);
    if (kim instanceof Human) {
      document.write('kim은 사람입니다.<br>');
    }
```

instanceof.html

```
    let happy = new Dog('시츄', 900, false);
    if (happy instanceof Human) {
      document.write('happy는 사람입니다.<br>');
    }
  </script>
</body>
```