웹 앱 개발을 위한 JavaScript 기초 강의 노트

제 9회차

객체의 사용

한국기술교육대학교 KOREA UNIVERSITY OF TECHNOLOGY & EDUCATION

▣ 학습목표

- 객체를 생성한 후 속성을 추가/제거 할 수 있다.
- this 키워드가 참조하는 대상을 설명할 수 있다.
- 배열과 객체를 이용하여 성적관리 프로그램을 작성할 수 있다.
- 생성자 함수에 대해 이해하고 사용할 수 있다.

▣ 학습내용

- 속성의 수정
- 객체와 배열
- 생성자 함수

1. 속성의 수정

1) 속성의 추가

- 속성값 수정

- 속성값은 대괄호 표기법/마침표 표기법에 의해 접근하여 할당에 의해 수정
- 예

```
<script>
    var car = {
        name : 'car1',
        model : 400,
        color :'black'
    };

    car['name'] = 'car2';
    alert(car['name']);

    car.model = 500;
    alert(car.model);
</script>
```

```
<script>
    var car = {
        name: 'car1',
        model: 400,
        color: 'black'
    };
    ① car.coler = 'white';
    ② alert(car.color);
```

① color 이 아니라 coler을 속성값으로 할당함

② color(컬러) 속성 값 : black

- 대괄호 표기법 혹은 마침표 표기법을 이용하여 값을 할당

- 속성이 존재하면 속성값이 수정
- 속성이 존재하지 않으면 속성이 동적으로 생성된 후 값을 할당
- 메서드도 동일한 방법으로 추가

1. 속성의 수정

2) 속성의 제거

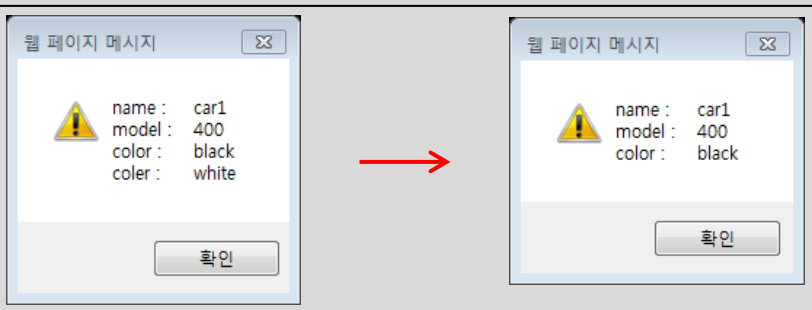
- 객체를 생성한 후 속성을 제거하는 것을 동적으로 속성을 제거한다라고 함
- delete 키워드 뒤에 제거하고자 하는 속성 입력

> • delete 키워드 뒤의 괄호( )의 사용은 선택사항
> • 객체의 속성을 삭제하는 것이며, 객체 자체를 삭제하지는 못함
> • 예) delete car.coler or delete (car.coler)

- 동일한 방법으로 메서드 제거 가능
- 동적으로 속성을 추가하는 프로그램

```
<script>
    var car = {
       name: 'car1',
       model: 400,
       color: 'black'
    };

    car.coler = 'white';
    alert(car.color);

    var out = '';
    for (var i in car) {
       out += i + ' : ₩t' + car[i]+'₩n';
    }
    alert(out);
    delete (car.color);
    var out2 = '';
    for (var j in car) {
       out2 += j + ' : ₩t' + car[j] + '₩n';
    }
    alert(out2);
</script>
```

- car의 color 속성을 제거 후 car 객체의 속성을 살펴보는 코드 추가
- delete 키워드 뒤에 삭제하려는 속성 입력

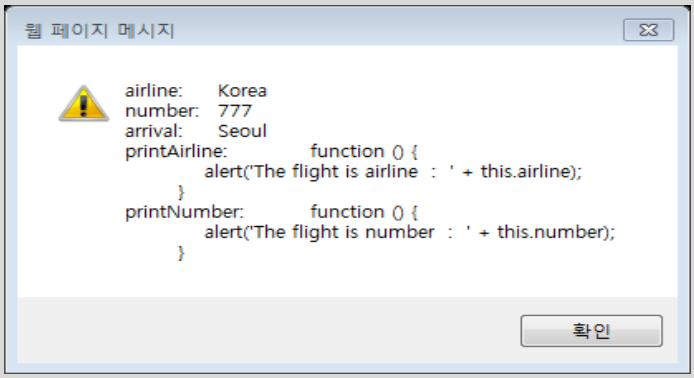


1. 속성의 수정

3) 메서드 추가/제거 예

```
<script>
    var flight = {};

    flight.airline = 'Korea';
    flight.number = 777;
    flight.arrival = 'Seoul';
    flight.printAirline = function () {
        alert('The flight is airline : ' + this.airline);
    }
    flight.printNumber = function () {
        alert('The flight is number : ' + this.number);
    }

  ① var out=''
    for (var i in flight) {
        out += i + ': ₩t' + flight[i] + '₩n';
    }
    alert(out);
  ② flight.printAirline();
  ③ flight.printNumber();

    delete (flight.printAirline);
  ④ var out2 = ''
    for (var j in flight) {
        out2 += j + ': ₩t' + flight[j] + '₩n';
    }
    alert(out2);
</script>
```

① 결과

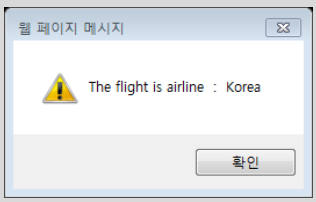


1. 속성의 수정

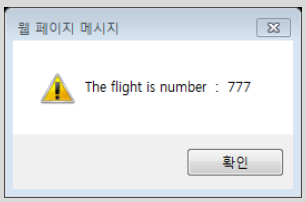
3) 메서드 추가/제거 예

```
<script>
    var flight = {};

    flight.airline = 'Korea';
    flight.number = 777;
    flight.arrival = 'Seoul';
    flight.printAirline = function () {
      alert('The flight is airline : ' + this.airline);
    }
    flight.printNumber = function () {
      alert('The flight is number : ' + this.number);
    }

  ① var out=''
    for (var i in flight) {
      out += i + ': ₩t' + flight[i] + '₩n';
    }
    alert(out);
  ② flight.printAirline();
  ③ flight.printNumber();

    delete (flight.printAirline);
  ④ var out2 = ''
    for (var j in flight) {
      out2 += j + ': ₩t' + flight[j] + '₩n';
    }
    alert(out2);
</script>
```

② 결과



③ 결과

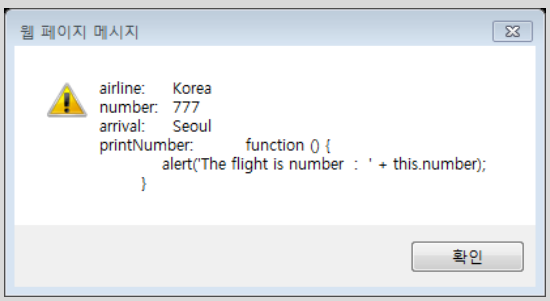


1. 속성의 수정

3) 메서드 추가/제거 예

```
<script>
    var flight = {};

    flight.airline = 'Korea';
    flight.number = 777;
    flight.arrival = 'Seoul';
    flight.printAirline = function () {
        alert('The flight is airline : ' + this.airline);
    }
    flight.printNumber = function () {
        alert('The flight is number : ' + this.number);
    }

  ① var out=''
    for (var i in flight) {
        out += i + ': ₩t' + flight[i] + '₩n';
    }
    alert(out);
  ② flight.printAirline();
  ③ flight.printNumber();

    delete (flight.printAirline);
  ④ var out2 = ''
    for (var j in flight) {
        out2 += j + ': ₩t' + flight[j] + '₩n';
    }
    alert(out2);
</script>
```

④ 결과



1. 속성의 수정

4) 객체 호출 시 this

- this

- 함수 호출 시 입력하는 매개 변수와 함께 암묵적으로 this 인자 함께 전달
- 함수가 호출되는 방식에 따라 this는 다른 객체를 참조함
- 메서드 호출 시 해당 메서드를 호출한 객체를 참조
- 예

```
<script>
    var car = {
        name: 'car1',
        model: 400,
        color: 'black',
        run : function(){
                alert(this.name + '   is running.');
            }
    };

    var student = {
        name: '홍길동'
    }

    student.run = car.run;

    car.run();
    student.run();
  </script>
```

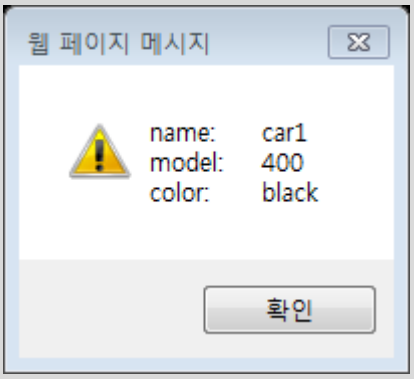
1. 속성의 수정

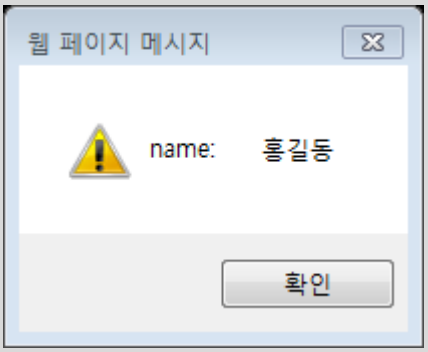
4) 객체 호출 시 this

- this

- this가 참조하는 객체의 속성을 for in문을 이용해 수정하기

```
<script>
    var car = {
        name: 'car1',
        model: 400,
        color: 'black',
        run: function () {

            var out = '';
            for (var i in this) {          → this의 속성을 i에 할당
                if (i != 'run') {          → 할당된 속성이름이 run이 아닐 경우
                    out += i + ': ₩t' + this[ i ] + '₩n';
                }                                  ↓
            }
            alert(out);          속성이름, 속성값을 변수 out에 더함
        }
    };

    var student = {
        name: '홍길동'
    }

    student.run = car.run;

    car.run();          → car객체의 run 메서드 호출
    student.run();      → student객체의 run 메서드 호출
</script>
```





2. 객체와 배열

1) 객체와 배열

- 배열

  - 하나의 변수에 여러 개의 값을 저장

- 객체

  - 데이터를 구조화하는데 용이 / 학생 개인의 정보 -> 추상화(객체)

- 학생들 성적의 총점과 평균을 구하여 출력하는 프로그램

```
<script>
    var student0 = { name: '홍길동', korean: 97, english: 93, math: 98, history: 96 };
    var student1 = { name: '김길동', korean: 87, english: 88, math: 80, history: 88 };
    var student2 = { name: '이길동', korean: 80, english: 90, math: 83, history: 93 };
    var student3 = { name: '고길동', korean: 95, english: 87, math: 90, history: 92 };
    var student4 = { name: '박길동', korean: 94, english: 94, math: 91, history: 85 };

    var students = [];
    students.push(student0);
    students.push(student1);
    students.push(student2);
    students.push(student3);
    students.push(student4);

    for (var i in students) {
       students[i].getSum =  function () {
                              return this.korean + this.english + this.math + this.history;
                               }

       students[i].getAverage = function () {
                                    return this.getSum() / 4;
                                }
    }
    var out = '';
    for (var j in students) {
       with (students[j]) {
          out += name + ' ₩t총점: ' + getSum() + '₩t 평균: ' + getAverage() + '₩n';

       }
    }
    alert(out);
</script>
```

2. 객체와 배열

2) 함수를 이용한 객체 생성

```
<script>
    function make(name, korean, english, math, history) {
      var student = {
        name: name,
        korean:korean,
        english: english,
        math: math,
        history: history,
        getSum: function () {
            return this.korean + this.english + this.math + this.history;
        },
        getAverage: function () {
            return this.getSum() / 4;
        }
      }
      return student;
    }
    var students = [];
    students.push(make('홍길동',97,93,98,96));
    students.push(make('김길동',87,88,80,88));
    students.push(make('이길동',80,90,83,93));
    students.push(make('고길동',95,87,90,92));
    students.push(make('박길동',94,94,91,85));

    var out = '';
    for (var j in students) {
      with (students[j]) {
        out += name + ' ₩t총점: ' + getSum() + '₩t 평균: ' + getAverage() + '₩n';
      }
    }
    alert(out);
  </script>
```

- 생성자 함수를 사용하면 더 많은 객체의 기능을 사용할 수 있음

3. 생성자 함수

1) 생성자 함수의 개요

- 함수

- 객체를 생성하여 객체를 돌려줌
- 객체를 생성하는 틀 역할
- new를 붙이면 새로운 객체를 생성한 후에 이를 리턴함
- 예 : var student = new Make()

```
<script>
    function Make(name, korean, english, math, history) {
        this.name = name;
        this.korean = korean;
        this.english = english;
        this.math = math;
        this.history = history;

        this.getSum = function () {
            return this.korean + this.english + this.math + this.history;
        };
        this.getAverage = function () {
            return this.getSum() / 4;
        };
    }
```

3. 생성자 함수

2) 일반 함수와 생성자 함수

- this

- 매서드 호출 시 해당 매서드를 호출한 객체 참조
- 함수가 호출되는 방식에 따라 다른 객체를 참조
- 일반 함수 호출 시 : this는 전역객체를 참조
- 생성자 함수 호출 시 : 생성될 객체를 참조
- this의 참조

```
<script>
    var object = '';
    function Func() {
        object = this;
    }

    var object1 = Func();
    if (object == window) {
        alert('일반 함수 호출 시 this는 window 객체를 참조');
    }

    var object2 = new Func();
    if (object == object2) {
        alert('생성자 함수 호출 시 this는 새로 생성될 객체를 참조');
    }
</script>
```

3) Instqanceof 키워드

- var object2 = new Func()

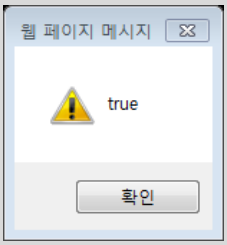
  객체, 인스턴스 생성자 함수

3. 생성자 함수

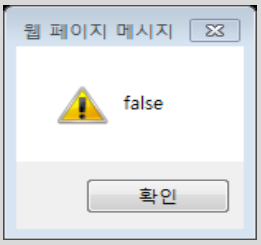
3) Instqanceof 키워드

```
<script>
    function Make(name, korean, english, math, history) {
        this.name = name;
        this.korean = korean;
        this.english = english;
        this.math = math;
        this.history = history;
    }

    function Func() {
       object = this;

    }

    var student = new Make('홍길동', 97, 93, 98, 96);
    var object2 = new Func();

 ① alert(student instanceof Make);
    alert(object2 instanceof Func);
 ② alert(object2 instanceof Make);
    alert(student instanceof Func);

  </script>
```

① 결과



② 결과



▣ 정리하기

1. 속성의 수정

- 대괄호 표기법과 마침표 표기법으로 속성에 값 할당했을 때, 속성이 존재하면 속성값이 수정이 되며 속성이 존재하지 않으면 속성이 추가됨
- delete 키워드 뒤에 제거하고자 하는 속성을 입력하여 속성을 제거할 수 있음
- 함수 호출 시 암묵적으로 전달되는 this는 메서드 호출 시, 해당 메서드를 호출한 객체를 참조함

2. 객체와 배열

- 객체를 배열에 저장하면 조작이 용이하여 간단한 성적관리 프로그램을 만들 수 있음

▣ 정리하기

3. 생성자 함수

- 생성자 함수는 객체를 생성하는 역할을 하는 함수이며, new 키워드를 붙여서 호출
- this는 일반 함수 호출 시 window 객체를 참조하며, 생성자 함수 호출 시 생성될 객체를 참조
- 생성자 함수 호출로 인해 생성된 객체를 instance라 하며, instanceof 키워드를 이용하여 생성자 함수에 의해 생성된 객체가 어떤 생성자 함수로 인해 생성되었는지 확인 가능