원시 객체
- 원시 객체 타입 -

원시 객체 타입

❖ Object

- ㅇ 자바스크립트 객체의 루트 객체

멤버	설명
constructor	객체를 만든 생성자에 대한 참조이다.
toString()	객체를 문자열로 표현한다.
toLocaleString()	지역화된 문자열을 반환한다.
valueOf()	원시값을 추출한다.
hasOwnProperty(p)	상속받지 않은 고유의 속성이 있는지 조사한다.
isPrototypeOf(p)	특정 프로토타입의 객체인지 조사한다.
propertyIsEnumerable(p)	속성이 존재하는지, for in으로 열거할 수 있는지 조사한다.

원시 객체 타입

❖ 07_1_01_newoperator.html

```
<body>
   <script>
      var a = new Object();
      var b = new Object(1234);
      var c = new Object("문자열");

     document.write("타입 = " + typeof(a) + ", 값 = " + a + "<br>");
     document.write("타입 = " + typeof(b) + ", 값 = " + b + "<br>");
     document.write("타입 = " + typeof(c) + ", 값 = " + c + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ 07_1_02_primitivestring.html

```
<body>
   <script>
      var a = 1234;
      var s = "문자열";
      var b = false;
      var human = {
         name: "김상형",
         age: 29
      };

      document.write("a  = " + a.toString() + "<br>");
      document.write("s  = " + s.toString() + "<br>");
      document.write("b  = " + b.toString() + "<br>");
      document.write("human  = " + human.toString() + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ 07_1_03_objectstring.html

```
<body>
   <script>
      var human = {
         name: "김상형",
         age: 29,
         toString : function() {
            return "name : " + this.name + ", age : " + this.age;
         }
      };

      document.write("human  = " + human.toString() + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ 07_1_03_objectstring2.html

```
<body>
   <script>
      function Human(name, age) {
         this.name = name;
         this.age = age;
      }
      Human.prototype.toString = function() {
         return "name : " + this.name + ", age : " + this.age;
      }

      var human = new Human("김상형", 29);
      document.write("human  = " + human + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ 07_1_04_valueof.html

```
<body>
   <script>
      var a = new Object(1000);
      var b = a.valueOf();
      var c = 2000 + a.valueOf();

     document.write("타입 = " + typeof(a) + ", 값 = " + a + "<br>");
     document.write("타입 = " + typeof(b) + ", 값 = " + b + "<br>");
     document.write("타입 = " + typeof(c) + ", 값 = " + c + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ Number

- ㅇ 숫자형을 표현하는 객체
- ㅇ 주요 속성

상수	설명
MAX_VALUE	표현 가능한 최대수
MIN_VALUE	표현 가능한 최소수. 0에 가장 가까운 수
POSITIVE_INFINITY	양의 무한대
NEGATIVE_INFINITY	음의 무한대
NaN	숫자가 아니라는 뜻이며 일반적으로 에러를 의미한다.

원시 객체 타입

❖ 07_1_05_numberobject.html

```
<body>
   <script>
      var a = 1234;
      var b = new Number(1234);
      var c = new Number("1234");
      var d = Number("1234");

     document.write("타입 = " + typeof(a) + ", 값 = " + a + "<br>");
     document.write("타입 = " + typeof(b) + ", 값 = " + b + "<br>");
     document.write("타입 = " + typeof(c) + ", 값 = " + c + "<br>");
     document.write("타입 = " + typeof(d) + ", 값 = " + d + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ 07_1_06_numberconst.html

```
<body>
<script>
   document.write("최대수 : " + Number.MAX_VALUE + "<br>");
   document.write("최소수 : " + Number.MIN_VALUE + "<br>");
   document.write("+무한대 : " + Number.POSITIVE_INFINITY + "<br>");
   document.write("-무한대 : " + Number.NEGATIVE_INFINITY + "<br>");
   document.write("NaN : " + Number.NaN + "<br>");
</script>
</body>
```

원시 객체 타입

❖ 주요 메서드

메서드	설명
toExponential(x)	부동 소수점 형식으로 변환한다. x는 10진수로 소수점 이하 몇자리까지 표현할 것인가를 지정한다. 0~20까지의 자리수를 지정할 수 있으며 생략시 최대한 많은 자리수를 표현한다.
toFixed(x)	고정 소수점 형식으로 변환한다. x의 의미는 위와 같되 생략시 0이 적용되어 정수부만 표현한다.
toPrecision(x)	지정한 유효숫자까지 변환한다. x는 총 자리수이며 생략시 변환하지 않고 그대로 출력한다.

원시 객체 타입

❖ 07_1_07_tofixed.html

```
<body>
<script>
   var a = 31415.9265358979;
   document.write("단순 출력 : " + a + "<br>");

   document.write("부동 소수점() : " + a.toExponential() + "<br>");
  document.write("부동 소수점(3) : " + a.toExponential(3) + "<br>");
  document.write("부동 소수점(5) : " + a.toExponential(6) + "<br>");

   document.write("고정 소수점() : " + a.toFixed() + "<br>");
   document.write("고정 소수점(3) : " + a.toFixed(3) + "<br>");
   document.write("고정 소수점(5) : " + a.toFixed(5) + "<br>");

   document.write("유효자리수() : " + a.toPrecision() + "<br>");
   document.write("유효자리수(3) : " + a.toPrecision(3) + "<br>");
   document.write("유효자리수(5) : " + a.toPrecision(5) + "<br>");
</script>
</body>
```

원시 객체 타입

❖ String 객체

```
var s = new String("문자열");
var s = "문자열";
```

- 주요 속성
  - length : 문자열의 길이

원시 객체 타입

❖ 07_1_08_stringobject.html

```
<body>
   <script>
      var a = "korea";
      var b = "한국 1234";

      document.write("a의 길이 = " + a.length + "<br>");
      document.write("b의 길이 = " + b.length + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ Boolean

```
var a = new Boolean(true);                    // true
var b = new Boolean(1);                       // true
var c = new Boolean(0);                       // false
var d = new Boolean("문자열");                 // true
var e = new Boolean(null);                    // false
```

❖ 래퍼 객체

- 자바스크립트에서 모든 것은 객체
- 원시값도 객체로 생성하면 Object 타입의 객체로 생성

원시 객체 타입

❖ 07_1_09_wrapper.html

```
<body>
<script>
   var a = new String("korea");
   var b = "korea";

   document.write("타입 = " + typeof(a) + ", 값 = " + a +
                  ", 길이 = " + a.length + "<br>");
   document.write("타입 = " + typeof(b) + ", 값 = " + b +
                  ", 길이 = " + b.length + "<br>");
   document.write("korea".length + "<br>");
</script>
</body>
```

원시 객체 타입

❖ 07_1_10_threeequal.html

```
<body>
   <script>
      var a = new String("korea");
      var b = "korea";

      document.write("== 비교 : " + (a == b) + "<br>");
      document.write("=== 비교 : " + (a === b) + "<br>");
   </script>
</body>
```

원시 객체 타입

❖ 07_1_11_constructorcompare.html

```
<body>
   <script>
      var a = new String("korea");
      var b = "korea";

      if (typeof(a) == typeof(b)) {
         document.write("타입이 같다");
      }
      if (a.constructor == b.constructor) {
         document.write("생성자가 같다");
      }
   </script>
</body>
```

원시 객체 타입

❖ 07_1_12_addproperty.html

```
<body>
   <script>
      var a = new String("korea");
      var b = "korea";

      a.capital = "서울";
      b.capital = "서울";

      document.write("a의 수도 = " + a.capital + "<br>");
      document.write("b의 수도 = " + b.capital + "<br>");
   </script>
</body>
```

원시 객체
- 속성 설명자 -

속성 설명자

❖ 속성의 추가 : ES5에서 추가된 속성

메서드	설명
create	프로토타입과 속성을 사용해 객체를 생성한다.
defineProperty	속성를 추가한다.
defineProperties	여러 개의 속성을 생성한다.
freeze	읽기 전용으로 설정한다.
isFrozen	읽기 전용인지 조사한다.
seal	속성의 추가 및 삭제를 금지한다.
isSealed	잠긴 상태인지 조사한다.
preventExtensions	새 속성 추가를 금지한다.
isExtensible	새 속성을 추가할 수 있는지 조사한다.
getOwnPropertyDescriptor	속성의 옵션을 조사한다.
getOwnPropertyNames	속성의 이름 배열을 조사한다. 열거할 수 없는 속성도 조사된다.
getPrototypeOf	프로토타입 객체를 조사한다.
keys	열거 가능한 속성의 이름 배열을 조사한다.

속성 설명자

❖ 07_2_01_adddelmember.html

```
<body>
   <script>
      var human = {
         name: "김상형",
         age: 29
      };
      human.salary = 520;

      human.salary = 650;
      for (var i in human) {
         document.write(human[i] + "<br>");
      }
   </script>
</body>
```

속성 설명자

❖ 속성 설명자

- ㅇ 객체의 속성 추가(정의)

Object.defineProperty(객체, 속성, 설명자)

- 객체 : 대상 객체 인스턴스
- 속성 : 추가할 속성 이름
- 설명자 : 속성의 특성을 규정

속성 설명자

❖ 속성 설명자

옵션	설명
value	속성의 초기값
writable	값을 변경할 수 있다.
enumerable	for in문으로 열거 가능하다.
configurable	옵션값을 중간에 변경할 수 있다.
get	값을 읽는 게터 함수
set	값을 변경하는 세터 함수

속성 설명자

❖ 07_2_02_defineproperty.html

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

      Object.defineProperty(human, "salary", {
         value : 520,
         enumerable:false,
         writable:false,
      });

      human.salary = 650;
      for (var i in human) {
         document.write(human[i] + "<br>");
      }
      document.write("--------<br>");
      document.write(human.salary + "<br>");
   </script>
</body>
```

속성 설명자

❖ 07_2_03_configurable.html

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

      Object.defineProperty(human, "salary", {
         value : 520,
         enumerable:false,
         configurable:true,
      });

      // 1차 열거
      for (var i in human) {
         document.write(human[i] + "<br>");
      }
```

속성 설명자

❖ **07_2_03_configurable.html**

```
      // 옵션 변경
      Object.defineProperty(human, "salary", {
         enumerable:true,
      });
      document.write("---------<br>");

      // 2차 열거
      for (var i in human) {
         document.write(human[i] + "<br>");
      }
   </script>
</body>
```

속성 설명자

❖ 속성 설명자

```
Object.defineProperties(객체, {
    속성1 : { 옵션객체 },
    속성2 : { 옵션객체 },
});
```

속성 설명자

❖ 07_2_04_defineproperties.html

```
<script>
   var human = {
      name: "김상형",
      age: 29
   };

   Object.defineProperties(human, {
      title : {
         value : "차장",
         enumerable:true,
      },
      salary : {
         value : 680,
         enumerable:true,
      },
   });

   for (var i in human) {
      document.write(human[i] + "<br>");
   }
</script>
```

속성 설명자

❖ 액세서

- getter/setter 설정
- 대입 연산 시 호출
- 값을 읽을 때 호출

```
Object.defineProperty(human, "salary", {
   enumerable:true,
   get : function() {
      return $salary;
   },
   set : function(value) {
      if (value > 0) {
         $salary = value;
      }
   },
});
```

속성 설명자

❖ 07_2_05_accessor.html

```
<script>
   var human = {
      name: "김상형",
      age: 29
   };

   var $salary;
   Object.defineProperty(human, "salary", {
      enumerable:true,
      get : function() {
         return $salary;
      },
      set : function(value) {
         if (value > 0) {
            $salary = value;
         }
      },
   });
```

속성 설명자

❖ **07_2_05_accessor.html**

```
   human.salary = 520;
   human.salary = -200;
   for (var i in human) {
      document.write(human[i] + "<br>");
   }
</script>
```

속성 설명자

❖ **ES5의 액세서 설정**

```
get 이름 { return 내부값; }
set 이름(인수) { 내부값=인수; }
```

속성 설명자

❖ 07_2_05_accessor2.html

```
<script>
   var human = {
      name: "김상형",
      $age: 29,
      get age() { return this.$age; },
      set age(newage) {
         if (newage > 0 && newage < 130) {
            this.$age = newage;
         }
      },
      intro: function() {
         document.write("name = " + this.name + "<br>");
         document.write("age = " + this.age + "<br>");
      }
   };

   human.age = 250;
   human.intro();
   human.age = 19;
   human.intro();
</script>
```

속성 설명자

❖ 객체 생성

- 객체 리터럴의 경우 설명자 설정 불가
  - 리터럴의 속성은 모두 읽고, 쓰기, 열거 가능
- Object.create()
  - 속성의 옵션까지 지정 가능

Object.create(원본객체, 설명자집합)

속성 설명자

❖ 07_2_06_createmethod.html

```
<body>
   <script>
      var human = Object.create({}, {
         name : {
            value : "김상형",
            enumerable:true,
         },
         age : {
            value : 29,
         },
      });

      for (var i in human) {
         document.write(human[i] + "<br>");
      }
   </script>
</body>
```

속성 설명자

❖ 07_2_07_inherit.html

```
<body>
   <script>
      var human = Object.create({}, {
         name : {
            value : "김상형",
            enumerable:true,
         },
         age : {
            value : 29,
            enumerable:true,
         },
      });
```

속성 설명자

❖ **07_2_07_inherit.html**

```
      var staff = Object.create(human, {
         salary : {
            value : 680,
            enumerable:true,
         },
         title : {
            value : "차장",
         },
      });

      for (var i in staff) {
         document.write(staff[i] + "<br>");
      }
      document.write("--------<br>");
      document.write(Object.getOwnPropertyNames(staff) + "<br>");
      document.write(Object.keys(staff) + "<br>");
   </script>
</body>
```

속성 설명자

❖ 07_2_07_inherit2.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>");
      };

      function Student(name, age, major) {
         this.parent= Human;
         this.parent(name, age);
         this.major = major;
      }
      Student.prototype = Human.prototype;
```

속성 설명자

❖ 07_2_07_inherit2.html

```
      Student.prototype.intro = function() {
         document.write("name = " + this.name + "<br>");
         document.write("age = " + this.age + "<br>");
         document.write("major = " + this.major + "<br>");
      };

      Student.prototype.study = function() {
         document.write("공자왈 맹자왈<br>");
      };

      var kim = new Human("김상형", 29);
      kim.intro();

      var lee = new Student("이승우", 42, "경영학");
      lee.intro();
      lee.study();
   </script>
</body>
```

속성 설명자

❖ 캡슐화

- ES5에서 객체의 캡슐화 및 확장/수정 금지 제어

- Object.preventExtensions()
  - 속성 확장 금지

- Object.freeze(human)
  - 속성 값 수정 불가

속성 설명자

❖ 07_2_08_capsual.html

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

      Object.preventExtensions(human);

      human.salary = 345;
      for (var i in human) {
         document.write(human[i] + "<br>");
      }
   </script>
</body>
```

속성 설명자

❖ 07_2_09_freeze.html

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

      Object.freeze(human);  // 속성 값 수정 불가

      delete human.age;
      human.name = "장동건";

      for (var i in human) {
         document.write(human[i] + "<br>");
      }
   </script>
</body>
```