

# 객체(참조타입)

- 객체
- 시간 관련 객체

# 객체

- 객체

- 자바스크립트에서는 거의 모든 것이 객체
- null과 undefined 를 제외한 모든 원시 타입도 객체로 취급
- 원시 타입은 단 하나의 값만 나타낼 수 있지만, 객체는 여러 가지 값이나 복잡한 값을 나타낼 수 있음
- 객체의 내용은 프로퍼티(property) 혹은 멤버(member)라고 부름
- 프로퍼티는 이름(key)와 값(value)로 구성이 됨
- 값에는 함수도 할당 가능

```
const car = {  
    color: 'blue',  
    'the color': 'blue'  
} //color의 타입은 string
```

## 객체 생성

```
var car = new Object();  
var car = {};
```

# 객체

- 내장 객체(Built-in Object)
  - 자바스크립트 엔진에 내장되어 있어, 필요한 경우 생성해 사용
  - 문자(string), 정규식(RegExp), 날짜(Date), 배열(Array), 수학(Math)등
- 브라우저 객체 모델(Browser Object Model)
  - 브라우저에 계층적으로 내장되어 브라우저를 제어하는 데 사용하는 객체
  - window, screen, location, history, navigation
- 문서 객체 모델(Document Object Model)
  - HTML문서 구조
  - HTML의 모든 요소들을 문서 객체로 선택하여 자유롭게 속성을 바꾸거나 CSS를 적용

# 객체

- 값 접근

- 점(.)

```
car.color → 'blue'
```

- 대괄호

```
car['color'] → 'blue'
```

```
const car = {  
    color: 'blue',  
    'the color': 'blue'  
}
```

- 중첩 객체의 값 접근

```
car.brand.name → 'Ford'
```

```
car['brand']['name'] → 'Ford'
```

```
car.brand ['name'] → 'Ford'
```

```
car['brand'].name → 'Ford'
```

```
const car = {
```

```
    brand: {  
        name: 'Ford'  
    },
```

```
    color: 'blue'  
}
```

# 객체

- 값 설정

```
car.color = 'yellow'
```

```
car['color'] = 'red'
```

```
car.model = 'Fiesta' //새로운 프로퍼티 설정
```

```
const car = {
```

```
  color: 'blue',
```

```
  'the color': 'blue'
```

```
}
```

- 프로퍼티 제거(delete 키워드)

```
delete car.brand
```

```
delete car['brand']
```

# 객체

- 유용한 유틸

- 객체에 포함된 프로퍼티 개수 세기(length)

```
const car = {  
    color: 'Blue',  
    brand: 'Ford',  
    model: 'Fiesta'  
}
```

```
Object.keys(car).length
```

- Object.keys: 검사대상 객체를 전달
- length: 검사할 객체의 속성 개수

- 변수의 타입 확인(typeof)

```
const list = {}  
const count = 2  
typeof list // "object"  
typeof count // "number"  
typeof "test" // "string"  
typeof color // "undefined"
```

```
const car = { model: 'Fiesta' }  
  
if (typeof car.color === 'undefined') {  
    // color is undefined  
}
```

typeof의 활용

# 객체

## • Object 객체

- 자바스크립트의 최상위 객체
- 자바스크립트의 객체들은 Object 객체에서 파생된 것
- 래퍼 객체(포장 오브젝트)
  - String, Number, Boolean은 원시 타입이면서 오브젝트 (래퍼 오브젝트를 가짐) → 객체처럼 동작

constructor: 생성자 프로토타입을 알려줌

```
> var a={name:"Fiesta"}  
< undefined  
_____  
> a  
< ▼ {name: "Fiesta"} ⓘ  
  name: "Fiesta"  
  ▼ __proto__:  
    ▶ constructor: f Object()  
    ▶ __defineGetter__: f __defineGetter__()  
    ▶ __defineSetter__: f __defineSetter__()  
    ▶ hasOwnProperty: f hasOwnProperty()  
    ▶ __lookupGetter__: f __LookupGetter__()  
    ▶ __lookupSetter__: f __LookupSetter__()  
    ▶ isPrototypeOf: f isPrototypeOf()  
    ▶ propertyIsEnumerable: f propertyIsEnumerable...  
    ▶ toString: f toString()  
    ▶ valueOf: f valueOf()  
    ▶ toLocaleString: f toLocaleString()  
    ▶ get __proto__: f __proto__()  
    ▶ set __proto__: f __proto__()
```

자바스크립트에는 기본적으로 사용할 수 있는 유용한 내장 객체가 존재

<https://www.w3schools.com/jsref/default.asp>

# 객체

## • Object 객체의 메서드

- 즉 JS의 모든 객체는 Object 객체가 제공하는 메서드를 모두 사용할 수 있음

메서드 이름	설명
constructor()	객체의 생성자 함수 정보
hasOwnProperty(name)	객체가 name 속성이 있는지 확인(유효성 검사)
isPrototypeOf(object)	객체가 object의 프로토타입인지 검사
propertyIsEnumerable(name)	반복문으로 열거할 수 있는지 확인
toLocaleString()	객체가 호스트 환경에 맞는 언어의 문자열로 바꿈
toString()	객체를 문자열로 바꿈
valueOf()	객체의 값을 반환

# 객체

- **Object** 객체의 메서드 예시

```
o = new Object();                                프로퍼티의 존재 여부를 테스트  
o.prop = 'exists';  
  
function changeO() { o.newprop = o.prop; delete o.prop; }  
  
o.hasOwnProperty('prop'); // returns true  
  
changeO();  
  
o.hasOwnProperty('prop'); // returns false
```

```
o = new Object();                                직접 프로퍼티와 상속된 프로퍼티의 비교  
o.prop = 'exists';  
  
o.hasOwnProperty('prop'); // returns true  
o.hasOwnProperty('toString'); // returns false  
o.hasOwnProperty('hasOwnProperty'); // returns false
```

# 객체

## • Number 객체

- 원시 타입 number를 다룰 때 유용한 프로퍼티와 메서드를 제공하는 래퍼(wrapper) 객체
- 생성 방법

```
var a = new Number(123); → Number{123} 객체를 반환  
var a = new Number('123');
```

```
var b = Number('123'); → 123 원시타입을 반환  
var b = 123; → 123 주로 문자열을 숫자열로 바꿀 때  
                  사용(생성자가 아닌 함수의 용도로)
```

```
a instanceof Number; // true 객체의 인스턴스가 아님에도 불구하고  
b instanceof Number; // false 변수 b는 Number 객체가 제공하는 프로퍼티와 메서드를  
                      사용할 수 있음(뒤에 참조)
```

```
var b = Number('123a'); → NaN
```

# 객체

- Auto Boxing

- `String`, `Number`, `Boolean`은 `new` 키워드를 사용하지 않아도 원시 타입이면서 객체처럼 동작

```
const name = "Doggo"  
const age = 7
```

```
console.log(typeof name) // string, not object  
console.log(typeof age) // number, not object
```

```
console.log(name.length) → 5  
console.log(age.toString()) → "7"
```

??원시 타입은 속성이라는 개념이 없는데??

- Auto Boxing: 원시 타입의 속성이나 메서드에 접근할 경우 원시 값(`String`, `Number`, `Boolean`)이 객체에 자동으로 감싸짐

# 객체

- **String 객체**

- 원시 타입 String을 다룰 때 유용한 프로퍼티와 메서드를 제공하는 래퍼(wrapper) 객체

charAt()
charCodeAt()
concat()
endsWith()
fromCharCode()
includes()
indexOf()
lastIndexOf()
localeCompare()
match()
repeat()
replace()

search()
slice()
split()
startsWith()
substr()
substring()
toLocaleLowerCase()
toLocaleUpperCase()
toLowerCase()
toString()
toUpperCase()
trim()
valueOf()

# 객체(시간 관련)

- **Date** 객체

- 날짜와 시간을 표시하는 객체

```
var date = new Date() // 현재 시간
```

```
var date = new Date('December 9, 1991')
```

```
var date = new Date('December 9, 1991 02:24:23')
```

```
var date = new Date(1991, 11, 9) → 11, 9는 12월 9일을 의미
```

```
var date = new Date(1991, 11, 9, 2, 24, 23)
```

```
var date = new Date(1991, 11, 9, 2, 24, 23, 1)
```

# 객체(시간 관련)

- Date 객체

- 메서드

Method	Description
getFullYear()	Get the <b>year</b> as a four digit number (yyyy)
getMonth()	Get the <b>month</b> as a number (0-11) → 0부터 시작
getDate()	Get the <b>day</b> as a number (1-31) → 1부터 시작
getHours()	Get the <b>hour</b> (0-23)
getMinutes()	Get the <b>minute</b> (0-59)
getSeconds()	Get the <b>second</b> (0-59)
getMilliseconds()	Get the <b>millisecond</b> (0-999)
getDay()	Get the weekday as a number (0-6)
getTime()	Get the time (milliseconds since January 1, 1970)

1970년 1월 1일 00:00:00 UTC와 주어진 날짜 사이의  
경과 시간 (**밀리 초**)을 나타내는 숫자

# 객체(시간 관련)

- Date 객체

- getTime()함수

```
var now = new Date();
var before = new Date('April 2, 2020');

var interval = now.getTime() - before.getTime();
interval = Math.floor(interval / (1000 * 60 * 60 * 24));
alert('Interval: ' + interval + '일');
```

→ 수학 관련 객체(필요할 때 참조, 설명은 생략)

→ alert창 띄우기

# 객체(시간 관련)

- **Date** 객체 연습

- 날짜를 입력하면 요일을 알려주는 프로그램을 작성해보자

```
var day= ['일요일', '월요일', '화요일', '수요일','목요일','금요일','토요일']
new Date("2019-04-09");
화요일
```

- 생년월일을 입력하면 만 나이를 출력하는 프로그램을 작성해 보자

```
new Date("1995-11-04");
23
```

# 객체(시간 관련)

- **Date** 객체 연습

- D-day카운터 만들기(4월이 얼마나 남았을까?)

- 다음과 같이 2020년 5월 1일 0시 0분 0초를 나타내는 객체를 date객체를 생성하고
    - 아래와 같이 D-day를 나타내는 alert창을 띄워보세요

```
var dDay = new Date("May 1, 2020 00:00:00").getTime();
```

4월은 18일 6시간 39분 13초 남았습니다.

확인

# 객체(시간 관련)

- **타임존(Time Zone)**

- 동일한 로컬 시간을 따르는 지역
- 미국이나 캐나다처럼 면적이 넓은 나라는 지역별로 타임존이 다름



# 객체(시간 관련)

## • 표준시

- GMT: Greenwich Mean Time
- UTC: GMT의 미세한 오차를 대체하기 위해 나온 표준시
- 오프셋: UTC와 특정 지역의 시간 차
  - UTC+09:00 → UTC의 기준시간보다 9시간 빠름(대한민국)
- 전 세계의 60% 국가는 DST(Daylight Saving Time) 없이 1년 내내 표준시 만을 사용, 나머지 지역은 여름 동안 1시간 빠른 DST를 채택



런던 그리치니 천문대(경도 0도에 위치)

# 객체(시간 관련)

## • 자바스크립트의 타임존 지원

- 자바스크립트는 OS에 설정된 타임존을 사용
- IANA Time Zone Database라는 타임 정보를 가진 범용 데이터베이스 사용

