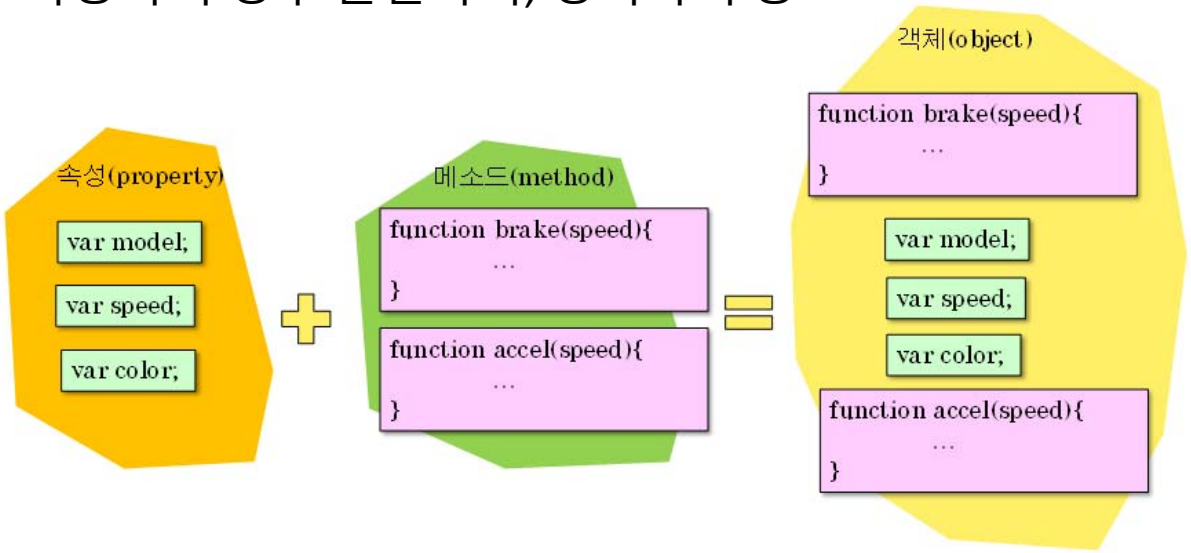
웹 프로그래밍

8. JavaScript 기초 (2)

객체

- 사물의 속성(property)과 동작(method)을 묶어서 표현하는 기법
  - Property: 이름을 갖는 값
    - 자동차의 경우 메이커, 모델, 색상, 마력, 등
  - Method: 동작을 정의하는 함수
    - 자동차의 경우 출발하기, 정지하기 등



객체

- 객체의 종류
  - *내장 객체(bulit-in object):* 생성자가 미리 작성되어 있음
  - *사용자 정의 객체(custom object):* 사용자가 생성자를 정의

- 내장 객체들은 생성자를 정의하지 않고도 사용이 가능
  - Date, String, Math, Array 등

객체 생성 방법

- 객체를 생성하는 2가지 방법
  - 객체를 객체 상수로부터 직접 생성

```
var person = {
   firstName: "John",
   lastName : "Doe",
   age: 50,
   eyeColor: "blue“,
   fullName : function() {
      return this.firstName + " " + this.lastName;
   }
};
```

객체 생성 방법

- 객체를 생성하는 2가지 방법
  - 생성자 함수를 정의한 후, new를 이용한 객체 생성

```
function Person(first, last, age, eye) {
   this.firstName = first;
   this.lastName = last;
   this.age = age;
   this.eyeColor = eye;
   this.fullName = function() {return this.firstName + " " + this.lastName;};
}

var myFather = new Person("John", "Doe", 50, "blue");
```

객체

- JS objects are mutable
  - copy되는 것이 아니라 다른 변수에 의해서 공유되어 객체 값이 변경 가능

```
<p id="demo"></p>

<script>
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}

var x = person;
x.age = 10;

document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";        // John is 10 years old.
</script>
```

Property

- Property 접근

objectName.***property*** ex) person.age
objectName[**"*property*"**] ex) person["age"]
objectName[***expression***] ex) x = "age";
person[x]

- for….in loop

```
for (variable in object) {
   code to be executed
}
```

```
var person = {fname:"John", lname:"Doe", age:25};

for (x in person) {
   txt += person[x];              // John Doe 25
}
```

method

- Method 접근

  *objectName.**methodName()*** ex) person.fullName()

- ()의 차이

```
Var name = person.fullName();          // John Doe

Var name = person.fullName;            // 함수 정의 자체 반환
                    // function () {return this.firstName + " " + this.lastName;}
```

Math property & method

속성	설명
E	오일러의 상수 (약 2.718)
LN2	자연 로그(밑수: 2) (약 0.693)
LN10	자연 로그(밑수:10) (approx. 2.302)
PI	파이 상수 (약 3.14)
SQRT1_2	1/2의 제곱근(약 0.707)
SQRT2	2의 제곱근 (약 1.414)

메소드	설명
abs(x)	절대값
acos(x), asin(x), atan(x)	아크 삼각함수
ceil(x), floor(x)	실수를 정수로 올림, 내림 함수
cos(x), sin(x), tan(x)	삼각함수
exp(x)	지수함수
log(x)	로그함수
max(x,y,z,...,n)	최대값
min(x,y,z,...,n)	최소값
pow(x,y)	지수함수 x^y
random()	0과 1 사이의 난수값 반환
round(x)	반올림
sqrt(x)	제곱근

Math

- Math.random();
  - returns a random number between 0 (inclusive), and 1 (exclusive)

```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```

Date

- 날짜와 시간 작업을 하는데 사용되는 가장 기본적인 객체
  - JS date as **string**
    - Mon Mar 19 2018 15:15:20 GMT+0900 (대한민국 표준시)
  - JS data as a **number**
    - 1521440120766
    - the number of milliseconds since January 1, 1970, 00:00:00

- Date 객체 생성
  - new Date() // 현재 날짜와 시간
  - new Date(milliseconds) //1970/01/01 이후의 밀리초
  - new Date(dateString)// 다양한 문자열
    - ISO date “2018-03-20”
    - Short date “03/20/2018”
    - Long date “Mar 20 2018” or “20 Mar 2018”
    - “2017-3-20”, “2017/03/20”, “20-03-2017” **(X)**
  - new Date(year, month, date[, hours[, minutes[, seconds[,ms]]]])

Date

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Dates</h2>
<p id="demo"></p>

<script>
var d = new Date(100000000000);
document.getElementById("demo").innerHTML = d; // Sat Mar 03 1973 18:46:40 GMT+0900 (대한민국 표준시)

var d = new Date(99,5,24,11,33,30,0);
document.getElementById("demo").innerHTML = d; // Thu Jun 24 1999 11:33:30 GMT+0900 (대한민국 표준시)

var d = new Date(99,5,24);
document.getElementById("demo").innerHTML = d; // Thu Jun 24 1999 00:00:00 GMT+0900 (대한민국 표준시)

document.getElementById("demo").innerHTML =
new Date("2015-03-25");                          // Wed Mar 25 2015 09:00:00 GMT+0900 (대한민국 표준시)

document.getElementById("demo").innerHTML =
new Date("2015-03");                             // Sun Mar 01 2015 09:00:00 GMT+0900 (대한민국 표준시)

document.getElementById("demo").innerHTML =
new Date("2015");                                // Thu Jan 01 2015 09:00:00 GMT+0900 (대한민국 표준시)
</body>
</html>
```

Date method

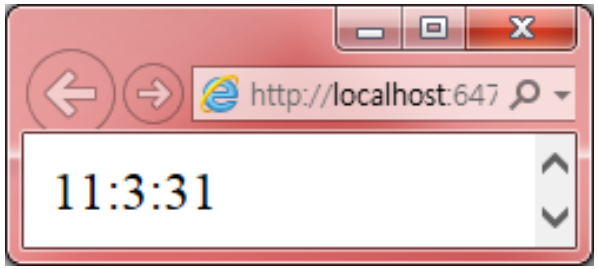
- getDate() (1-31 반환)
- getDay() (0-6 반환)
- getFullYear() (4개의 숫자로 된 연도 반환)
- getHours() (0-23 반환)
- getMilliseconds()(0-999)
- getMinutes()(0-59)
- getMonth()(0-11)
- getSeconds()(0-59)

- setDate()
- setDay()
- setFullYear()
- setHours()
- setMilliseconds()
- setMinutes()
- setMonth()
- setSeconds()

```
var d = new Date();
d.getTime();              // Get the time (milliseconds since January 1, 1970)

var d = new Date();
d.setFullYear(2020, 0, 14);
                          //Tue Jan 14 2020 15:44:12 GMT+0900 (대한민국 표준시)

var d = new Date();
d.setDate(d.getDate() + 50);        // 2017년 3월 19일 기준
                          //Tue May 08 2018 15:46:37 GMT+0900 (대한민국 표준시)
```

시계 예제

```
<div id='clock'></div>

<script>
   function setClock() {
      var now = new Date();
      var s = now.getHours() + ':' + now.getMinutes() + ':' + now.getSeconds();
      document.getElementById('clock').innerHTML = s;
      setTimeout('setClock()', 1000);                    // setTimeout(setClock, 1000);
   }
   setClock();
</script>
```



Array

- 하나의 변수 이름에 여러 데이터 값을 저장할 수 있는 자료 구조
  - var *array_name* = [*item1*, *item2*, ...];
    - 배열 원소의 자료형이 서로 달라도 됨
    - 객체, 함수, 배열 등이 배열의 원소가 될 수 있음
- 배열 생성 2가지 방법
  - 리터럴로 배열 생성

    var fruits = ["apple", "banana", "peach"];
  - Array 객체로 배열 생성

    var fruits=new Array(“apple",“banana",“orange“);
  - 두 가지 방법의 차이가 없음, 단순히 리터럴로 생성하는 것이 효율적!

Array

- Array 첨자
  - 배열의 기본 첨자는 **숫자**(numbered indexes)

```
<p id="demo"></p>
<script>
var person = ["John", "Doe", 46];
document.getElementById("demo").innerHTML = person[0];  // John
</script>
```

  - 기본적으로 연관 배열(Associative arrays) 지원 안함!
    - Named index를 지원하는 배열

```
var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
var x = person.length;        // person.length will return 0
var y = person[0];
```

Array property

- 속성
  - length : 배열의 size 반환

```
<p id="demo"></p>

<script>
        var fruits, text, fLen, i;
        fruits = ["Banana", "Orange", "Apple", "Mango"];
        fruits[6] = "Lemon";

        fLen = fruits.length;
        text = "";
        for (i = 0; i < fLen; i++) {
           text += fruits[i] + "<br>";
        }
        document.getElementById("demo").innerHTML = text;
</script>
```

Banana
Orange
Apple
Mango
undefined
undefined
Lemon

Array methods

```
var a = ["Stef", "Jason"];    // Stef, Jason
a.push("Brian");              // Stef, Jason, Brian
a.unshift("Kelly");           // Kelly, Stef, Jason, Brian
a.pop();                      // Kelly, Stef, Jason
a.shift();                    // Stef, Jason
a.sort();                     // Jason, Stef
```

- push and pop add /remove from back
- unshift and shift add/remove from front
- Shift and pop return the element that is removed

- Other methods:
  - concat, join, reverse, slice, toString,…

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias", "Linus"];
var myChildren = myGirls.concat(myBoys);        // Cecilie,Lone,Emil,Tobias,Linus
```

split and join

```
var s = "the quick brown fox";
var a = s.split(" ");              // ["the", "quick", "brown", "fox"]
a.reverse();                       // ["fox", "brown", "quick", "the"]
s = a.join("!");                   // "fox!brown!quick!the"
```

- `split` breaks apart a string into an array using a delimiter
  - can also be used with **regular expressions** (seen later)
- `join` merges an array into a single string, placing a delimiter between them

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.toString();
```

→ Banana,Orange,Apple,Mango