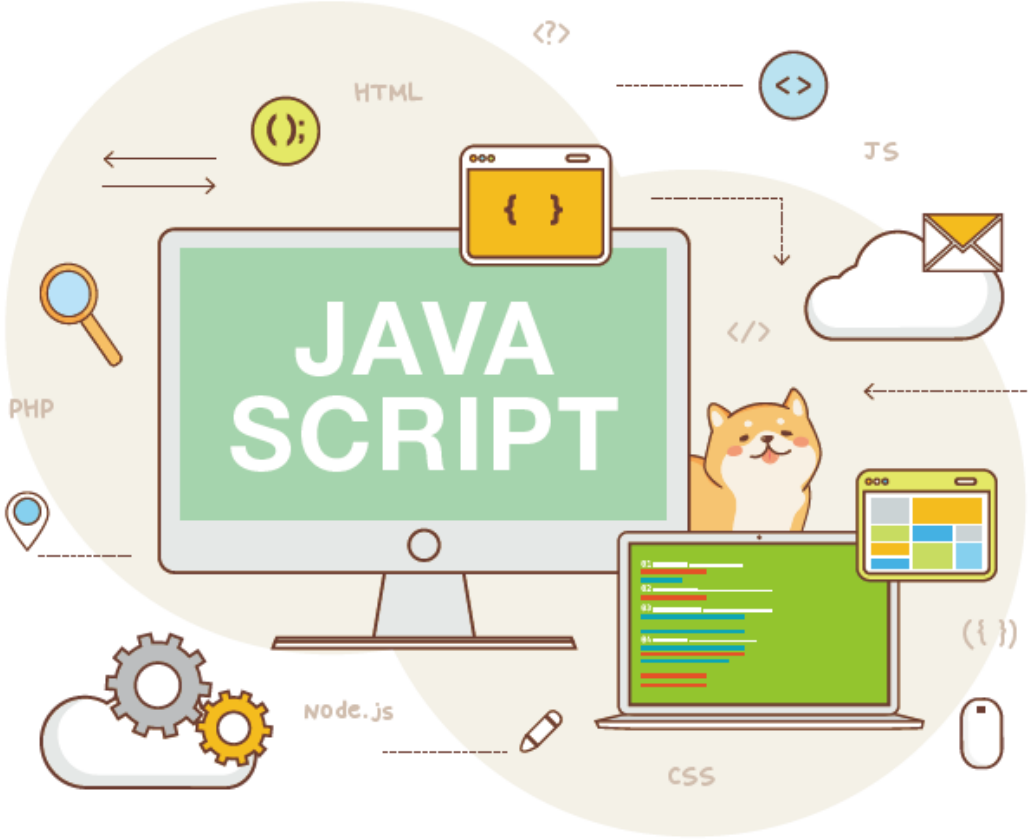
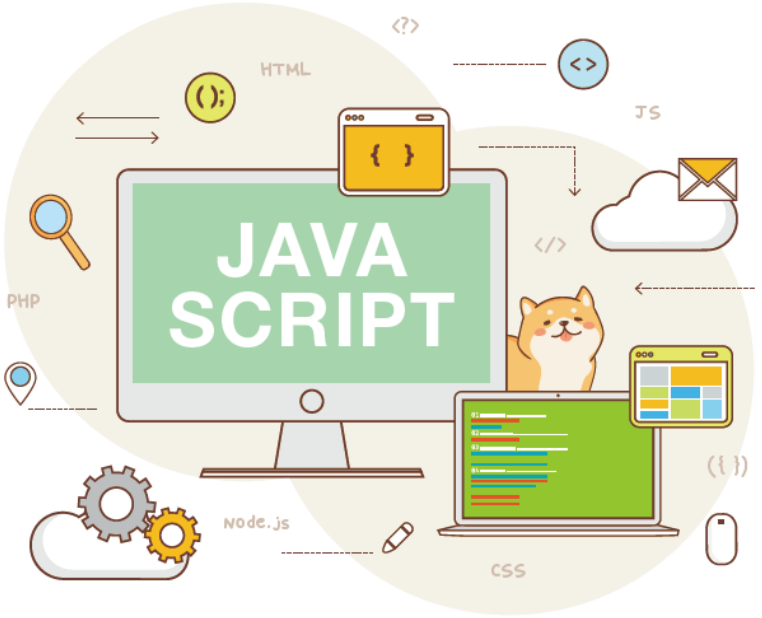


자바스크립트 심화
(Chapter 6, 7, 12)

스마트헬스 케어 – 13주차

학습 내용

1. 객체지향과 객체
2. 내장 객체
3. 브라우저 객체



이론 및 예제 실습

1_객체 지향의 주요 개념

- 클래스 (class) : 객체를 찍어 내기 위해 프로그래머가 정의하는 틀 (사용자가 정의 데이터 형식)
- 객체 (instance) : 클래스로 부터 생성된 실체. new를 통해서 생성
- 참조 변수 : 객체를 가리키는 변수. 객체가 아님
- 멤버변수 = 속성
  멤버함수 = 메소드

- 객체 지향 프로그래밍 : 클래스(틀)을 만들고 여기에서 객체(실체)를 만들어 프로그래밍 하는 방법

- 객체지향의 3대 특징
  . 캡슐화
  . 상속성
  . 다형성

1_객체 - 기본생성

- 객체 구성을 직접 정의
- 형식

  let [객체명] = { 구성 데이터 …}

  * 구성 데이터는 "키":"값" 형식으로 정의

```
let myDog = {
    color : "검정",
    name : "곰",
    age : 3,
    breed : function() {
        return this.color + " " + this.name;

    }
}

console.log(myDog);

console.log( myDog.breed() );
```

1_객체 - 틀(클래스)을 만들어 생성 (OLD 버전)

- 클래스를 정의한 후 인스턴스를 만든다
- 클래스 정의 형식

```
function [클래스명] (초기인자1, 초기인자2, …) {
  this.멤버데이터 = 초기값;
  this.멤버함수 = function(인자…) {
  };
};
```

- 멤버함수에서 멤버(속성 및 다른 멤버함수)를 접근할 때는 this로 접근한다.
- 객체(인스턴스)를 만드는 형식

```
let/var [객체명] = new [클래스명](인자1, 인자2…);
```

```
function Dog(color, name, age) {
    this.color = color;
    this.name = name;
    this.age = age;
    this.breed = function() {
        return this.color + " " + this.name;

    }
}

var myDog = new Dog("검정색", "곰", 3);

console.log(myDog);
console.log( myDog.breed() );
```

1_객체 사용

- 내부 요소 접근
  . 배열형식 접근 : 객체명['프로퍼티']
  . 객체지향형 접근 : 객체명.프로퍼티
- 프로퍼티 사용
  . 값 사용 : let name = myDog.name;
  . 값 할당 : myDog.name = "돌돌이";
  . 멤버함수 호출 : myDog.breed();
  . 프로퍼티 추가 : myDog.newProp = "새값";
  . 요소 삭제 : delete myDog.newProp;

```
let myDog = new Dog("검정색", "곰", 3);

let name = myDog.name;
console.log(name);
myDog.name = "돌돌이";
console.log(myDog.name);

let str = myDog.breed();
console.log(str);

myDog.newProp = "무서움";
console.log(myDog);

delete myDog.newProp;
console.log(myDog);
```

1_객체 사용

- 프로퍼티 순회

```
for (key in obj) {
    console.log(obj[key]);
}
```

```
let myDog = new Dog("검정색", "곰", 3);

for (key in myDog) {
    console.log(typeof myDog [key]);
    console.log(myDog[key]);
}
```

1_객체 - 틀(클래스)을 만들어 생성 2 (ES6 지원)

- 클래스 정의 형식

```
class [클래스명] {
   멤버데이터
   constructor (초기인자1, 초기인자2, …){
      this.멤버데이터 = 초기인자1;
   }
   멤버함수 (인자…) { … }
};
```

- 멤버함수에서 멤버데이터(속성)을 접근할 때는 this로 접근한다.
- 객체(인스턴스)를 만드는 형식 : let/var [객체명] = new [클래스명](인자1, 인자2…);

```
class Dog {
  constructor(color, name, age) {
    this.color = color;
    this.name = name;
    this.age = age;
  }
  breed() {
    return this.color + " " + this.name;
  }
}

var myDog = new Dog("검정색", "곰", 3);
console.log(myDog);
console.log( myDog.breed() );
```

2_내장 객체 - Math

-수학에서 자주 사용하는 상수와 함수들을 미리 구현해 놓은 자바스크립트 표준 내장 객체

- 가장 많이 사용되는 대표적인 Math 메소드

1. Math.min() : 인수로 전달받은 값 중에서 가장 작은 수를 반환합니다
2. Math.max() : 인수로 전달받은 값 중에서 가장 큰 수를 반환
3. Math.random() : 0보다 크거나 같고 1보다 작은 무작위 숫자(random number)를 반환
4. Math.round() : 반올림 연산 (인수로 전달받은 값을 소수점 첫 번째 자리에서 반올림)
5. Math.floor() : 버림 연산 (인수로 전달받은 값과 같거나 작은 수 중에서 가장 큰 정수를 반환합니다)
6. Math.ceil() : 올림 연산 (인수로 전달받은 값과 같거나 큰 수 중에서 가장 작은 정수를 반환)

```
Math.min(1, 10, -100, -10, 1000, 0);     // -100
Math.max(1, 10, -100, -10, 100, 0);      // 100

Math.random();                           // 0~1
Math.random() * (100 - 1) + 1;           // 1~100

Math.round(10.49);  // 10
Math.floor(10.95);  // 10
Math.ceil(10.95);  // 11
```

2_내장 객체 - Date

- Date 객체를 사용하여 매 순간 변화하는 시간과 날짜에 관한 정보 제공
- Date 객체는 연월일, 시분초의 정보와 함께 밀리초(millisecond)의 정보도 함께 제공

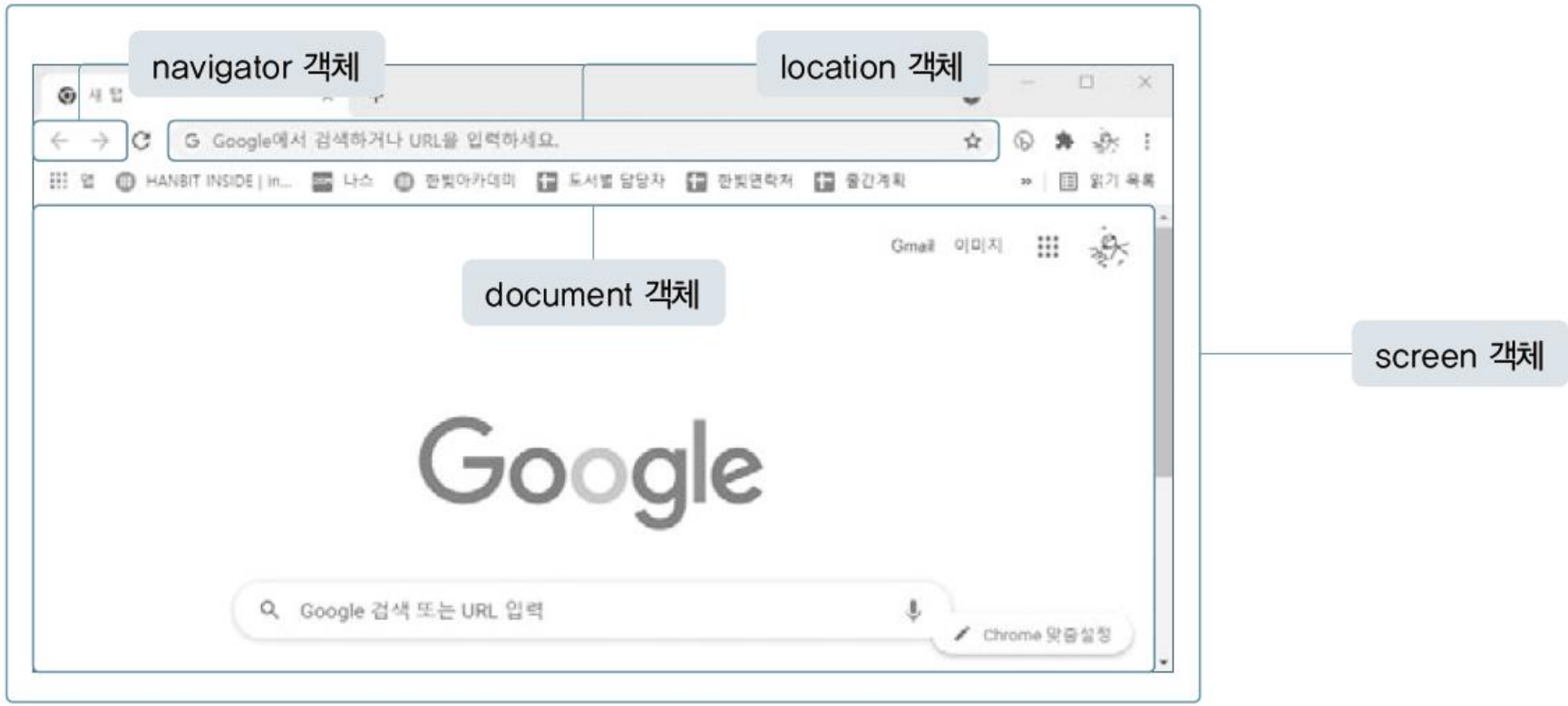
1. 연도(year) : 1900년(00) ~ 1999년(99)
2. 월(month) : 1월(0) ~ 12월(11)
3. 일(day) : 1일(1) ~ 31일(31)
4. 시(hours) : 0시(0) ~ 23시(23)
5. 분(minutes) : 0분(0) ~ 59분(59)
6. 초(seconds) : 0초(0) ~ 59초(59)

클래스	설명
new Date()	현재 시간으로 Date 객체를 생성합니다.
new Date(유닉스_타임)	유닉스 타임(1970년 1월 1일 00시 00분 00초부터 경과한 밀리초)으로 Date 객체를 생성합니다.
new Date(시간_문자열)	문자열로 Date 객체를 생성합니다.
new Date(연, 월 - 1, 일, 시간, 분, 초, 밀리초)	시간 요소(연, 월 – 1, 일, 시간, 분, 초, 밀리초)를 기반으로 Date 객체를 생성합니다.

```
var date = new Date();                    // 현재 날짜 시간에 대한 Date 객체 생성
new Date("December 14, 1977 13:30:00");   // 지정한 날짜에 대한 Date 객체 생성
new Date("1977-12-14T13:30:00");           // 날짜와 시간까지 표현함.
new Date(80000000);                       // 1970년 1월 1일 0시부터 80000000 밀리초만큼 지난 날짜
```

3_브라우저 객체 모델

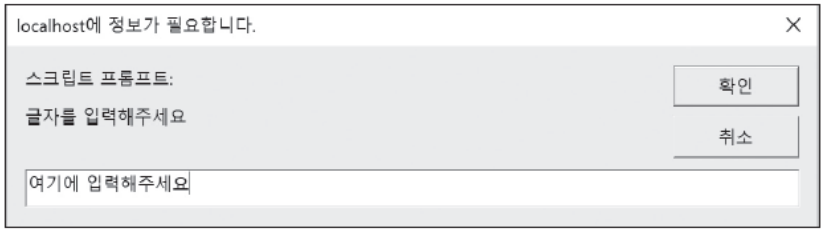
- 브라우저 객체 모델(BOM, Browser Object Model)
  : 자바스크립트로 브라우저의 정보에 접근하거나 브라우저의 여러 기능들을 제어할 수 있도록 JS객체로 제공하는 것
- 이 모델은 자바스크립트가 브라우저의 기능적인 요소들을 직접 제어하고 관리할 방법을 제공
- 브라우저 객체 모델(BOM)은 문서 객체 모델(DOM)과는 달리 W3C의 표준 객체 모델은 아님(브라우저 마다 차이)
- BOM 모델의 객체들을 전역 객체(global object)로 사용



3_브라우저 객체 - Window

- 웹 페이지 자체를 나타냄
- 새로운 화면을 열거나 웹 브라우저의 크기를 변경하는 등의 일을 하는 데 활용 되며 대표적으로 경고 출력을 하는 경고창과 입력을 하는 프롬프트를 제공함

함수	설명
alert(메시지)	경고창을 출력합니다.
prompt(메시지, 임시_글자)	프롬프트를 출력합니다.




```
alert("입력했어요");
let value = prompt("글자를 입력해 주세요" , 0);

var windowWidth = window.innerWidth || document.documentElement.clientWidth ||
document.body.clientWidth;
var windowHeight = window.innerHeight || document.documentElement.clientHeight ||
document.body.clientHeight;

document.write("웹 브라우저의 너비는 " + windowWidth + "픽셀이고, 높이는 " + windowHeight + "픽셀
입니다.");
```

3_브라우저 객체 - Location

- 현재 브라우저에 표시된 HTML 문서의 주소를 얻거나, 브라우저에 새 문서를 불러올 때 사용
- Window 객체의 location 프로퍼티와 Document 객체의 location 프로퍼티에 같이 연결되어 있다
- location 객체의 프로퍼티와 메소드를 이용하면, 현재 문서의 URL 주소를 다양하게 해석하여 처리할 수 있다

표 12-3 location 객체의 속성

속성	설명	예
href	문서의 URL 주소	
host	호스트 이름과 포트 번호	localhost:52273
hostname	호스트 이름	localhost
port	포트 번호	52273
pathname	디렉터리 경로	/folder/HTMLPage.html
hash	앵커 이름(#~)	#test
search	요청 매개 변수	?param=10
protocol	프로토콜 종류	http:

표 12-4 location 객체의 메소드

메소드	설명
assign(링크)	매개 변수로 전달한 위치로 이동합니다.
reload()	새로고침 합니다.
replace()	매개 변수로 전달한 위치로 이동합니다(뒤로 가기 불가능).

```
document.write("현재 문서의 주소는 " + location.href + "입니다.");
document.write("현재 문서의 파일 경로명은 " + location.pathname + "입니다.");

location.href = "https://www.daum.net/";
location.assign("https://www.daum.net/");
```

3_브라우저 객체 - Navigator

- 브라우저 공급자 및 버전 정보 등을 포함한 브라우저에 대한 다양한 정보를 저장하는 객체
- 사용자의 웹 브라우저, 운영체제를 구분할 수 있음
- 브라우저 스니핑(browser sniffing) : 과거에는 방문자의 웹 브라우저의 종류를 미리 파악하여 조치함으로써, 브라우저 간의 호환성을 유지

표 12-6 navigator 객체의 속성

속성	설명
appCodeName	브라우저의 코드 이름
appName	브라우저의 이름
appVersion	브라우저의 버전
platform	사용 중인 운영체제의 시스템 환경
userAgent	브라우저의 전체적인 정보

```
document.write("현재 사용 중인 브라우저의 이름은 " + navigator.appName + "입니다.<br>");
document.write("또한, 해당 브라우저의 코드명은 " + navigator.appCodeName + "입니다.");
document.write("userAgent 프로퍼티로 알 수 있는 추가 정보는 " + navigator.userAgent + "입니다.");
document.write("현재 브라우저가 실행되고 있는 운영체제는 " + navigator.platform + "입니다.");
```

응용 예제

실습-1

실습 문제 : 웹 브라우저의 너비와 높이, 현재 운영체제, 현재 문서의 주소 정보를 받아서 browserObj 객체를 만들어 아래 형식 처럼 입력한 후 객체의 각 요소를 출력하시오. (for – in)

```
{
   "width" : 1980,
   "height" : 1200,
   "OSName" : "Win32",
   "docAddress : " http://127.0.0.1:9099/10week/prog1.html"
}
```

실습-1

실습 문제 : 웹 브라우저의 너비와 높이, 현재 운영체제, 현재 문서의 주소 정보를 받아서 browserObj 객체를 만들어 아래 형식 처럼 입력한 후 객체의 각 요소를 출력하시오. (for – in)

```
let browserObj = {};

browserObj.width = window.innerWidth;
browserObj.height = window.innerHeight;

browserObj.OSName = navigator.platform;
browserObj.docAddress = location.href;


for (let key in browserObj) {
  console.log(key, " = ", browserObj[key]);
}
```

팀 프로젝트