INTERACTION PROGRAMMING 1

인터랙션 프로그래밍 1

5Week.

2018. 4. 5.

JavaScript Review

논리연산자

```
var a = 0;
var b = 1;
var c = 2;
var d = 2;

// && - and
if(a !== b && c === d){
    console.log('두 조건 중 모두 일치합니다.');
}
```

논리연산자

```
var a = 0;
var b = 1;
var c = 2;
var d = 2;

// || - or
if(a === b || c === d){
    console.log('두 조건 중 하나는 일치합니다.');
}
```

Switch 조건문

```
var greeting = 0;
switch (greeting){
    case 0 :
        console.log('Hello');
        break;
    case 1 :
        console.log('World');
        break;
    case 2 :
        console.log(':^)');
        break;
    default :
        console.log('X(');
        break;
}
```

While 반복문

```
var i = 0;
while(i < 10){
    console.log(i);
    i++
}
```

Array 객체

한 번에 두가지 이상의 값을 포함할 수 있는 객체

사용빈도가 아주 높다.

Array 배열

```
var a = 10;
var b = 'apple';
var c = null;
var d = a;
var _array = [a, b, c, d];
console.log(_array[3]);
```

```
function getMembers(){
    return ['rh', 'june', 'mind'];
}
var members = getMembers();
console.log(members[0]);
console.log(members[1]);
console.log(members[2]);
```

배열의 추가 / 제거 / 정렬

unshift(); push(); shift(); pop();

concat(); splice(); sort(); reverse();

```
var _heros = ['Iron Man', 'Hulk', 'Thor', 'Doctor Strange'];
_heros.unshift('Captain America');
_heros.push('Spider-Man');
_heros.concat(['Black Panther', 'Ant-Man']);
_heros.splice(2, 0, 'Vision');
_heros.splice(2, 1, 'Loki');
_heros.shift();
_heros.pop();
```

```
_array.splice(start, deleteCount, string[]);
```

Array 배열의 제어

```
var _heros = ['Iron Man', 'Hulk', 'Thor', 'Doctor Strange'];
_heros.sort();
_heros.reverse();
```

배열, 객체의 반복문

for ... in

```
function getMembers(){
    return ['rh', 'june', 'mind'];
}
var members = getMembers();
for(var i = 0; i < members.length; i++){
    console.log(members[i]);
}
```

Array 배열의 반복

```
var _heros = ['Iron Man', 'Hulk', 'Thor', 'Doctor Strange'];
for(var name in _heros){
    console.log(name);
}
```

```
var _person = {
    name    : '김용원',
    job     : '교수',
    phone   : '010-9137-8688',
    email   : 'rh@102labs.com'
};
for(var key in _person){
    console.log(key + ' : ' + _person[key]);
}
```

Math 수학

```
Math.abs(-100 + 50);

Math.max(0, 10);
Math.min(0, -10);

Math.max.apply(null, [0, 2, 3, 4, 5]);
Math.min.apply(null, [-5, -4, -3, -2, -1, 0]);
```

JavaScript Quest

Quest 1.

1. var _cars = ['Tesla', 'Audi', 'Volvo', 'Benz'];

2. _cars 배열의 문자열 원소를 아래 결과와 같은 **문자열**로 **console 에 출력**되도록 작성합니다.

   -> Benz, Volvo, Audi, Tesla.

Quest 2.

1. 변수를 선언하고 **숫자 원소들로만 이뤄진 배열**을 할당합니다.

   ex) var _numbers = [-1, 2, 5, 10, 1, -10, 8, 4];

2. 1(순서) 에서 선언한 변수의 배열 원소 중 **가장 큰 숫자**를 **console 에 출력**되도록 작성합니다.

3. 1(순서) 에서 선언한 변수의 배열 원소 중 **가장 작은 숫자**를 **console 에 출력**되도록 작성합니다.

Quest 3.

1. searchIndex 라는 함수를 선언하고, 2개의 **매개변수**(첫번째는 배열, 두번째는 숫자) 를 지정합니다.

2. 1(순서) 함수에 전달된 첫번째 매개변수 배열의 원소들과 두번째 매개변수의 숫자를 비교하여(반복문, 조건문 사용),
   일치할 경우 배열의 **index(원소 순서)** 를 **console 에 출력**하도록 작성합니다.

3. 1(순서) 함수에 **각 매개변수를 지정**하여 **호출**합니다.
   => ex) searchIndex([8, 10, 13, 30, 50], 30); 호출할 경우 3 이 출력됩니다.

Quest 4.

1. checkType 이라는 함수를 선언하고, 1개의 **매개변수**(배열)를 지정합니다.

2. 1(순서) 함수에 전달된 매개변수 배열의 원소들의 **데이터 타입을 원소**로 가지는 **새로운 배열을 생성**하여(반복문 사용) **console 에 출력**합니다.

3. 1(순서) 함수에 **매개변수를 지정**하여 **호출**합니다.
   => ex) checkType([10, 'Hello', 'World', {name : 'rh'}, [10, 20]]);
   호출할 경우 ['number', 'string', 'string', 'object', 'object'] 가 출력됩니다.

BOM

BOM (Browser Object Model)

- window
- navigator
- screen
- history
- location

window

window

BOM(DOM Level 0) 의 최상위 객체

브라우저 객체의 최상위 객체

브라우저 환경의 모든 객체 속성을 담고 있다.

document

BOM(DOM Level 0) 의 최상위 객체

브라우저 객체의 최상위 객체

브라우저 환경의 모든 객체 속성을 담고 있다.

브라우저 문서(html)를 모두 읽어들이면 해당 문서를 객체화하여 **document** 객체가 된다.

html 문서의 root 객체로 모든 node 를 갖게 된다.

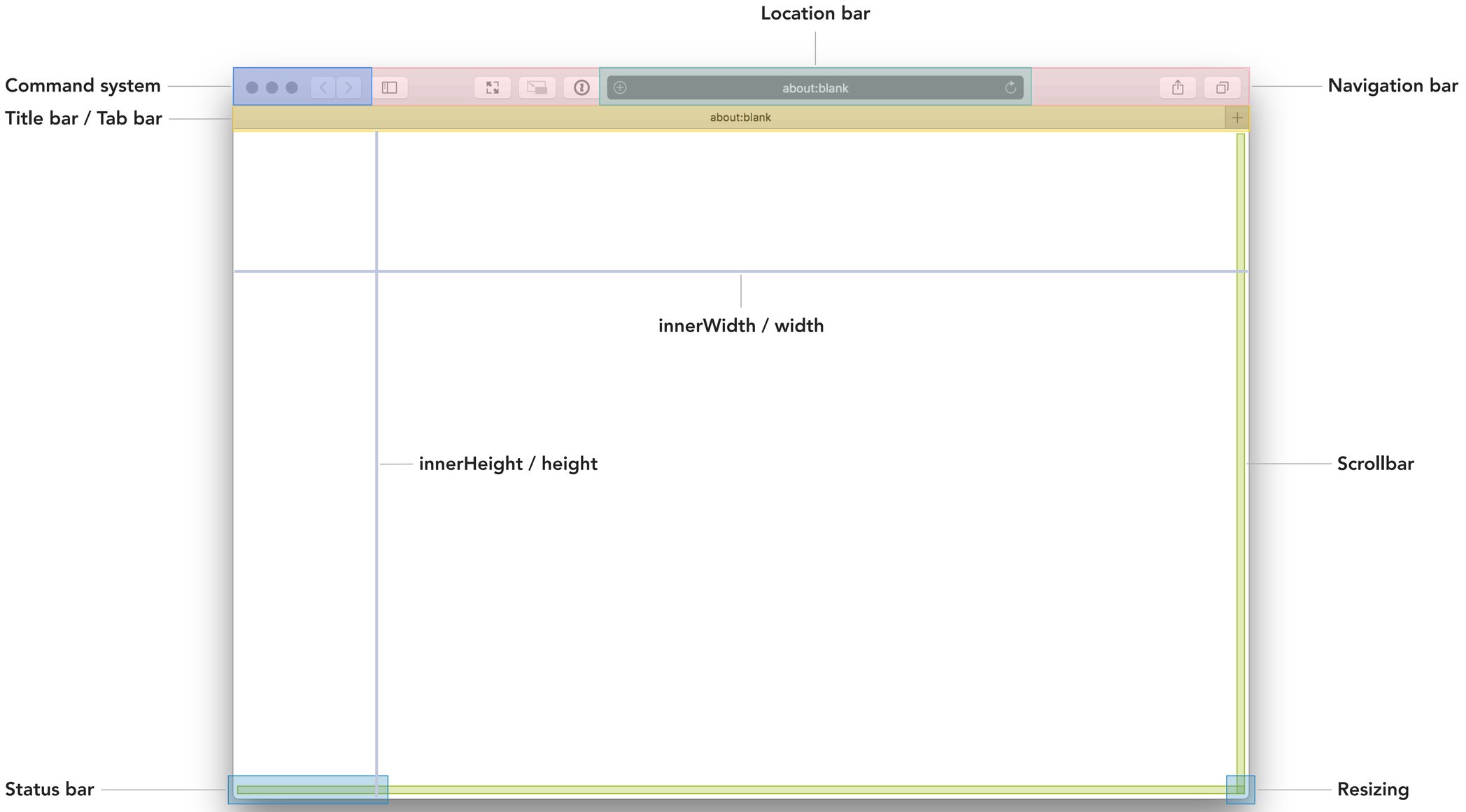
window.innerWidth

윈도우 콘텐츠 영역의 넓이 값(pixel)을 반환한다.

window.innerHeight

윈도우 콘텐츠 영역의 높이 값(pixel)을 반환한다.

window


Location bar
Command system
Navigation bar
Title bar / Tab bar
about:blank
about:blank
innerWidth / width
innerHeight / height
Scrollbar
Status bar
Resizing

window.screenTop

모니터상에서 종(수직) 좌표 값(pixel)을 반환한다.

window.screenLeft

모니터상에서 횡(수평) 좌표 값(pixel)을 반환한다.

window.pageXOffset

문서의 횡(수평) 스크롤 좌표 값(pixel)을 반환한다.

window.pageYOffset

모니터상에서 종(수직) 좌표 값(pixel)을 반환한다.

window.location

현재 웹페이지의 URL 을 반환한다.

window.alert();

window.alert('message');

OK 버튼을 포함하고 있는 경고창을 출력한다.

window.prompt();

window.prompt('message');

메세지와 사용자가 입력 가능한 프롬프트를 포함한 다이얼로그를 출력한다.

window.confirm();

window.confirm('message');

메세지와 '확인', '취소' 버튼을 포함하고 있는 다이얼로그를 출력한다.

window.scrollTo(x, y);

window.scrollTo(0, 0);

문서를 입력한 위치(x, y) 좌표로 스크롤한다.

window.setInterval();

window.setInterval(callback, time);

주기적으로 실행되는 타이머를 생성한다. (ms - millseconds 단위)

window.clearInterval();

window.clearInterval(timer);

setInterval 로 생성된 타이머를 초기화한다.

window.setTimeout();

window.setTimeout(callback, time);

한번만 실행되는 타이머를 생성한다. (ms - millseconds 단위)

window.clearTimeout();

window.clearInterval(timer);

setTimeout 으로 생성된 타이머를 초기화한다.

window.open();

window.**open(URL, name, specs, replace);**

새로운 브라우저 window 를 연다.

URL	새 window 의 URL
name	window 의 이름 (_blank, _parent, _self, _top, name)
specs	선택적인 매개변수
replace	현재 방문이력의 목록에서 대체 여부

window.open();

specs

top=pixels left=pixels width=pixels height=pixels

titlebar=yes|no|1|0 menubar=yes|no|1|0 scrollbars=yes|no|1|0 toolbar=yes|no|1|0

location=yes|no|1|0 status=yes|no|1|0 resizable=yes|no|1|0 fullscreen=yes|no|1|0

channelmode=yes|no|1|0 directories=yes|no|1|0

window.close();

window.close();

target.close();

새로 열린 window 를 닫는다.

navigator

window.navigator

브라우저의 정보를 담고 있는 객체

window.navigator.appName

브라우저의 이름을 반환한다.

window.navigator.appVersion

브라우저의 버전 정보를 반환한다.

window.navigator.userAgent

window.navigator.**userAgent**

target.navigator.**userAgent**

브라우저의 기본정보를 담고 있는 navigator 의 프로퍼티

해당 내용을 바탕으로 현재 문서가 열려있는 브라우저의 종류 및 버전, 사용중인 OS 정보까지 확인이 가능하다.

screen

window.screen

브라우저 화면의 정보를 담고 있는 객체

window.screen.availWidth

화면의 넓이를 반환한다. (작업표시줄 등을 제외한 높이)

window.screen.availHeight

화면의 높이를 반환한다. (작업표시줄 등을 제외한 높이)

window.screen.width

화면의 넓이를 반환한다.

window.screen.height

화면의 높이를 반환한다.

history

window.history

사용자가 방문한 URL 등의 이력 정보를 담고 있는 객체

window.history.length

방문이력의 수를 반환한다.

window.history.back();

이전 방문이력을 로드한다.

window.history.forward();

다음 방문이력을 로드한다.

window.history.go();

window.history.go(index);

방문이력 중 index 에 위치한 이력의 URL 로 이동한다.