



# INTERACTION PROGRAMMING 1

인터랙션 프로그래밍 1

---

*7 Week.*

*2018. 4. 19.*

# BOM Review

# BOM (Browser Object Model)

window

navigator

screen

history

location

**window**

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

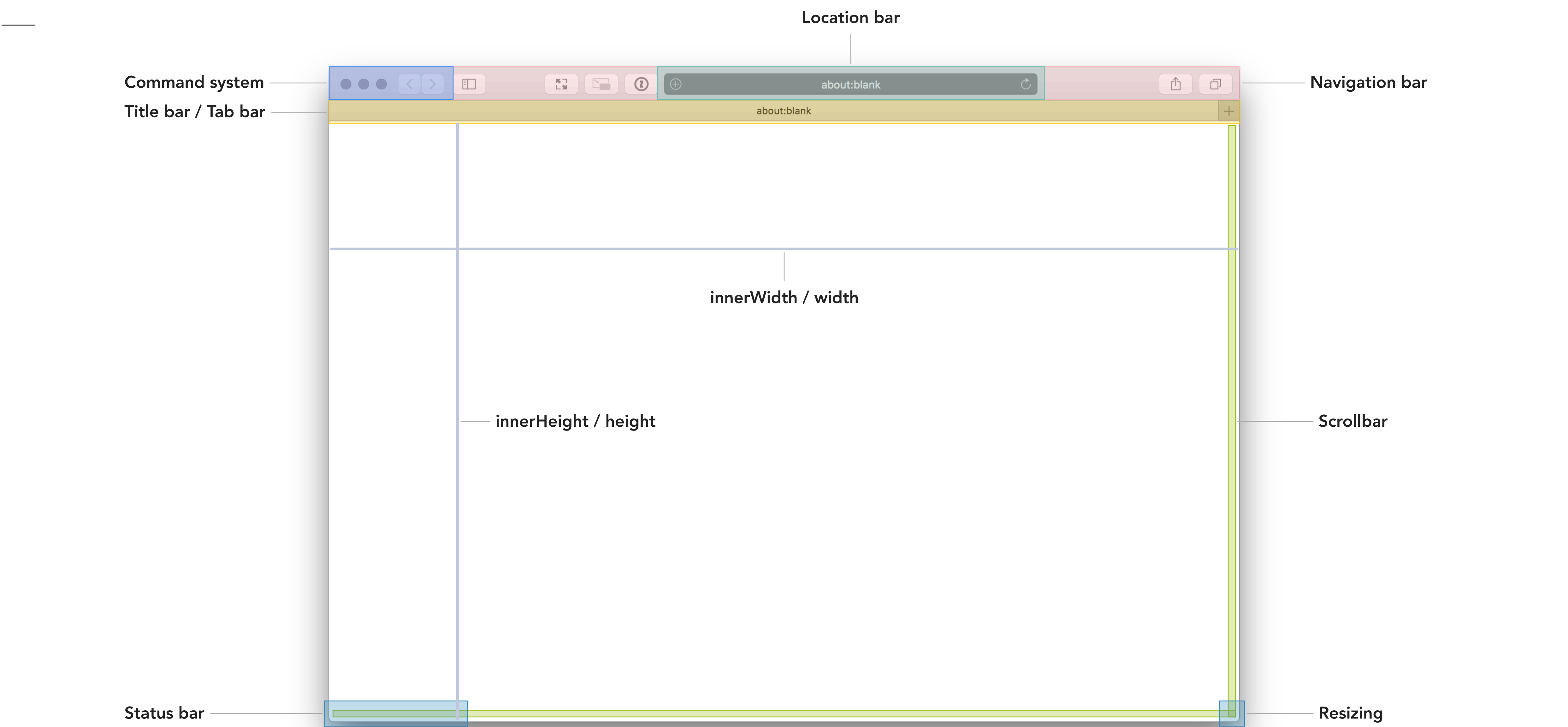
윈도우 콘텐츠 영역의 넓이 값(pixel)을 반환한다.

## window.innerHeight

윈도우 콘텐츠 영역의 높이 값(pixel)을 반환한다.



window



## 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

---

# 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 - milliseconds 단위)

## window.clearInterval();

window.clearInterval(timer);

setInterval 로 생성된 타이머를 초기화한다.

## window.setTimeout();

window.setTimeout(callback, time);

한번만 실행되는 타이머를 생성한다. (ms - milliseconds 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

**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

---

**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**



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)

screen

---

**window.screen.width**

화면의 넓이를 반환한다.

**window.screen.height**

화면의 높이를 반환한다.

**history**

# window.history

사용자가 방문한 URL 등의 이력 정보를 담고 있는 객체

history

---

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.





**BOM**

**location**

location

---

# window.location

현재 URL 에 대한 정보를 담고 있는 객체

location

---

**window.location.hash**

URL 에 포함된 hash 값을 반환한다. (#)

**window.location.host**

URL 의 호스트네임과 포트를 반환한다. (80 포트는 생략됨)

location

---

`window.location.hash`

URL 에 포함된 hash 값을 반환한다. (#)

`http://host.com/path?key=value#hash`

location

---

`window.location.host`

URL 의 호스트네임과 포트를 반환한다. (80 포트는 생략됨)

`http://host.com:port/path?key=value#hash`

**window.location.hostname**

URL 의 호스트네임을 반환한다.

**window.location.href**

URL 전체를 반환합니다. 값을 할당하는 경우 해당 경로로 이동



location

---

`window.location.hostname`

URL 의 호스트네임을 반환한다.

`http://host.com/path?key=value#hash`

## window.location.href

URL 전체를 반환합니다. 값을 할당하는 경우 해당 경로로 이동

`http://host.com/path?key=value#hash`

`window.location.href = http://host.com;`

location

---

**window.location.origin**

프로토콜을 포함한 호스트네임과 포트를 반환한다.

**window.location.pathname**

URL 중 path 를 반환합니다.

location

---

`window.location.origin`

프로토콜을 포함한 호스트네임과 포트를 반환한다.

`http://host.com/path?key=value#hash`

location

---

`window.location.pathname`

URL 중 path 를 반환합니다.

`http://host.com/path?key=value#hash`

location

---

window.location.port

URL 중 포트를 반환합니다.

window.location.protocol

URL 의 프로토콜을 반환합니다.

location

---

window.location.port

URL 중 포트를 반환합니다.

http://host.com:port/path?key=value#hash

**window.location.protocol**

URL 의 프로토콜을 반환합니다.

http://host.com/path?key=value#hash

https://host.com/path?key=value#hash



location

---

window.location.search

URL 의 Query 문자열을 반환한다.

location

---

**window.location**.search

URL 의 Query 문자열을 반환한다.

http://host.com/path?key1=value&key2=value#hash

`window.location.assign();`

`window.location.assign(url);`

새로운 문서를 로드한다. (방문이력유지)

`window.location.reload();`

현재 문서를 새로 고침한다.

location

---

`window.location.replace();`

`window.location.replace(url);`

새로운 문서를 로드한다. (방문이력대체)



**DOM**

# DOM (Document Object Model)

DOM 은 문서의 구조화된 표현을 제공하며 프로그래밍 언어가  
DOM 구조에 접근할 수 있는 방법을 제공하여 문서 구조, 스타일, 내용 등을 변경할 수 있게 돕는다.

DOM 은 구조화된 nodes 와 property 와 method 를 갖고 있는 objects 로 문서를 표현한다.

웹 페이지를 스크립트 또는 프로그래밍 언어로 사용될 수 있게 연결시켜 주는 역할을 담당한다.

**DOM === window.document**



## document.documentElement

document 객체의 자신의 element 노드를 반환한다.

## document.body

document 객체의 하위 객체 중 body element 노드를 반환한다.

**document.cookie**

document 객체의 cookie 정보를 반환한다.

**document.domain**

document 객체의 domain 정보를 반환한다.

## document.scripts

document 객체의 하위 객체 중 script element 노드 목록을 반환한다.

## document.title

document 객체 title 노드의 값을 반환한다.

**document.createElement();**

`document.createElement('div');`

element 노드를 새로 생성한다.

**document.getElementById();**

`document.getElementById('id');`

document 하위의 element 노드 객체 중 매개변수로 주어진 전달인수와 같은 이름의 id 를 가진 첫번째 element 객체 1개를 반환한다.

# `document.getElementsByClassName();`

`document.getElementsByClassName('class');`

`document` 하위의 `element` 노드 객체 중 매개변수로 주어진 전달인수와 같은 이름의 `class` 를 가진 `element` 노드 목록을 반환한다.

# `document.getElementsByName();`

`document.getElementsByName('name');`

`document` 하위의 `element` 노드 객체 중 매개변수로 주어진 전달인수와 같은 이름의 `name` 속성을 가진 `element` 노드 목록을 반환한다.

## `document.getElementsByTagName();`

```
document.getElementsByTagName('input');
```

`document` 하위의 `element` 노드 객체 중 매개변수로 주어진 전달인수와 같은 `tag` 이름을 가진 `element` 노드 목록을 반환한다.

## `document.write();`

```
document.write('hello');
```

`document` 하위 객체 중 `body` 에, 전달된 `String` 문자열을 써 넣는다.

# DOM Element

문서상의 HTML Element 를 의미한다.

## element.attributes

element 노드의 속성들을 담고 있는 NamedNodeMap 객체를 반환한다.  
(NamedNodeMap 은 속성 객체의 컬렉션 객체)

## element.childNodes

element 객체의 자식 노드들을 담고 있는 배열을 반환한다.



## `element.children`

`element` 객체의 자식 노드들 중 `element` 노드들만을 포함하고 있는 배열을 반환한다.

## `element.classList`

`element` 객체의 `class` 속성의 값을 담고 있는 `DOMTokenList` 객체를 반환한다.

**element.classList.add();**

**element.classList.add(class1, class2, ...);**

**element** 객체의 **classList** 객체에 하나 이상의 **class** 를 추가하는 메서드.

**element.classList.remove();**

**element.classList.remove(class1, class2, ...);**

**element** 객체의 **classList** 객체에 하나 이상의 **class** 를 삭제하는 메서드.

```
element.classList.toggle();
```

```
element.classList.toggle(class1, true|false);
```

element 객체의 classList 객체에 class 를 toggle 로 add/remove 할 수 있도록 하는 메서드.

```
element.classList.contains();
```

```
element.classList.contains('class');
```

element 객체의 classList 객체 중 매개변수로 주어진 전달인수와 같은 이름의 Class 를 가진 요소의 포함 여부를 Boolean 으로 반환한다.

## element.id

element 객체의 id 속성을 문자열로 반환한다.

## element.className

element 객체의 class 속성을 문자열로 반환한다.

## `element.firstChild`

`element` 객체의 첫번째 자식 노드를 반환한다.

## `element.firstChild`

`element` 객체의 첫번째 `element` 객체를 반환한다.

## `element.lastChild`

`element` 객체의 마지막 자식 노드를 반환한다.

## `element.lastElementChild`

`element` 객체의 마지막 `element` 객체를 반환한다.

## element.innerHTML

element 객체 내부의 HTML 을 문자열로 반환한다.

## element.style

element 객체의 style 속성의 값을 반환한다.

**element.addEventListener();**

`element.addEventListener('event', callback);`

element 노드에 이벤트 핸들러를 할당한다.

**element.appendChild();**

`element.appendChild(element1);`

element 노드 객체에 element1 노드를 추가한다.



**element.click();**

element 노드에 click 이벤트를 발생시킨다.

**element.cloneNode();**

element 노드를 복사한 element 를 반환한다.

**element.contains();**

**element.contains(element1);**

**element** 노드에 **element1** 노드의 포함 여부를 **Boolean** 으로 반환한다.

**element.getAttribute();**

**element.getAttribute('attribute1');**

**element** 노드 **attribute1** 속성 값을 반환한다.

**element.getAttributeNode();**

`element.getAttributeNode('attribute1');`

element 노드의 속성 노드들 중 attribute1 노드를 반환한다.

**element.getElementsByClassName();**

`element.getElementsByClassName('class');`

element 노드의 자식 element 중 class 를 포함하고 있는 element 노드 목록을 반환한다.

**element.hasAttributes();**

element 노드가 속성 노드를 포함하고 있는지 여부를 Boolean 으로 반환한다.

**element.insertBefore();**

**element.insertBefore(element1, element2);**

element 노드 내부에 element1 을 element2 의 앞쪽 위치에 추가한다.

**element.querySelector();**

`element.querySelector('query');`

element 노드의 자식 노드들 중 query 에 해당하는 노드를 탐색하여 그 첫번째 노드를 반환한다.

**element.querySelectorAll();**

`element.querySelectorAll('query');`

element 노드의 자식 노드들 중 query 에 해당하는 노드를 탐색하여 노드 목록을 반환한다.

**element.removeAttribute();**

**element.removeAttribute('attribute1');**

**element** 노드의 attribute1 속성을 삭제한다.

**element.removeChild();**

**element.removeChild(node);**

**element** 노드의 자식 노드들 중 **node** 를 삭제한다.

**element.removeEventListener();**

`element.removeEventListener('event', callback);`

element 노드의 event 이벤트에 할당된 이벤트 핸들러를 삭제한다.

**element.setAttribute();**

`element.setAttribute('attribute1', 'value');`

element 노드에 attribute1 속성을 추가하고 그 값은 value 로 할당한다.

**element.toString();**

**element** 노드를 **String** 으로 변환한다.



과제입니다.

# HOMEWORK

---

## Github 5Week Quest 풀이

### 제출

- github에 업로드

### 기한

- 2018. 4. 25. 23:00 까지