

13. HTML5 API






충남대학교 컴퓨터공학과
데이타베이스시스템 연구실

HTML5 Geolocation

❖ 사용자의 지리적 위치를 가져오는데 사용

- 위치정보는 사용자의 개인 정보이기 때문에 사용자가 승인하지 않으면 활용할 수 없음
- Geolocation은 smartphone과 같은 GPS를 가진 디바이스에서 더 정확함

❖ 브라우저 지원 현황

API					
Geolocation	5.0	9.0	3.5	5.0	16.0

HTML5 Geolocation

- ❖ 위치 정보는 navigator 객체가 가지고 있는 geolocation 객체를 통해 얻을 수 있음

```
var geolocation = navigator.geolocation;
```

❖ 메소드들

method	description
getCurrentPosition()	사용자의 현재 위치 정보를 반환한다.
watchPosition()	장치의 현재 위치에 대한 정보를 주기적으로 반환한다
clearWatch()	현재 진행 중인 watchPosition() 실행을 중지한다.

getCurrentPosition() 사용

❏ getCurrentPosition()

○ 세 개의 인수

- 성공 시 호출되는 콜백 함수, 실패 시 호출되는 콜백 함수, 옵션
- 옵션: enableHighAccuracy(높은정확도), timeout(시간제한설정), maximumAge(유효기간 설정)

• 예) var options={enableHighAccuracy:true, timeout:1000, maximumAge:6000};

❏ 예 : 사용자 위치의 경도 및 위도를 반환 *Try it!*

- If (navigator.geolocation)문 : 위치 정보가 지원되는지를 검사
 - 지원된다면 getCurrentPosition() 메소드 실행
 - getCurrentPosition ()이 위치 정보를 얻는데 성공하면, showLocation() 함수가 호출되고 인수로 좌표 객체를 가지고 있는 Position 객체가 반환
- showGeolocation()함수는 Position객체에서 위도 및 경도를 추출해서 화면에 표

getCurrentPosition() 사용 (cont'd)

- ❏ getCurrentPosition() 메소드가 성공 할 때 전달되는 객체에 포함된 속성들

Property	Description
coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

- ❏ 오류 처리 *Try it!*

- getCurrentPosition() 메소드의 두 번째 매개 변수로 오류를 처리하는 콜백 메소드를 넘겨줌

지도에 위치 표시하기

- ❖ 지도의 결과를 표시하기 위해서, 구글 맵(google Map) 처럼 위도 및 경도를 사용할 수 있는 지도 서비스에 대한 액세스가 필요
 - 예 : 현재 위치의 위도 및 경도 데이터를 사용하여 구글 서비스에서 정적 이미지 형태로 새로운 윈도우에서 보여줌 Try it!
- ❖ Google Maps JavaScript API v3

watchPosition(), clearWatch()

❏ watchPosition()

- 사용자의 현재 위치를 연속하여 출력
- 자동차처럼 사용자가 이동하고 있으면 계속 업데이트된 위치를 반환

❏ clearWatch()

- watchPosition() 메소드를 중지

❏ 예 : 스마트폰처럼 정밀한 GPS 장치를 가지고 있으면 이동 중에 변환되는 위치가 출력됨 *Try it!*

위치 정보 응용

예 : 구글 API를 이용하여 현 위치의 주소를 출력 Try it!

- google_maps_Geocoder() 객체와 geocode 메소드 응용
 - google_maps_Geocoder 클래스 : 주소와 LatLng 간의 변환을 해주는 서비스
Reference

Click the button to get your coordinates:

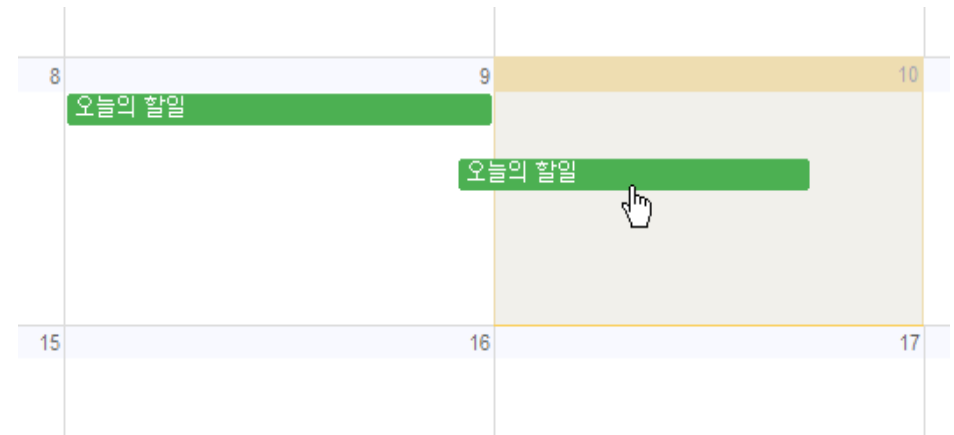
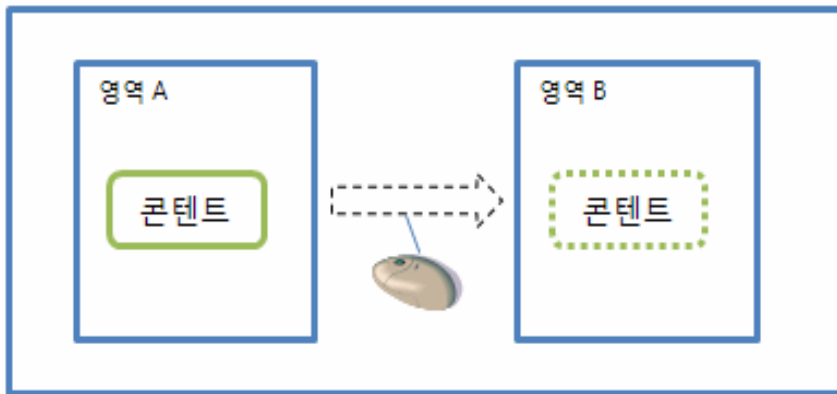


Address : 대한민국 대전광역시 서구 둔산1동 1420

Try It






HTML5 Drag and Drop

- ❖ 애플과 윈도우에서 가장 많이 사용하는 사용자 인터페이스 중의 하나
 - 예) 구글 캘린더에서 끌어다 놓기 기능을 이용하여 할일 목록을 다른 날짜로 이동할 수 있음
- ❖ 영역 A → 영역 B로 특정 콘텐츠를 마우스로 이동



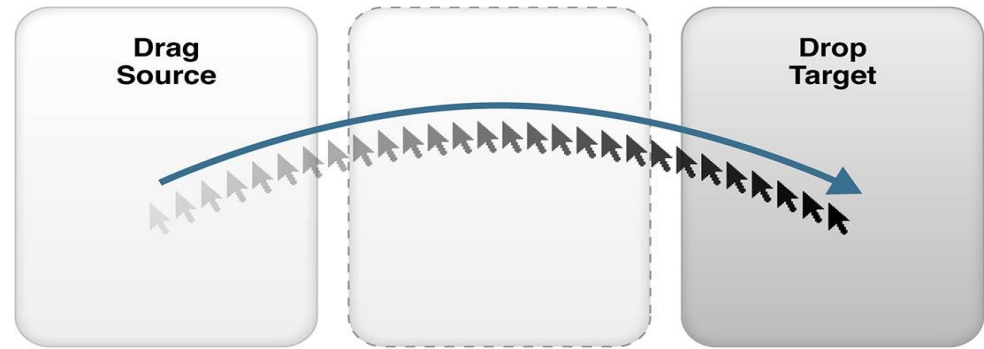
HTML5 Drag and Drop (cont'd)

브라우저 지원 현황

API					
Drag and Drop	4.0	9.0	3.5	6.0	12.0

- 그 외 참고 사이트 : <http://caniuse.com/>

Drag and Drop 구성



❖ 드래그 대상(drag source)

- HTML 요소에 draggable 속성값을 true로 설정

```

```

❖ 드롭 타겟(drop target)

- drop이 가능하도록 설정하려면, 이벤트의 preventDefault() 메소드를 호출하여 기본 값을 취소

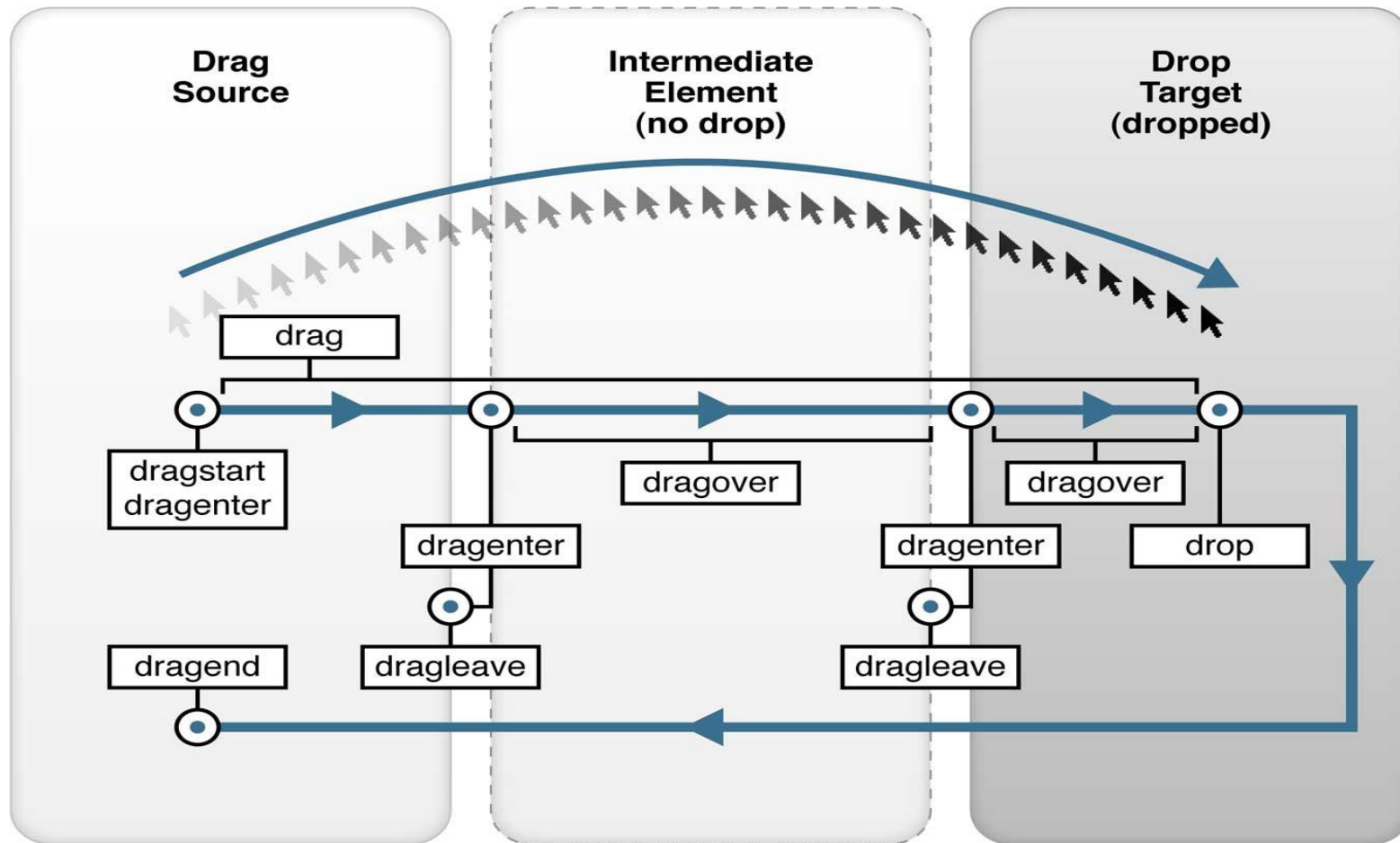
❖ 드래그 데이터(drag data)

- 이동할 데이터는 dataTransfer 객체를 통해 이루어지는데, 드래그 대상(drag source)에서 이 객체에 값을 저장하고 드롭 타겟(drop target)에서 이 객체의 값을 꺼내어 오는 방식

이벤트

이벤트	이벤트 알리는 곳	설명
dragstart	드래그 대상 요소	드래그가 시작
drag	드래그 대상 요소	드래그 중
dragenter	드래그 중 마우스 커서가 위치한 요소	드래그 조작이 요소 안의 범위에 들어옴
dragover	드래그 중 마우스 커서가 위치한 요소	드래그 조작이 요소 안의 범위를 통과 중
dragleave	드래그 중 마우스 커서가 위치한 요소	드래그 조작이 요소 안의 범위를 벗어남
drop	드롭할 곳의 요소	드롭되었음
dragend	드래그 대상 요소	드래그 종료

Drag and Drop event flow



HTML5 Drag and Drop 예 Try it!

Try it!을 위한 설명

- dragstart -> function drag(ev)
 - dataTransfer 객체에 "Text" 형식의 id("drag1")를 보냄

```
function drag (ev) {  
    ev.dataTransfer.setDate("text", ev.target.id);  
}
```

- dragover -> function allowDrop(ev)
 - 기본적으로 모든 요소들은 다른 요소에 드롭될 수 없음
 - 그렇게 때문에 drop 이벤트에 대한 처리를 위해서는 기본적인 동작을 막아야 함
 - 그러므로 디폴트 처리를 방지하려면 event.preventDefault()를 함

```
event.preventDefault();
```

HTML5 Drag and Drop 예 (cont'd)

- Drop -> function drop(ev)
 - preventDefault()를 호출하여 브라우저의 기본적인 동작 막음
 - dataTransfer 객체에서 getData()를 이용하여 필요한 데이터("drag1")를 꺼내 자식 노드에 추가

```
function drop (ev) {  
    ev.preventDefault();  
    var data=ev.dataTransfer.getData("text");  
    ev.target.appendChild(document.getElementById(data));  
}
```

HTML5 Web Storage

📦 쿠키(cookie)

- 클라이언트에 간단한 정보를 저장하기 위해 사용
- 간단하고 편리함
- 단점
 - 데이터 저장 용량이 4kb 밖에 되지 않음
 - HTTP Request 호출 시 마다 헤더 전송 필요 (응답 속도 저하의 원인)
 - 창 단위로 범위가 나뉘어지지 않음 -> Session Storage로 극복
 - 유효기간 제한 -> Local Storage로 극복
 - 보안에 취약
 - 문자열만 저장 가능

HTML5 Web Storage (cont'd)






- 웹 스토리지는 로컬 스토리지(local storage)와 세션 스토리지(session storage)로 나뉨

	LocalStorage	SessionStorage
저장공간	도메인 기준	브라우저 기준
저장타입	장기적인 저장 공간 (브라우저를 닫아도 데이터 유지)	일시적인 저장 공간 (브라우저를 닫으면 데이터 사라짐)
데이터공유	현재 열린 모든 브라우저 창	현재 브라우저 창
기타 특징	대용량 쿠키	새 브라우저 창이 열렸을 경우 부모 창 데이터 복사

- 웹 스토리지 안에서 데이터는 키/값 (key/value)쌍으로 저장됨
- 웹 스토리지는 약 5MB정도 까지 저장이 가능하며, 쿠키와 달리 서버에 정보를 보내지 않는다.

HTML5 Web Storage (cont'd)

브라우저 지원

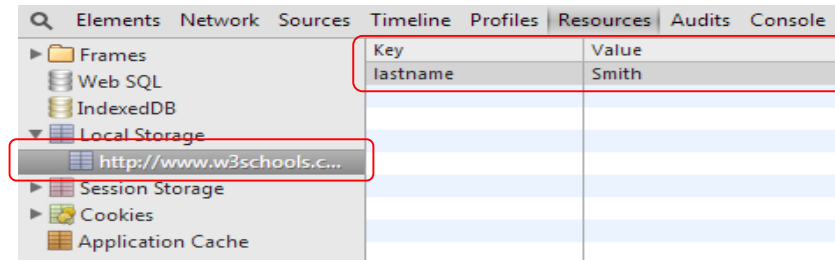
API					
Web Storage	4.0	8.0	3.5	4.0	11.5

Web Storage의 속성과 메소드

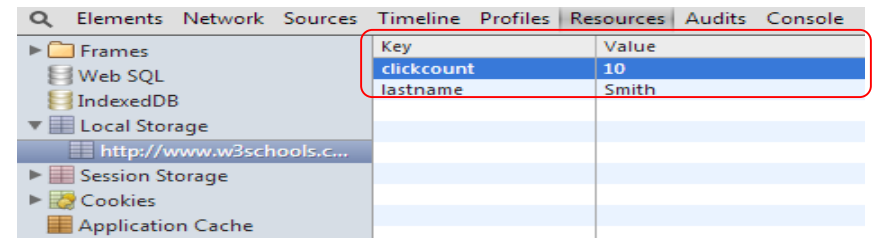
메소드/속성	내용
length	저장되어 있는 데이터(key/value)의 수
key(index)	index의 위치에 있는 key를 반환
getItem(key)	Key과 연관된 value 반환, 존재하지 않으면 null 반환
setItem(key, value)	key/value를 저장
removeItem(key)	key/value를 삭제
clear()	모든 key/value를 삭제

Web Storage의 예

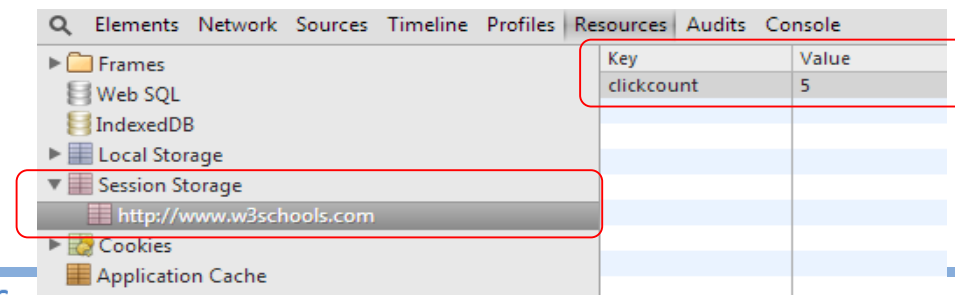
- ❖ localStorage 예 1: key/value ("lastname"/"Smith")을 저장하고 읽는 예 *Try it!*



- ❖ localStorage 예2 : 사용자가 버튼을 클릭한 횟수를 세는 예 *Try it!*



- ❖ sessionStorage 예 : 현재의 session에서 사용자가 버튼을 클릭한 횟수를 세는 예 *Try it!*



웹 스토리지 API 사용

■ 값 설정

- ① `sessionStorage.setItem('a', 1);`
- ② `sessionStorage.a = 1;`
 - 두 번째 방법은 한글이나 공백이 포함된 값과 같이 변수로 사용될 수 없는 값에는 제한

■ 값 읽기

- ① `var a = localStorage.getItem('a');`
- ② `var a = localStorage.a;`
 - 역시, 두 번째 방법은 변수로 사용될 수 없는 key 값에 대해서는 제한

웹 스토리지 API 사용 (cont'd)

key 읽기

- ① `localStorage.setItem('a',1);`
- ② `localStorage.setItem('b',2);`
- ③ `var keys = localStorage.key(0) + localStorage.key(1)`
 - IE와 오페라의 경우는 'ab', 사파리, 크롬, 파이어폭스는 'ba'

값 삭제

- `localStorage.removeItem('a');`
- `localStorage.clear();`

HTML5 Application Cache

❖ 애플리케이션 캐시(application cache)란?






- 애플리케이션이 사용하는 파일을 클라이언트의 캐시(cache)에 저장하는 기술

❖ 애플리케이션 캐시가 애플리케이션에게 제공하는 장점

- 오프라인 브라우징 - 오프라인 상태일 때도 사용자는 웹 애플리케이션을 사용할 수 있음
- 스피드- 캐시된 파일은 더 빨리 로드됨
- 서버 부하가 감소됨 - 브라우저는 서버로부터 변경된 리소스만을 다운로드 하면 됨

HTML5 Application Cache (cont'd)

브라우저 지원

API		 Internet Explorer			
Application Cache	4.0	10.0	3.5	4.0	11.5

매니페스트 파일 (manifest file)

- ❖ 애플리케이션 캐시를 활성화하려면, 문서의 <html> tag에 manifest 속성을 포함해야 함

- 시계 애플리케이션의 예 *Try it!* (참고 : <http://www.whatwg.org/>)

```
<!DOCTYPE HTML>
<html manifest="clock.appcache">
<body>
...
</body>
</html>
```

- 매니페이스트 파일의 추천된 파일 확장명은 ".appcache"임
- 매니페이스트 파일의 MIME 형식은 "text/cache-mainfest"임. 이것을 처리 할 수 있도록 웹 서버의 config 파일에 추가해야 함

매니페스트 파일 (manifest file) (cont'd)

- 단순한 텍스트 파일이며, 캐시해야 될 것을 브라우저에게 지시
 - clock.appcache 파일의 예

CACHE MANIFEST

clock.html

clock.css

clock.js

매니페스트 파일 (manifest file) 구성

3개의 섹션으로 구성됨

CACHE MANIFEST

- 이 헤더 아래에 나열된 파일들은 처음으로 다운로드 된 후에 캐시됨
- 예 : 매니페이스 파일이 로드될 때, 브라우저는 웹 사이트의 루트 디렉토리에서 아래의 3개의 파일을 다운로드함.

CACHE MANIFEST

/theme.css

/logo.gif

/main.js

매니페스트 파일 (manifest file) 구성 (cont'd)

○ NETWORK

- 이 헤더 아래에 나열된 파일들은 반드시 서버에 연결해야 접근이 가능한 파일을 명시함
- 예 : “login.asp”파일은 캐시될 수 없음. 즉, 오프라인에서 활용될 수 없음

```
NETWORK :  
login.asp
```

- 예 : 모든 리소스와 화일들은 인터넷 연결을 해야 사용할 수 있음을 나타냄

```
NETWORK :  
*
```

매니페스트 파일 (manifest file) 구성 (cont'd)

○ FALLBACK

- 이 헤더 아래에 나열된 파일들은 만일 페이지를 접근할 수 없을 경우에 대체되는 페이지임
- 예 : 인터넷 연결이 될 수 없을 때, /html/ 카탈로그 안에 있는 모든 화일들에 대해 "offline.html" 로 대체
 - 처음 URI : 리소스, 두번째 : fallback

FALLBACK:

/html/ /offline.html

매니페스트 파일 (manifest file) 구성 (cont'd)

❏ 완전한 매니페스트 파일의 예

```
CACHE MANIFEST
#2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
Login.asp

FALLBACK:
/html/ /offline.html
```

- ❏ '#'은 주석 라인임

캐시 업데이트

- ❖ 어플리케이션이 캐시될 때, 다음 중 하나가 발생할 때까지 캐시 상태를 유지함
 - 사용자가 브라우저의 캐시를 지운다.
 - 매니페스트 파일이 수정된다
 - 어플리케이션 캐시가 프로그램으로 업데이트된다.
- ❖ Tip
 - 반드시 매니페이스 파일이 바뀔 때에만 캐시를 갱신함
 - 주식 줄에서 날짜와 버전을 업데이트하는 것은 브라우저가 파일을 다시 캐시하도록 하는 방법 중에 하나
- ❖ Chrome에서 사이트별 상태 확인
 - `chrome://appcache-internals/`

File API

- ❑ 웹 브라우저가 사용자 컴퓨터에 있는 로컬 파일을 읽어올 수 있도록 해 주는 API
- ❑ PC의 파일을 스크립트로 읽기
 - 사용자가 드래그 앤 드롭을 한 파일
 - Input 요소의 type속성을 file로 지정하여 선택한 파일
- ❑ 파일의 수정이나 삭제는 할 수 없음
- ❑ 사용자가 입력 양식을 통하여 파일 이름을 입력하면 이것에서 File 객체를 추출하고 FileReader 객체를 통하여 파일의 내용을 읽음

File API (cont'd)

브라우저 지원 (<http://caniuse.com/>)

File API - **Working Draft**

Method of manipulating file objects in web applications client-side, as well as programmatically selecting them and accessing their data.

*Usage stats:

Global

Support: 72.19%

Partial support: 6.22%

Total: 78.41%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
	8.0		31.0			4.2-4.3		4.0		
	9.0		32.0			5.0-5.1		4.1		
	10.0	27.0	33.0			6.0-6.1		4.2-4.3	7.0	
Current	11.0	28.0	34.0	7.0	20.0	7.0	5.0-7.0	4.4	10.0	10.0
Near future		29.0	35.0		21.0					
Farther future		30.0	36.0		22.0					
3 versions ahead		31.0	37.0							

파일 API에서 사용되는 객체

File 객체 *Try it!*

- 로컬 파일 시스템에서 얻어지는 파일 데이터

속성/메소드	설명
name	파일의 이름
size	파일의 크기(단위:바이트)
type	파일의 타입(MIME type)
lastModifiedDate	최종 변경 날짜
slice(start, length)	시작 위치와 길이를 지정하여 파일의 내용을 잘라서 새로운 Blob(Binary Large Object) 객체를 만듦

파일 API에서 사용되는 객체 (cont'd)

❏ FileReader 객체 *Try it!*

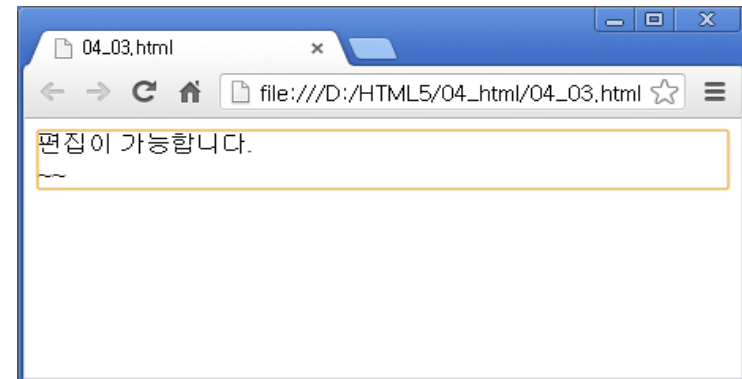
- 이벤트 처리를 통하여 파일의 데이터에 접근하는 메소드를 제공하는 객체

속성/메소드	설명
readAsBinaryString(fileBlob)	파일 내용을 읽어 들여 바이너리 문자열로 저장
readAsText(fileBlob, encoding)	파일 내용을 읽어 들여 문자열로 저장
readAsDataURL(file)	파일 내용을 읽어 들여 dataURL 형식의 문자열로 저장
result	읽어 들인 파일의 내용
error	실패시 발생
load	읽어 들이기에 성공했을 때 발생
progress	읽어 들이는 동안에 주기적으로 발생

텍스트 편집: 편집 속성 *Try it!*

- Contenteditable(특정 요소 편집여부), designMode(페이지 전체) 속성 사용
- contenteditable 속성값은 true, false, inherit
- designMode 속성값은 on, off

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <script type="text/javascript">
        window.onload = function()
        {
            var editor = document.getElementById("editor");
            if(editor.isContentEditable)
            {
                editor.focus();
            }
        };
    </script>
</head>
<body>
    <div id="editor" contenteditable></div>
</body>
</html>
```

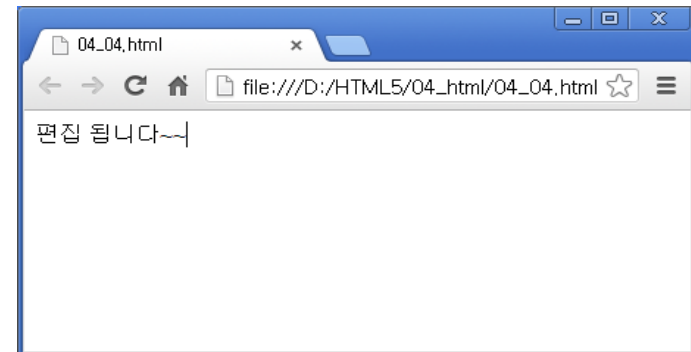


텍스트 편집 *Try it!*

■ 편집 속성 부여

- Contenteditable(특정 요소 편집여부), designMode(페이지 전체) 속성 사용
- contenteditable 속성값은 true, false, inherit
- designMode 속성값은 on, off

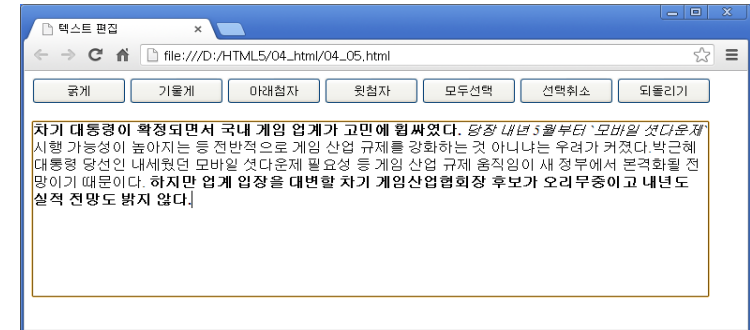
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <script type="text/javascript">
        window.onload = function()
        {
            document.designMode = "on";
        };
    </script>
</head>
<body>
    <div></div>
</body>
</html>
```



execCommand 메소드를 이용한 텍스트 편집 *Try it!*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> 텍스트 편집 </title>

    <script type="text/javascript">
      window.onload = function()
      {
        var editor = document.getElementById("editor");
        if (editor.isContentEditable) editor.focus();
      };
      function setBold()
      {
        document.execCommand("bold");
      }
    :
  <body>
    <input type="button" value="굵게" onclick="setBold();">
    <input type="button" value="기울게" onclick="setItalic();">
    <input type="button" value="아래첨자" onclick="setSubscript();">
    :
    <div id="editor" contenteditable></div>
  </body>
</html>
```



웹 워커(Web Workers)

- ❖ 페이지의 성능에 영향을 주기 않고, 백그라운드에서 실행되는 자바 스크립트
- ❖ 시간이 많이 걸리는 작업을 웹 워커에게 위임하면, 사용자는 웹 페이지에서 자신이 원하는 작업을 계속할 수 있음
 - 매우 복잡한 수학적 계산 작업
 - 원격지에 있는 소스에 대한 access
 - 로컬 스토리지 access 작업
 - 백그라운드에서 조용히 오랜 시간 해야 하는 작업
 - UI Thread에 방해 없이 지속적으로 수행해야 하는 작업

웹 워커(Web Workers) (cont'd)

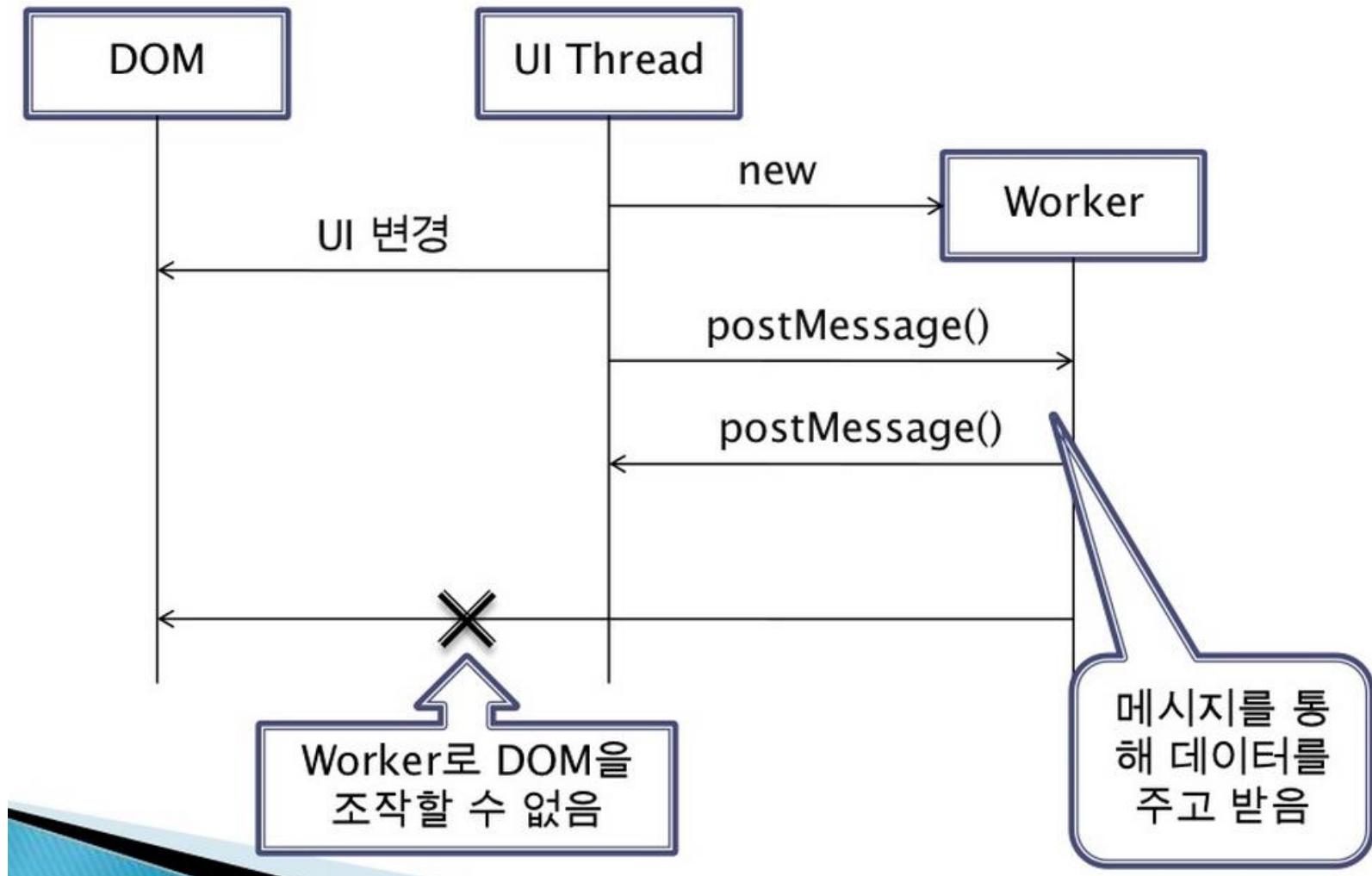
❏ DOM 조작 불가능

- Window object 조작 불가능
- Document object 조작 불가능
- Parent object 조작 불가능

❏ 웹 워커에서 사용 가능한 요소

- Object, Array, Date, Math, String 등의 JavaScript 객체
- navigator object
- location object(read only)
- setTimeout(), clearTimeout(), setInterval(), clearInterval()
- XMLHttpRequest
- Application Cache

웹 워커(Web Workers) (cont'd)








웹 워커(Web Workers) (cont'd)

웹 워커의 종류

- 전용 워커(Dedicated Worker)
 - 워커를 생성한 페이지에만 백그라운드 프로세스가 응답하는 형태
- 공유 워커(Shared Worker)
 - 하나의 워커가 여러 개의 문서에 응답하는 형태

웹 워커의 브라우저 지원

API					
Web Workers	4.0	10.0	3.5	4.0	11.5

전용 워커 (Dedicated Worker) *Try it!*

- ❖ 웹 워커를 만들기 전에 사용자의 브라우저가 지원하는지를 검사

```
if (typeof(Worker) != "undefined")  
{  
    //Yes! 웹 워커 지원  
}  
else  
{  
    // Sorry! 웹 워커를 지원하지 않음  
}
```

전용 워커 (Dedicated Worker) (cont'd)

❖ 웹 워커 파일 만들기(외부 자바스크립트 파일로 작성할 경우)

- "demo_workers.js"파일로 저장

```
var i=0;

function timedCount()
{
    i=i+1;
    postMessage(i);
    setTimeout("timedCount()", 500);
}
timedCount();
```

- postMessage(i) : 웹 워커에서 HTML 페이지로 다시 메시지를 보낼 때 사용

전용 워커 (Dedicated Worker) (cont'd)

❖ 웹 워커 객체 만들기

- HTML 페이지에서 이전 장에서 만든 JS 파일 호출
- 워커가 이미 존재하는지를 검사. 웹 워커가 없으면 새로운 웹 워커 객체를 만들고, "demo_workers.js"를 실행

```
if (typeof(w) == "undefined")  
{  
    w = new Worker("demo_workers.js");  
}
```

- 웹 워커로부터 메시지를 수신하기 위해 웹 워커의 "onmessage" event listener를 추가

```
w.onmessage = function(event) {  
    document.getElementById("result").innerHTML=event.data;  
};
```

전용 워커 (Dedicated Worker) (cont'd)

❖ 웹 워커의 종료

- 웹 워커 객체가 만들어질 때, 객체가 종료될 때까지 메시지 수신을 계속함.
- 웹 워커를 종료하기 위해 `terminate()` 메소드를 사용

```
w.terminate();
```

공유 워커(Shared Worker)

- 공유 워커 객체 생성 (in main JavaScript thread)

```
var worker = new SharedWorker('sharedWorker.js');
```

- 공유 워커에 메시지를 보냄 (in main JavaScript thread)

```
worker.port.postMessage(msg);
```

- <http://tutorials.jenkov.com/html5/web-workers.html#sharedworker>

웹 소켓(Web Socket)

웹 소켓이란?

- 웹 환경에서 실시간 양방향 통신을 위한 스펙
- 웹 소켓을 이용하면 일반적인 TCP 소켓과 같이 연결지향 양방향 통신이 가능

브라우저 지원 (<http://caniuse.com/>)

# Web Sockets - Candidate Recommendation										*Usage stats: Global	
Bidirectional communication technology for web apps										Support:	72.24%
										Partial support:	1.94%
										Total:	74.18%
Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile	
								2.1			
								2.2			
						3.2		2.3			
						4.0-4.1		3.0			
	8.0		31.0			4.2-4.3		4.0			
	9.0		32.0			5.0-5.1		4.1			
	10.0	27.0	33.0			6.0-6.1		4.2-4.3	7.0		
Current	11.0	28.0	34.0	7.0	20.0	7.0	5.0-7.0	4.4	10.0	10.0	
Near future		29.0	35.0		21.0						
Farther future		30.0	36.0		22.0						
3 versions ahead		31.0	37.0								

웹 소켓(Web Socket) (cont'd)

❏ 서버 연결

- HTML5가 제공하는 WebSocket 객체를 통해 서버 연결
- 일반 통신 : ws, 보안 통신 : wss 프로토콜을 이용

```
var ws = new WebSocket ("ws://echo.websocket.org");
```

- //echo.websocket.org는 WebSocket.org사이트에서 테스트용으로 제공하는 에코 서버. 즉, 클라이언트가 보낸 데이터를 다시 돌려보내줌 *Try it!*

❏ 데이터 송신

- WebSocket 객체의 send함수를 사용하여 데이터를 서버로 송신

```
ws.send("송신 메시지");
```

```
ws.send(document.getElementById("data").value);
```

웹 소켓(Web Socket) (cont'd)

데이터 수신

- 서버에서 전송하는 데이터를 받으려면 웹 소켓 객체의 message 이벤트 처리기를 구현

```
ws.onmessage = function (evt) {  
    var msg = evt.data;  
    document.getElementById("result").innerHTML=msg;  
};
```

웹 소켓(Web Socket) (cont'd)

웹 소켓의 이벤트

- 웹 소켓 객체에서 발생하는 이벤트

이벤트	설명
onopen	소켓 연결이 확립되면 발생
onclose	연결이 종료되면 발생
onerror	통신에서 오류가 있을 때 발생
onmessage	서버로부터 데이터가 도착하면 발생

웹 소켓의 메소드

이벤트	설명
send(data)	웹 소켓을 통해 데이터를 보낸다.
close()	연결을 해제한다.

웹 소켓 예 *Try it!*

```
<!DOCTYPE html>
<html> <head>
  <script>
    var ws;
    function open1(){
      if("WebSocket" in window){
        ws = new WebSocket("ws://echo.websocket.org");

        ws.onopen=function() { alert("웹 소켓 오픈 성공");};

        ws.onmessage=function(evt) { var msg=evt.data;
          document.getElementById("result").innerHTML=msg; };

        ws.onclose=function(){ alert("웹 소켓 연결 해제");}}
      else { alert("웹 소켓이 지원되지 않음");} }

    function send() { ws.send(document.getElementById("data").value); }

    function quit() { ws.close();}
  </script>
```

웹 소켓 예 (cont'd)

```
<body>
    <button onclick="open1()">웹 소켓 연결</button>
    <button onclick="quit()">웹 소켓 연결 종료</button>
    <input type="text" id="data">
    <button onclick="send()">데이터 송신</button><br>
    에코 서버로 부터 받은 데이터 :
    <output id="result"></output>

</body>
</html>
```

웹 소켓 서버 구축하기

❖ 웹 소켓 서버

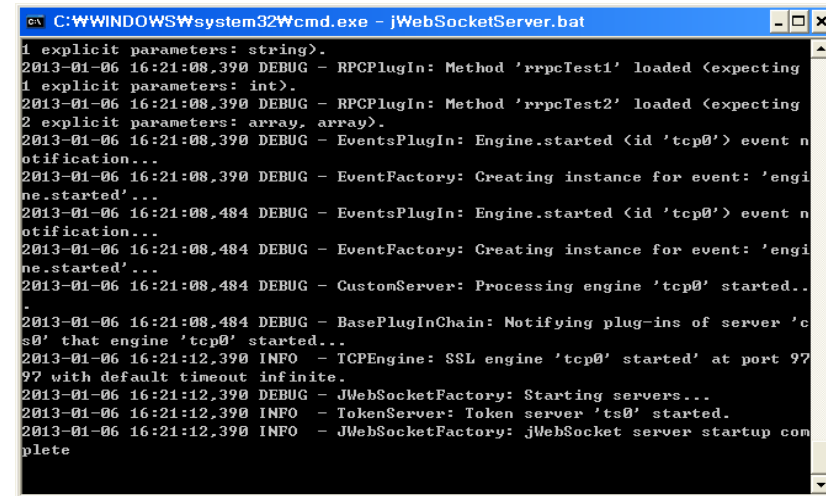
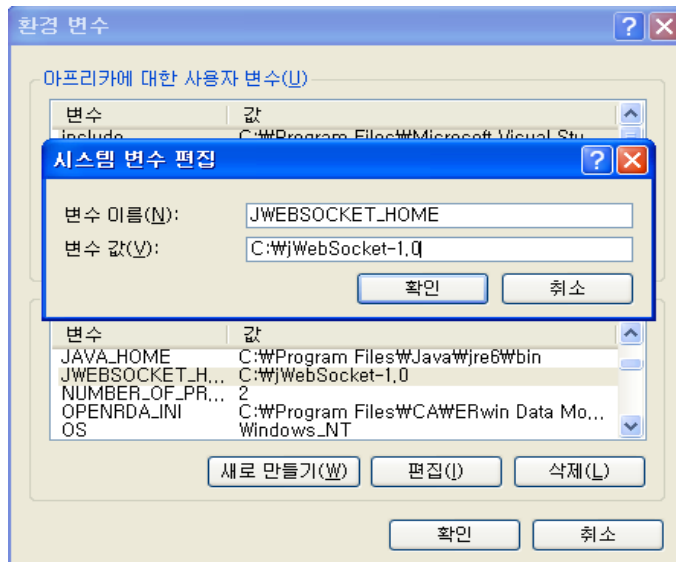
- 웹 소켓은 일반적인 TCP 소켓과는 다른 프로토콜로 설계
- 기존 TCP 서버를 그대로 이용할 수 없고 새로 구현해야 하는데 웹 소켓 서버 사양에 맞도록 구현
- pywebsocket , phpwebsocket, jWebSocket, web-socket-ruby, Socket.IO-node

❖ WebSocket 라이브러리

- <http://jwebsocket.org/>
 - 자바로 구현된 웹 소켓 서버모듈인 jWebSocketServer
 - 자바스크립트로 구현된 웹 소켓 클라이언트 데모인 jWebSocketClient
 - jWebSocket 라이브러리의 전체 소스코드는 jWebSocketFullSource

웹 소켓 서버 구축하기(cont'd)

- ❖ jWebSocketServer 다운받아 압축을 해제
- ❖ 환경변수에 JWEBSOCKET_HOME 경로 지정








- ❖ bin 폴더의 jWebSocketServer.bat 파일실행
 - 통신로그와 메시지가 이 창에 기록됨
 - 실행창을 닫으면 서버도 종료됨

HTML5 서버 전송 이벤트(Server-Sent Events)

- 웹 페이지가 서버로부터 자동적으로 데이터를 전달받을 수 있는 방법

- 예: Facebook/Twitter 갱신, 주식 가격 갱신, 뉴스 피드, 스포츠 결과 등

- 브라우저 지원

API					
SSE	6.0	Not supported	6.0	5.0	11.5

HTML5 서버 전송 이벤트 (cont'd) *Try it!*

❖ 웹 브라우저가 서버-전송 이벤트를 지원하는지 검사 (server_event.html)

```
If (typeof(EventSource) == "undefined")
{
    //Yes! Server-sent events support!
    //some code
}
else
{
    //sorry! No server-sent events support..
}
```

- 서버-전송 이벤트는 EventSource 객체를 통하여 구현되기 때문에 이 객체가 존재하는지 체크

HTML5 서버 전송 이벤트 (cont'd)

❏ 데이터 받기 (server_event.html)

```
var source = new EventSource("test.jsp");
source.onmessage=function(event) {
    document.getElementById("result").innerHTML +=event.data + "<br>";
};
```

- EventSource 객체를 생성하고, 인수로 주어진 URL ("test.jsp")에 접속해 데이터를 주기적으로 받음
- 서버로부터 갱신 데이터를 받을 때마다, onmessage 이벤트를 발생
- onmessage 이벤트가 발생할 때, id="result"안에 받은 데이터를 추가

HTML5 서버 전송 이벤트 (cont'd)

❖ EventSource 객체의 이벤트들

이벤트	설명
onopen	서버로 연결이 이루어지면 발생
onmessage	메시지를 받았을 때 발생
onerror	에러가 발생할 때

HTML5 서버 전송 이벤트 (cont'd)

❏ 서버측 코드 예 (test.jsp)

```
<%@ page import="java.util.Date" %>
<%
    response.setContentType("text/event-stream;charset=utf-8");
    Date time=new Date();
%>
data:<%=time%>
```

- "Content-Type"헤더를 "text/event-stream"으로 설정
- 현재 시간을 받아 변수 time에 저장
- 클라이언트로 데이터를 출력
 - data: 현재 시간