HTML5

2019.03

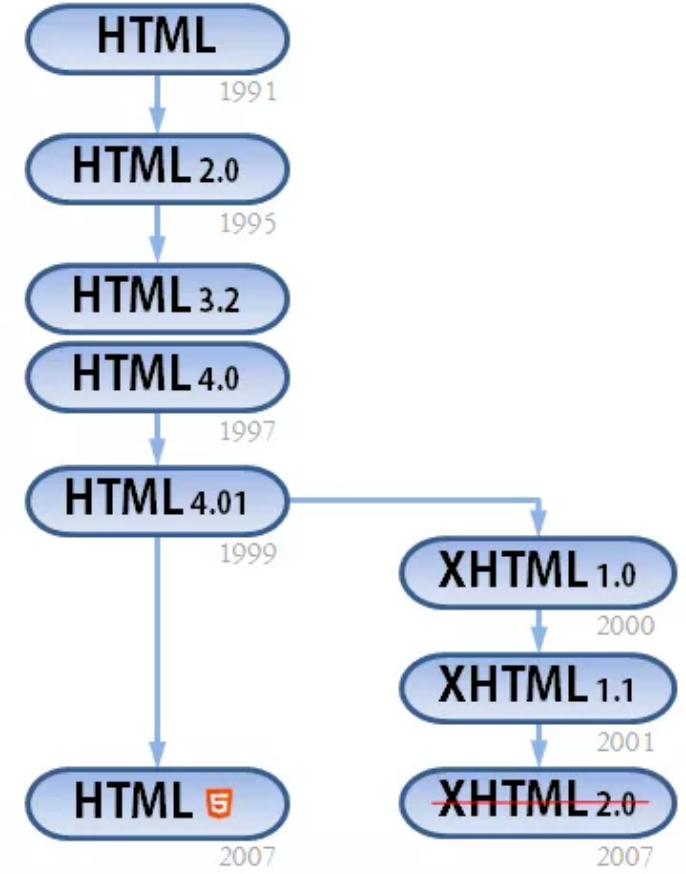
손찬욱

목차

HTML5가 탄생하기 까지

Brief History

- HTML4 1997년 표준 제정 이후, HTML 논의 중단됨
- 이후 W3C는 HTML 발전보다는 XHTML(XML + HTML) 작업을 시작
  → HTML4를 XML로 변경하는 작업으로 2000년 XHTML 1.0 발표
  → HTML과 XHTML 호환되지 않는 XHTML2 표준화 시작
- Apple, Mozilla와 Opera는 W3C의 XHTML 표준화에 대한 우려로 WHATWG (Web Hypertext Application Technology Working Group) 워킹 그룹 생성
- 2007년 W3C 내부에 HTML 워킹그룹이 생기고 WHATWG와 함께 HTML5 표준화 작업 시작
- 이후 모든 WHATWG 그룹내 논의되던 Web Applications 1.0 스펙은 W3C로 이동되고, 이를 HTML5라 명명
- HTML5의 스펙이 너무 느리다고 하여 WHATWG로 다시 분리.
  WHATWG는 빠르게 현실을 반영하고 HTML5는 WHATWG의 기능을 표준화함.



HTML5?

HTML5는 무엇인가?

HTML5 semantics + CSS3 +
Web Application JavaScript API

HTML5 semantics
→ 의미론적인 웹 구현. 기존 웹을 확장해 컴퓨터가 정보를 잘 이해할 수 있도록 의미를 잘 정리해 전달

CSS3
→ 컨텐츠 표현의 다양화 (애니메이션, 웹 폰트, 등)

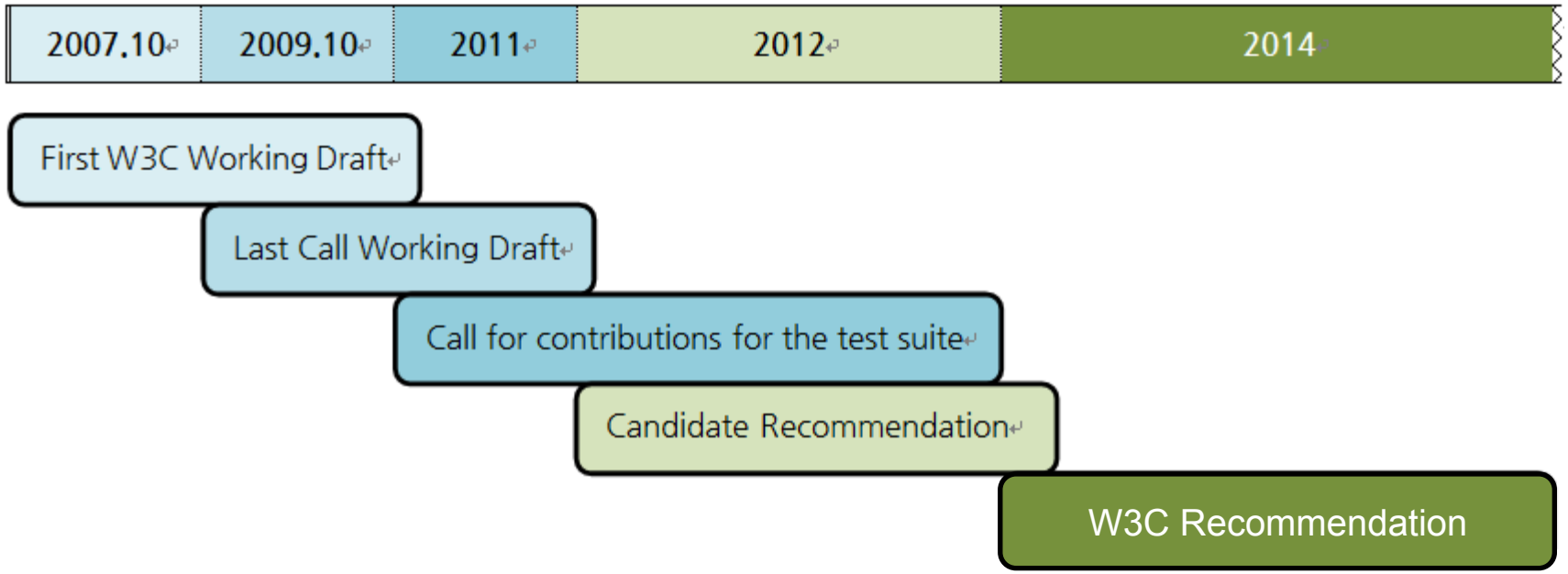
Web Application JavaScript API
→ JavaScript API를 통한 이용한 신기술 활용

HTML5 분류

분류	설명
Semantics	의미론적인 데이터 구성. Web Components
Device Access	GeoLocation, Device Orientation/Motion, Web Notification
Multimedia	Audio/video, Streaming
3D, Graphics & Effects	SVG, Canvas, Web GL/VR, Animation
Connectivity	효율적인 새로운 통신방법 Web Sockets, Server-sent Events, Web RTC
Performance & Integration	UI (Drag & Drop), Web Worker, Security, Payment
Offline & Storage	PWA (Service Worker, Push Messages) Web Storage, Database Storage, File
Others	Scripting (ES5/6/…)

HTML5 is Now!

Timetable



- 2014년 10월28일 - W3C Recommendation
- 2016년 11월 1일 HTML5의 마이너 업데이트 버전인 HTML5.1 표준안을 확정
- 2017년 12월 14일 HTML5.2 표준안을 확정했다
- HTML5.3 표준안은 현재 작업 초안 단계로 진행 중

HTML5 지원여부

브라우저 벤더 이슈

- 모든 HTML5 스펙을 모든 브라우저에서 사용할 수는 없다.
- 스펙이 발표되었더라도 브라우저 벤더에서 지원하지 않는다면 스펙일 뿐 실제 사용할 수는 없다.
- 브라우저 벤더에 따라 지원하는 HTML5 기술의 범위가 다르며, 각 회사의 이해 관계가 얽혀 있기도 하다.

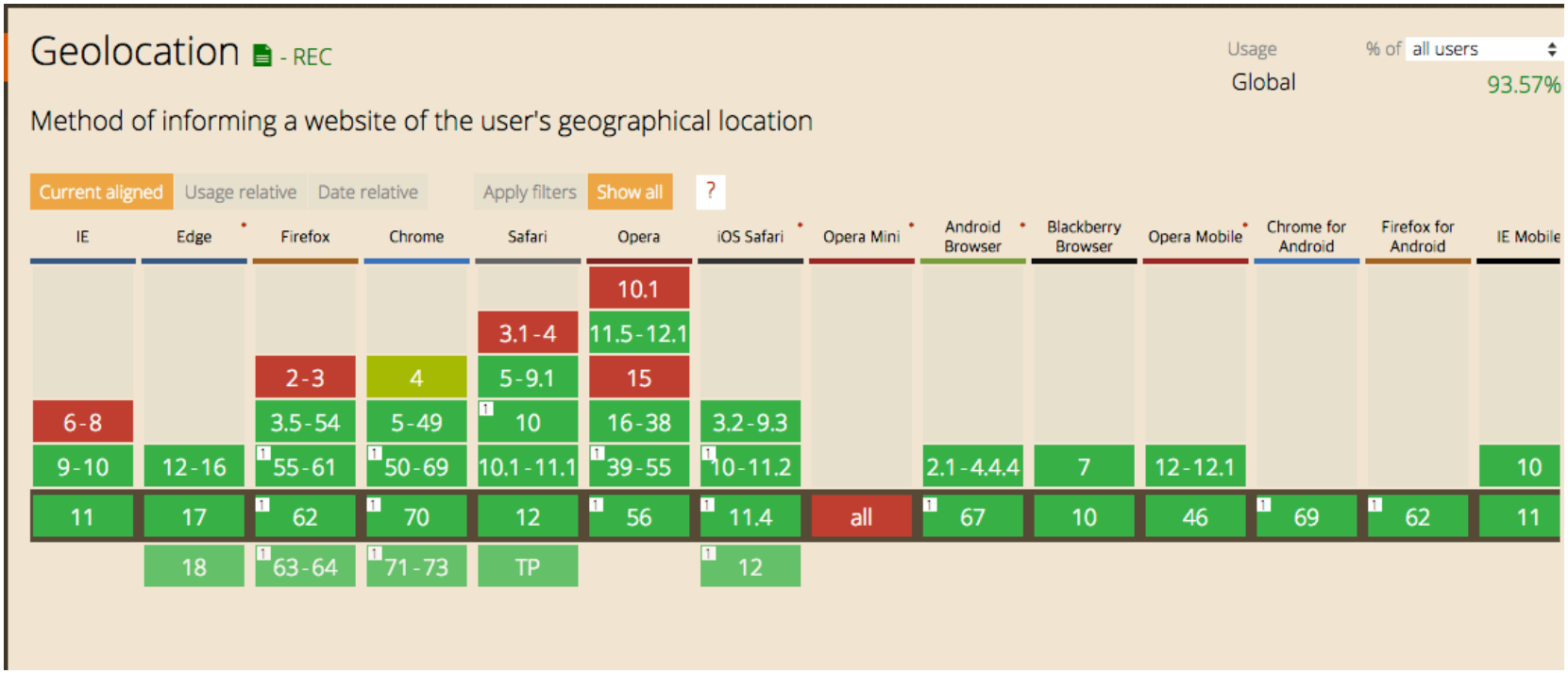
HTML5 지원여부

Browser support test : Chrome version - 70

→ http://html5test.com/

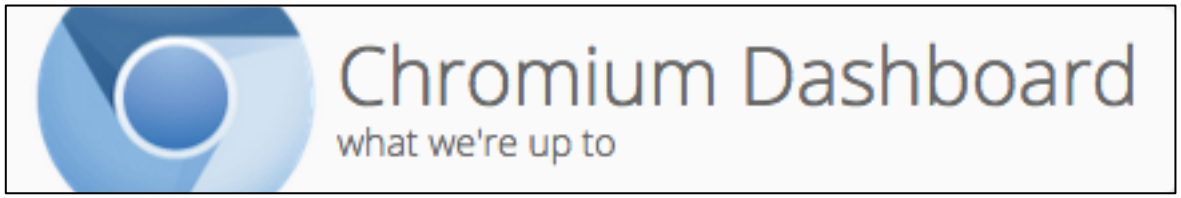
HTML5 지원여부

Browser support comparisons



→ http://caniuse.com/

브라우저 벤더별 스펙 현황



https://www.chromestatus.com/

https://status.modern.ie/

오늘 살펴볼 내용

- Multimedia: Video / Audio API
- 3D, Graphics & Effects: Canvas / Web Animation API
- Offline & Storage: File API (Reader API), Web Storage

목차

Audio/Video

개요

- 과거 웹에선 멀티미디어 컨텐츠를 재생하기 위해선 별도의 플러그인(Flash와 같은)을 필요로 했다.
- HTML5에서는 간단한 tag 작성만으로 멀티미디어 컨텐츠를 페이지에 삽입할 수 있다.
- 동영상 컨텐츠는 <video> 태그를 통해, 그리고 오디오 컨텐츠는 <audio> 태그를 통해 추가된다.

Audio 사용하기

웹에 오디오 컨텐츠 추가하기

아래와 같이 오디오 파일 정보를 기술하면, 간단히 페이지에 추가 됩니다.

```
<audio src="파일정보" controls>
        Your browser does not support the audio element.
</audio>
```

브라우저가 지원하는 포맷에 따라 여러 포맷을 지정할 수도 있습니다.

```
<audio controls>
        <source src="파일.mp3" type="audio/mpeg" />
        <source src="파일.ogg" type="audio/ogg" />
</audio>
```

Video 사용하기

웹에 비디오 컨텐츠 추가하기

아래와 같이 비디오 파일 정보를 기술하면, 간단히 페이지에 추가 됩니다.

```
<video src="파일정보" controls>
        Your browser does not support the video element.
</video>
```

브라우저가 지원하는 포맷에 따라 여러 포맷을 지정할 수도 있습니다.

```
<video controls>
  <source src="파일.mp4" type="video/mpeg" />
  <source src="파일.ogg" type='video/ogg; codecs="theora, vorbis"' />
</video>
```

Audio/Video 사용하기

속성

태그에 다음의 속성을 이용하면 다양한 옵션을 지정할 수 있습니다.

Attributes :	
preload	용량이 큰 경우, 버퍼링을 하도록 지정한다. 다음의 3가지 값 중 하나를 지정할 수 있다. - none : 버퍼링 하지 않음 - auto : 버퍼링 (최대한 가능한 만큼 버퍼링) - metadata : 메타 데이터에 대해서만 버퍼링 (첫 프레임과 비디오 정보만을 로딩한다)
autoplay	값이 설정된 경우, 자동 재생한다.
controls	값이 설정된 경우, 기본 컨트롤(재생/중지 등)이 노출된다.
width	동영상의 너비 값 (* Video에서만 사용)
height	동영상의 높이 값 (* Video에서만 사용)
loop	반복 재생 여부를 설정한다.
src	재생할 파일의 주소를 기술한다.

실습

- 간단하게 비디오와 오디오를 페이지에 삽입해 봅니다.

01_video_audio.html 파일

Audio/Video 사용하기

JavaScript API

재생 컨트롤 하기

```
const video = document.getElementsByTagName("video")[0];
if(video.paused) {
        video.play();
} else {
        video.pause();
}
```

오디오 파일은 tag 형태로 추가하지 않고 코드 상에서 바로 재생할 수 있다.

```
const audio = new Audio("파일주소");
audio.play();
```

Audio/Video 사용하기

JavaScript API

재생 컨트롤 하기

```
const audio = $("audio")[0];
audio.seekable.start(0)          // 시작시간 반환 (seconds)
audio.seekable.end(0)            // 종료시간 반환 (seconds)
audio.currentTime = nSec;        // 지정한 위치로 이동
audio.played.end(0);             // 현 시점까지 재생한 시간 반환
audio.volume += 0.1;             // 음량을 현재보다 0.1 올리기
audio.playbackRate = 2;          // 2배속으로 재생하기
```

특정 포맷을 재생할 수 있는지를 확인할 수 있다.

```
audio.canPlayType("audio/mpeg");
video.canPlayType('video/ogg; codecs="theora, vorbis"');
```

→ ‘probably’, ‘maybe’가 반환되면 지원, 빈 문자열이 반환되면 미 지원

Audio/Video 사용하기

이벤트

미디어 이벤트를 활용한 이벤트 바인딩

```
audioElement.addEventListener("ended", function(event) {
        alert("재생이 종료되었습니다.");
});

audioElement.addEventListener("volumechange"), function(event) {
        alert("음량이 변경되었습니다.");
});
```

→ Media Events : https://developer.mozilla.org/en/DOM/Media_events

→ Demo : http://www.w3.org/2010/05/video/mediaevents.html

실습

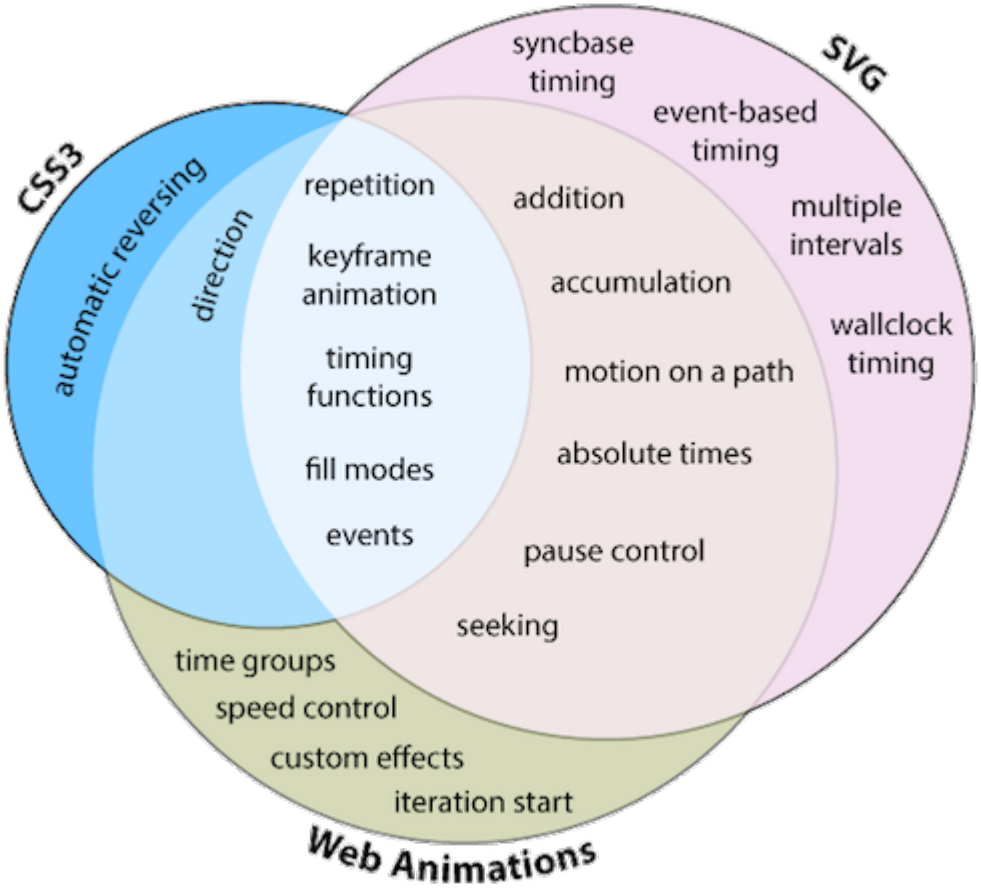
- 오디오와 비디오를 재생하고, 이벤트를 활용한 재생 컨트롤을 실습해 봅니다.

02_video_audio_control.html 파일

목차

Web Animation API

- 통합된 애니메이션 제어 모델을 제공하기 위한 자바스크립트 API



- CSS Transition
- setTimeout/setInterval
- requestAnimationFrame
- CSS Animation
- SVG Animation

Web Animation 사용하기

엘리먼트 순차적으로 애니메이션 하기

```
Animation = Element.animate(Keyframes, duration)
```

Keyframes 는 offset (0~1)에 적용될 style 속성들의 배열

```
[
  { offset: 0, transform: 'translate3d(0px,0px,0)' },
  { offset: 0.25, transform: 'translate3d(100px,0px,0)' },
  { offset: 0.5, transform: 'translate3d(100px,100px,0)' },
  { offset: 0.75, transform: 'translate3d(0,100px,0)' },
  { offset: 1, transform: 'translate3d(0,0px,0)' }
]
```

Web Animation 사용하기

엘리먼트 순차적으로 애니메이션 하기

```
document.getElementById("el").animate(
 [
   { offset: 0, transform: 'translate3d(0px,0px,0)' },
   { offset: 0.25, transform: 'translate3d(100px,0px,0)' },
   { offset: 0.5, transform: 'translate3d(100px,100px,0)' },
   { offset: 0.75, transform: 'translate3d(0,100px,0)' },
   { offset: 1, transform: 'translate3d(0,0px,0)' }
],
3000);
```

Web Animation 사용하기

애니메이션 동작 상태 제어하기 (Javascript)

```
Animation = Element.animate(Keyframes, AnimationEffectTiming)
```

Animation 객체를 통해 제어한다.

- play() : 애니메이션 시작하기
- pause() : 애니메이션 멈추기
- cancel() : 애니메이션 취소하기
- finish() : 애니메이션 동작후 멈추기
- playState : 애니메이션 상태 얻기

http://www.w3.org/TR/web-animations/#the-animation-interface

Web Animation 사용하기

애니메이션 동작 상태 제어하기 (Javascript)

```
Animation = Element.animate(Keyframes, AnimationEffectTiming)
```

AnimationEffectTiming **객체를 통해 제어한다.**

```
AnimationEffectTiming {
    "duration": 1500,          // milliseconds
    "easing": "ease-in-out", // "linear", "ease", "ease-in", "ease-out"
    "delay": 10,                  // milliseconds
    "iterations": Infinity,      // or a number
     //'normal', 'reverse', 'alternate', 'alternate-reverse' etc.
    "direction": 'alternate'
}
```

http://cubic-bezier.com/

실습

- 상자를 이동해보고, Animation객체의 메소드와 animate 옵션을 통해 애니메이션 제어 실습을 해봅니다.

03_webanimation.html 파일

목차

Canvas

개요

- 웹 페이지 내에서 벡터 그래픽을 처리할 수 있는 영역을 의미한다.
- 정의되는 canvas element들은 각각의 context(렌더링되는) 영역을 갖으며, 각 context 영역에 JavaScript로 벡터 그래픽을 그릴 수 있다.
- 2D, 3D (WebGL)을 지원한다.
- 이미지를 읽어들여 작업할 수 있으며, canvas 상에 표현된 그래픽을 외부로 export 할수 있다.

→ Spec 문서 : http://www.w3.org/TR/2dcontext/

Canvas 사용하기

Canvas 요소 추가와 기본준비

Canvas 태그를 통해 아래와 같이 간단하게 페이지에 추가할 수 있다.

```
<canvas id="canvas" width="300" height="300"></canvas>
```

브라우저가 canvas를 지원하는지는 다음과 같이 확인할 수 있다.

```
const canvas = document.getElementById("canvas");

If(canvas.getContext) {
        // Canvas 지원
} else {
        alert("이 브라우저는 canvas를 지원하지 않습니다.");
}
```

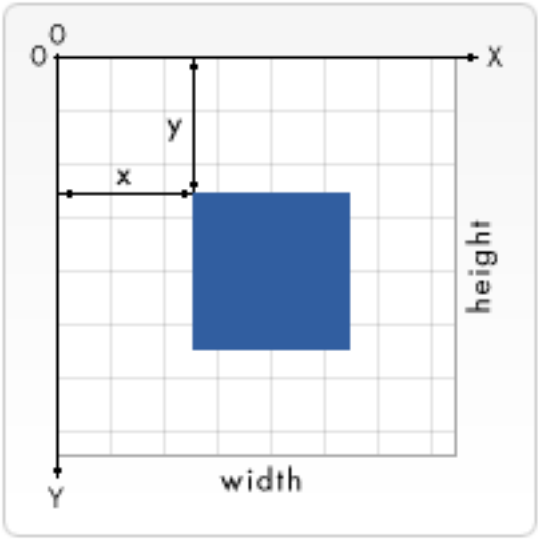
Canvas 사용하기

Canvas 요소 추가와 기본준비

이후 Canvas 요소의 컨텍스트를 다음과 같이 요청하면, canvas로 작업할 준비를 모두 마치게 된다.

```
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext("2d");
```

- Canvas내에서의 모든 작업은 canvas 영역을 기준으로 처리된다.
- 좌측 상단을 기준 좌표(x:0, y:0)로 그려지게 된다.



Canvas 사용하기

도형 그리기

Canvas는 사각형, 단 하나의 도형만이 지원된다.

```
// 채워진 사각형을 그린다.
ctx.fillRect(x, y, width, height);

// 채워지지 않은 사각형을 그린다.
ctx.strokeRect(x, y, width, height);

// 지정된 사각영역을 지운다.
ctx.clearRect(x, y, width, height);
```

실습

- 캔버스를 추가하고, 사각형을 그리는 실습을 해봅니다.

04_canvas.html

Canvas 사용하기

Path란?

- Path는 선/도형이 그려지게 되는 경로를 의미한다.
- Path는 내부적으로 합쳐져 도형을 이루는 path(직선, 호, 등)의 목록으로 저장된다.
- Path 초기화 메서드 호출시 목록이 초기화 되고 새로운 도형을 그릴 수 있도록 준비된다.

```
ctx.beginPath();                    // Path 초기화

// Path 메서드 호출
…
ctx.closePath();                    // Path 종료
ctx.stroke();                       // 저장된 path를 합쳐 그리기
ctx.fill();                         // 그려진 path 내부를 채운다.
```

Canvas 사용하기

Path 그리기

획의 시작점을 이동한다.

```
ctx.moveTo(x, y);
```

x와 y를 잇는 선을 그린다.

```
ctx.lineTo(x, y);
```

사각형을 그린다.

```
rect(x, y, width, height);
```

Canvas 사용하기

Path 그리기

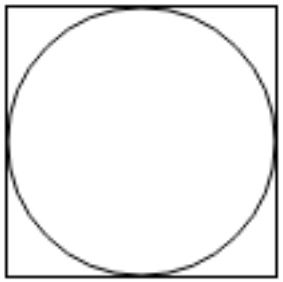
원을 그린다.

```
arc(x, y, radius, startAngle, endAngle, anticlockwise);
```

- x, y : 원 중심점의 위치
- radius : 원의 반지름
- startAngle, endAngle : 라디안의 시작점과 종료점을 의미
- anticlockwise : 원이 그려질 방향 (true-시계 반대 방향, false-시계방향)

```
arc(50, 50, 50, 0, Math.PI*2, false);
```

→ 100x100 canvas 중앙에 위치하는 원



Canvas 사용하기

스타일 - 색상 설정하기

채우기 색 또는 그라데이션 형태를 설정할 수 있다.

- fillStyle : 채우기 색 설정
- strokeStyle : 아웃라인 색 설정

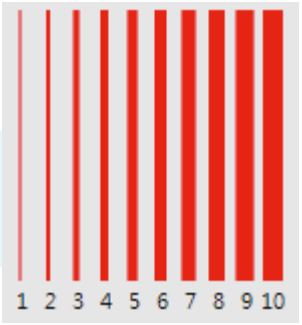
```
ctx.fillStyle = "rgb(255,165, 0)";          // RBG 값으로 설정
ctx.fillStyle = "rgba (0, 0, 200, 0.5)";    // RGB + Alpha 값으로 설정
ctx.strokeStyle = "#FFA500";                // 16진 코드로 설정
ctx.strokeStyle = "green";                  // 색상명으로 설정
```

Canvas 사용하기

스타일 - 라인 스타일 설정하기

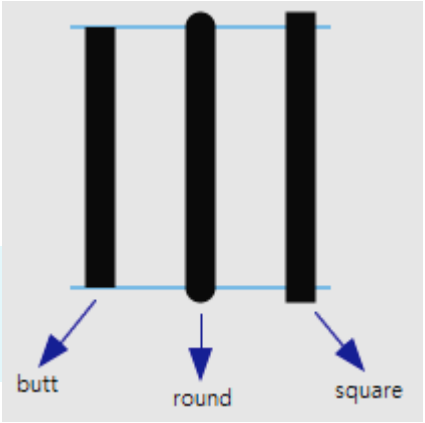
라인의 굵기 설정하기

```
ctx.lineWidth = 1
```



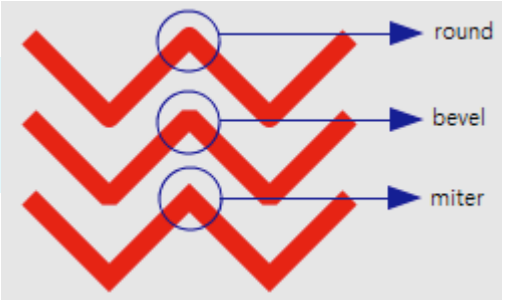
선의 시작과 끝 스타일을 설정한다.

```
ctx.lineCap = "butt | round | square";
```



선의 접점 지점에 대한 스타일을 설정한다.

```
ctx.lineJoin = "round | bevel | meter";
```



Canvas 사용하기

다양한 스타일 설정에 대한 정보 확인하기

→

https://developer.mozilla.org/ko/docs/Web/HTML/Canvas/Tutorial/Applying_styles_and_colors

실습

- 주어진 정보를 활용해 path를 그려 보도록 합니다.

05_canvas_path.html

Canvas 사용하기

이미지 삽입

이미지 파일을 canvas 영역에 삽입할 수 있다.

```
const img = new Image();
img.src = "이미지 경로";

img.onload = function() {
        ctx.drawImage(img, x, y);
}
```

Canvas 사용하기

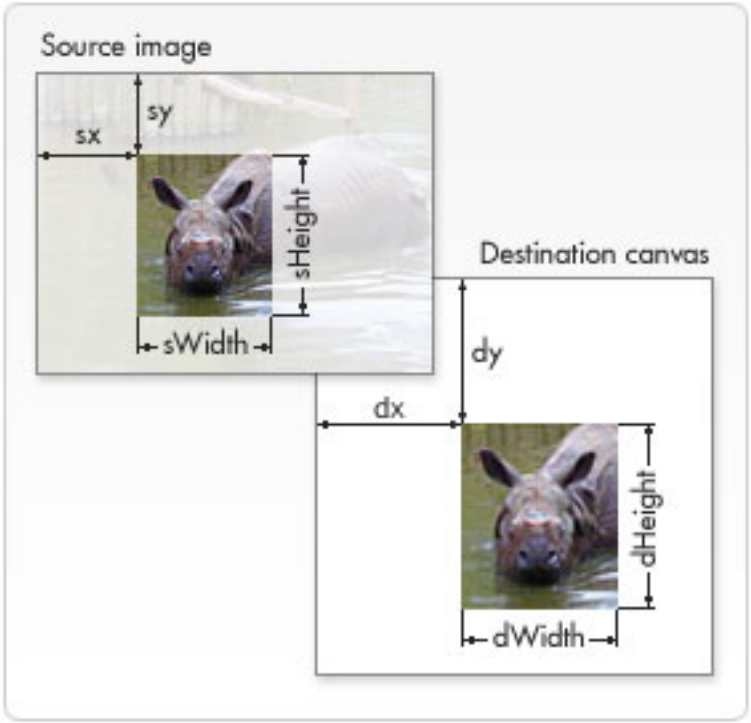
이미지 삽입

이미지 삽입시 파라미터를 활용하면 다양한 처리가 가능하다.
→ 이미지 스케일 조정하기

```
ctx.drawImage(img, x, y, width, height);
```

→ 이미지 슬라이싱

```
ctx.drawImage(img,
              sx, sy,
              sWidth, sHeight,
              dx, dy,
              dWidth, dHeight
);
```



실습

- 원본 이미지 크기를 변경하고, 특정 영역을 자르는 실습을 해봅니다.

06_canvas_image.html

Canvas 사용하기

Export

Canvas에 있는 모든 내용은 이미지 dataURL로 내보내기 될 수 있다.

→ MIME type이 지정되지 않은 경우 기본 값은 "image/png"이다.
→ 이미지가 포함되어 있는 경우, 이미지가 동일 서버에 위치해 있지 않다면 보안 이슈로 인해 보내기가 제한된다.

```
const image = canvas.toDataURL(mime_type);
→ data:mime_type;base64,iVOR…

imgElement.src = image;
```

실습

- 이전에 작성했던 스마일 얼굴 실습 파일을 사용합니다.
- exportToImage() 함수 실행 시 페이지에 canvas 에 그려진 이미지를 <img> 태그를 이용하여 추가하도록 해보자.

05_canvas_path.html

목차

File API

개요

- 웹에서 파일을 읽을 수 있는 기능을 제공한다.
- File API가 존재하지 않았던 시절에 파일을 액세스하기 위해선 서버 사이드에서의 작업을 필요로 했다.
- 파일을 업로드 하기 전에 썸네일을 구서한다던가, 파일의 종류를 검사해 파일의 업로드를 제한하는 등의 작업을 프론트-엔드에서 모두 구현이 가능해진다.

→ Spec 문서 : http://www.w3.org/TR/FileAPI/

File API

개요

- File API는 다음의 인터페이스들로 구성된다.
- **File**(단일 파일), **FileList**(여러 개의 파일) :
  파일에 대한 레퍼런스를 의미한다.
- **FileReader** :
  비동기적으로 파일을 메모리로 읽어들이는 방법을 제공한다.
- **Blob** (Binary Large Object) :
  파일 raw data를 의미하며, byte 단위로 파일을 분할할 수 있다.

File API 사용하기

지원여부 확인

브라우저에서 지원되는지 확인하기

```
If(window.File && window.FileReader && window.FileList &&
    window.Blob) {
            // File API 지원
} else {
            alert("이 브라우저는 File API를 지원하지 않습니다.");
}
```

File API 사용하기

파일 레퍼런스 얻기

<input type="file">을 통해 얻는 방법

```
<input type="file" id="files" name="files[]" multiple />

files.addEventListener("change", event => {
        const files = event.target.files;

        for (let i = 0, f; f = files[i]; i++) {
                f.size;                         // 파일 크기
                f.type;                         // 파일 종류
                f.name;                         // 파일명
                f.lastModifiedDate;     // 파일 수정일
        }
});
```

File API 사용하기

파일 레퍼런스 얻기

드래그&드롭을 통해 얻는 방법 (HTML5 드래그&드롭 이벤트 바인딩)

```
<div id="drop">파일을 이곳에 드롭하세요.</div>

drop.addEventListener("drop", event => {
        event.stopPropagation();                    // 버블링 중단
        event.preventDefault();                     // 기본 동작 막기

        const files = event.dataTransfer.files;

        for(let i=0, f; f=files[i]; i++) {
                ...
        }
});
drop.addEventListener("dragover", event => {
        event.stopPropagation();
        event.preventDefault();
        event.dataTransfer.dropEffect = "copy";
});
```

실습

- 폼을 이용한 파일 정보 출력하기
- 드래그&드롭을 통한 파일 정보 출력하기

07_file.html

FileReader API 사용하기

파일 읽기

FileReader 객체 생성 및 이벤트 바인딩

```
const reader = new FileReader();

// 파일이 읽혀들여진 이후 onload 이벤트가 발생된다.
reader.onload = function(event) {
        event.target.result;                    → 읽혀진 파일의 내용
};

// 파일 타입에 따라 메서드를 사용한다.
reader.readAsText(파일_레퍼런스, "utf-8"):      → 텍스트인 경우
reader.readAsDataURL(파일_레퍼런스):            → 이미지인 경우
```

FileReader API 사용하기

파일 읽기

파일 포맷에 따른 메서드

메서드	설명
readAsBinaryString(Blob|File)	바이너리로 읽는다.
readAsText(Blob|File, opt_encoding)	텍스트로 읽는다.
readAsDataURL(Blob|File)	dataURL로 읽는다.
readAsArrayBuffer(Blob|File)	ArrayBuffer 객체로 읽는다.

실습

- 이전 실습 파일에 다음의 기능을 추가해 봅니다.

→ 파일 타입에 따라 텍스트 또는 이미지를 읽고, 텍스트와 이미지를 console로 출력합니다.

07_file.html

목차

Web Storage

개요

- client 영역에 data를 저장하기 위해서는 cookie를 사용했지만, cookie의 한계(최대 20쌍의 값, 최대 4KB의 저장용량, request시 마다 전송되는 문제)로 인해 제한적인 용도로만 사용한다.

- Web Storage 또한 cookie와 마찬가지로 key/value쌍의 형태로 client 영역에 data를 저장할 수 있는데, 필요에 따라 세션(Session Storage)별 또는 특정 도메인(Local Storage)별 storage를 사용할 수 있으며, 기존 cookie의 한계를 극복한다.

Web Storage

Session Storage

- 세션 별로 data를 저장할 때 사용된다. 사용자가 현재 sessionStorage의 data가 저장되어 있는 브라우저의 tab을 닫거나, 또는 윈도우 자체를 닫으면 저장되어 있던 data는 모두 사라지게 된다.

- 즉, sessionStorage의 생명주기는 현재 사용하고 있는 브라우저 창의 세션 동안만 유효하다.

Web Storage

Local Storage

- 기존 cookie의 형태와 비슷하게 사용될 수 있다. 즉, 세션과는 상관없이 브라우저가 종료되어도 해당 도메인에서 설정된 localStorage data는 그대로 유지된다.

Web Storage 사용하기

기본 사용방법

Storage를 활용한 기본적인 입출력 및 삭제는 다음과 같이 할수 있다.

```
storage.setItem(key, value);
storage.getItem(key);
storage.removeItem(key);
```

또는 일반적인 객체와 같이 값 할당과 삭제 키워드를 통해 사용할 수도 있다.

```
storage.keyName = "abc";
storage.keyName;  // "abc"
delete storage.keyName;
```

Web Storage 사용하기

기타 속성 및 메서드

Storage를 활용한 기본적인 입출력 및 삭제는 다음과 같이 할수 있다.

```
// 모든 storage 데이터를 삭제한다.
storage.clear();

// 현재 저장된 key/value의 개수를 구한다.
storage.length;

// 저장된 key/value의 n번째 key를 얻는다.
// index의 순서는 역순이다. 즉, 가장 마지막에 추가된 key는 0번째 이다.
storage.key(n);
```

실습

- 간단한 값을 storage에 저장하고 확인해 봅니다.

08_localstorage.html

Reference

- http://html5doctor.com/
- http://diveintohtml5.info/

고맙습니다.