HTML Tutorial

Part 4.

Graphics, Media, APIs

1. HTML Canvas Graphics

- HTML <canvas>요소는 웹 페이지에 그래픽을 그리는 데 사용됩니다.

◆ HTML 캔버스란?

- HTML <canvas>요소는 JavaScript를 통해 즉석에서 그래픽을 그리는 데 사용됩니다.
- <canvas>요소는 유일한 그래픽 컨테이너입니다. 실제로 그래픽을 그리려면 JavaScript를 사용해야 합니다.
- Canvas에는 paths, boxes, circles, text, and adding images를 추가하는 여러 가지 방법이 있습니다.

캔버스 예:

- 캔버스는 HTML 페이지의 직사각형 영역입니다. 기본적으로 캔버스에는 테두리와 내용이 없습니다.

* 참고: 항상 id속성(스크립트에서 참조됨)을 지정 하고 width및 height속성을 지정하여 캔버스 크기를 정의합니다. 테두리를 추가하려면 style 속성을 사용하십시오 .

예시

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
Your browser does not support the HTML canvas tag.
</canvas>
```

```
</body>
</html>
```

◆ 자바스크립트 추가

- 직사각형 캔버스 영역을 생성한 후, 그리기를 하기 위해서는 자바스크립트를 추가해야 합니다.
- 여기 몇 가지 예가 있습니다.

예시 : 선 그리기

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>

</body>
```

```
</html>
```

예시 : 원 그리기

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>

</body>
</html>
```

예시 : 텍스트 그리기

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
```

```
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
</script>

</body>
</html>
```

예시 : 스트로크 텍스트

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",10,50);
</script>
```

```
</body>
</html>
```

예시 : 선형 그라데이션 그리기

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// Create gradient
var grd = ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>

</body>
</html>
```

예시 : 원형 그라데이션 그리기

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>

</body>
</html>
```

예시 : 이미지 그리기

```
<!DOCTYPE html>
<html>
```

```
<body>

<p>Image to use:</p>
<img id="scream" src="images/img_the_scream.jpg" alt="The Scream" width="220" height="277">

<p>Canvas to fill:</p>
<canvas id="myCanvas" width="250" height="300"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<p><button onclick="myCanvas()">Try it</button></p>

<script>
function myCanvas() {
  var c = document.getElementById("myCanvas");
  var ctx = c.getContext("2d");
  var img = document.getElementById("scream");
  ctx.drawImage(img,10,10);
}
</script>

</body>
</html>
```

- 더에 대한 자세한 내용은 HTML <canvas> Reference를 참고하세요.

2. HTML SVG Graphics

- SVG는 XML 형식의 벡터 기반 그래픽을 정의합니다.

◆ SVG 란 무엇입니까?

- ✓ SVG는 확장 가능한 벡터 그래픽을 나타냅니다.
- ✓ SVG는 웹용 그래픽을 정의하는 데 사용됩니다.
- ✓ SVG는 W3C 권장 사항입니다.

◆ HTML <svg> 요소

- HTML <svg>요소는 SVG 그래픽을 위한 컨테이너입니다.
- SVG에는 paths, boxes, circles, text, and graphic images를 그리는 여러 가지 방법이 있습니다.

◆ SVG Circle

예시

```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />
```

```
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

◆ SVG Rectangle

예시

```
<!DOCTYPE html>
<html>
<body>

<svg width="400" height="100">
  <rect width="400" height="100"
  style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

◆ SVG Rounded Rectangle

예시

```
<!DOCTYPE html>
<html>
<body>

<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

◆ SVG Star

예시

```
<!DOCTYPE html>
<html>
<body>

<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
Sorry, your browser does not support inline SVG.
</svg>
```

```
</body>
</html>
```

◆ SVG 로고

예시

```
<!DOCTYPE html>
<html>
<body>

<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%"
      style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%"
      style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana"
  x="50" y="86">SVG</text>
Sorry, your browser does not support inline SVG.
</svg>

</body>
```

```
</html>
```

◆ SVG와 Canvas의 차이점

- SVG는 XML에서 2D 그래픽을 설명하기 위한 언어입니다.
- Canvas는 JavaScript를 사용하여 즉석에서 2D 그래픽을 그립니다.

- SVG는 XML 기반이므로 SVG DOM 내에서 모든 요소를 사용할 수 있습니다. 요소에 대한 JavaScript 이벤트 핸들러를 연결할 수 있습니다.
- SVG에서 그려진 각 모양은 객체로 기억됩니다. SVG 개체의 속성이 변경되면 브라우저는 자동으로 모양을 다시 렌더링할 수 있습니다.

- 캔버스는 픽셀 단위로 렌더링됩니다. 캔버스에서 그래픽은 일단 그려지면 브라우저에서 잊어버립니다. 위치를 변경해야 하는 경우 그래픽으로 덮였을 수 있는 개체를 포함하여 전체 장면을 다시 그려야 합니다.

◆ 캔버스와 SVG의 비교

- 아래 표는 Canvas와 SVG 간의 몇 가지 중요한 차이점을 보여줍니다.

Canvas	SVG
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

참조 : https://ourcodeworld.com/articles/read/308/top-15-best-open-source-javascript-game-engines

3. HTML Multimedia

- 웹상의 멀티미디어는 사운드, 음악, 비디오, 영화 및 애니메이션입니다.

◆ 멀티미디어란?

- 멀티미디어는 다양한 형식으로 제공됩니다. 이미지, 음악, 사운드, 비디오, 레코드, 영화, 애니메이션 등과 같이 듣거나 볼 수 있는 거의 모든 것이 될 수 있습니다.
- 웹 페이지에는 다양한 유형과 형식의 멀티미디어 요소가 포함되어 있는 경우가 많습니다.

◆ 브라우저 지원

- 최초의 웹 브라우저는 단일 색상의 단일 글꼴로 제한되는 텍스트만 지원했습니다.
- 나중에 색상, 글꼴, 이미지 및 멀티미디어를 지원하는 브라우저가 등장했습니다!

◆ 멀티미디어 형식

- 멀티미디어 요소(예: 오디오 또는 비디오)는 미디어 파일에 저장됩니다.
- 파일 유형을 검색하는 가장 일반적인 방법은 파일 확장자를 확인하는 것입니다.
- 멀티미디어 파일에는 .wav, .mp3, .mp4, .mpg, .wmv 및 .avi와 같은 형식과 확장자가 있습니다.

◆ 일반적인 비디오 형식

- 비디오 형식에는 많은 비디오 format이 있습니다.

- MP4, WebM 및 Ogg format은 HTML에서 지원됩니다.
- YouTube에서는 MP4 format을 권장합니다.

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Not supported anymore in HTML.
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers.
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. Does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML.
WebM	.webm	WebM. Developed by Mozilla, Opera, Adobe, and Google. Supported by HTML.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Commonly used in video cameras and TV hardware. Supported by all browsers and recommended by YouTube.

- 참고: MP4, WebM 및 Ogg 비디오만 HTML 표준에서 지원됩니다.

◆ 일반적인 오디오 형식

- MP3는 압축된 녹음 음악에 가장 적합한 형식입니다. MP3라는 용어는 디지털 음악과 동의어가 되었습니다.

- 웹사이트가 녹음된 음악에 관한 것이라면 MP3가 적합합니다.

Format	File	Description
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers.
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Plays well on Windows computers, but not in web browsers.
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers.
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML.
MP3	.mp3	MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers.
MP4	.mp4	MP4 is a video format, but can also be used for audio. Supported by all browsers.

참고: MP3, WAV 및 Ogg 오디오만 HTML 표준에서 지원됩니다.

4. HTML Video

- HTML <video>요소는 웹 페이지에 비디오를 표시하는 데 사용됩니다.

예시

Big Buck Bunny의 의례 :

HTML <비디오> 요소

HTML로 비디오를 표시하려면 다음 <video>요소를 사용하십시오 .

예시

```
<video width="320" height="240" controls>
  <source src="images/movie.mp4" type="video/mp4">
  <source src="images/movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

◆ 작동 원리

- controls 속성은 재생, 일시 정지, 볼륨처럼, 비디오 컨트롤을 추가합니다.
- 항상 width 및 height 속성을 포함하는 것이 좋습니다. 높이와 너비를 설정하지 않으면 비디오가 로드되는 동안 페이지가 깜박일 수 있습니다.
- <source> 요소를 사용하면 브라우저가 선택할 수 있는 대체 비디오 파일을 지정할 수 있습니다. 브라우저는 인식된 첫 번째 형식을 사용합니다.
- <video>와 </video> 태그 사이의 텍스트는 <video> 요소를 지원하지 않는 브라우저에서만 표시됩니다.

◆ HTML <비디오> 자동 재생

- 비디오를 자동으로 시작하려면 다음 autoplay속성을 사용하십시오 .

예시

```
<video width="320" height="240" autoplay>
  <source src="images/movie_01.mp4" type="video/mp4">
  <source src="images/movie_01.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

- 자동 재생 후 음소거 추가: 비디오 재생을 자동으로 시작합니다(음소거됨).

예시

```
<video width="320" height="240" autoplay muted>
  <source src="images/movie.mp4" type="video/mp4">
  <source src="images/movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

◆ HTML Video - Methods, Properties, and Events

- HTML DOM은 <video>요소에 대한 메서드, 속성 및 이벤트를 정의합니다 .
- 이를 통해 비디오를 로드, 재생 및 일시 중지하고 재생 시간과 볼륨을 설정할 수 있습니다.

- 비디오가 재생되기 시작하거나 일시중지될 때 등을 알릴 수 있는 DOM 이벤트도 있습니다.

예: 자바스크립트 사용

```
<!DOCTYPE html>
<html>
<body>

<div style="text-align:center">
  <button onclick="playPause()">Play/Pause</button>
  <button onclick="makeBig()">Big</button>
  <button onclick="makeSmall()">Small</button>
  <button onclick="makeNormal()">Normal</button>
  <br><br>
  <video id="video1" width="420">
    <source src="images/movie.mp4" type="video/mp4">
    <source src="images/movie.ogg" type="video/ogg">
    Your browser does not support HTML video.
  </video>
</div>

<script>
var myVideo = document.getElementById("video1");

function playPause() {
  if (myVideo.paused)
    myVideo.play();
  else
    myVideo.pause();
```

```
}

function makeBig() {
    myVideo.width = 560;
}

function makeSmall() {
    myVideo.width = 320;
}

function makeNormal() {
    myVideo.width = 420;
}
</script>

<p>Video courtesy of <a href="https://www.bigbuckbunny.org/" target="_blank">Big Buck Bunny</a>.</p>

</body>
</html>
```

5. HTML Audio

- HTML <audio>요소는 웹 페이지에서 오디오 파일을 재생하는 데 사용됩니다.

◆ HTML <audio> 요소

HTML에서 오디오 파일을 재생하려면 다음 <audio>요소를 사용하십시오 .

예시

```
<!DOCTYPE html>
<html>
<body>

<audio controls>
  <source src="images/horse.ogg" type="audio/ogg">
  <source src="images/horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>

</body>
</html>
```

◆ HTML 오디오 - 작동 원리

- controls 속성은 오디오 컨트롤, 재생, 일시 정지 및 볼륨을 추가합니다.
- <source>요소는 브라우저에서 선택할 수 있습니다 다른 오디오 파일을 지정할 수 있습니다. 브라우저는 처음으로 인식된 형식을 사용합니다.
- <audio>와 </audio>태그 사이의 텍스트는 해당 <audio>요소를 지원하지 않는 브라우저에서만 표시됩니다 .

◆ HTML <오디오> 자동 재생

- 오디오 파일을 자동으로 시작하려면 다음 autoplay 속성을 사용하십시오 .

예시

```
<audio controls autoplay>
  <source src="images/horse.ogg" type="audio/ogg">
  <source src="images/horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

- 오디오 파일이 자동으로 음소거된 상태로 자동재생 하기 위해서는 autoplay 뒤에 muted 속성을 추가하세요.

예시

```
<audio controls autoplay muted>
  <source src="images/horse.ogg" type="audio/ogg">
  <source src="images/horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

◆ HTML 오디오 - 미디어 유형

File Format	Media Type
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

◆ HTML 오디오 - 메서드, 속성 및 이벤트

- HTML DOM은 <audio>요소에 대한 메서드, 속성 및 이벤트를 정의합니다 .
- 이를 통해 오디오를 로드, 재생 및 일시 중지하고 지속 시간과 볼륨을 설정할 수 있습니다.
- 오디오가 재생되기 시작하거나 일시 중지될 때 등을 알릴 수 있는 DOM 이벤트도 있습니다.

6. HTML YouTube Videos

- HTML로 동영상을 재생하는 가장 쉬운 방법은 YouTube를 사용하는 것입니다.

◆ 비디오 형식으로 어려움을 겪고 계십니까?

- 비디오를 다른 형식으로 변환하는 것은 어렵고 시간이 많이 소요될 수 있습니다.
- 더 쉬운 해결책은 YouTube가 웹 페이지에서 동영상을 재생하도록 하는 것입니다.

◆ YouTube 동영상 ID

- YouTube는 동영상을 저장(또는 재생)할 때 ID(예: tgbNymZ7vqY)를 표시합니다.
- 이 ID를 사용하고 HTML 코드에서 동영상을 참조할 수 있습니다.

◆ HTML로 YouTube 동영상 재생

- 웹 페이지에서 비디오를 재생하려면 다음을 수행하십시오.

  - YouTube에 동영상 업로드
  - 비디오 ID를 기록해 둡니다.
  - 웹 페이지에서 <iframe>요소 정의
  - video URL을 지정하는 src 속성 정의
  - 플레이어의 크기를 지정하는 width, height 속성 정의
  - URL에 다른 매개변수를 추가합니다(아래 참조).

예시

```
<!DOCTYPE html>
<html>
<body>

<iframe width="420" height="345" src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>

</body>
</html>
```

◆ YouTube 자동재생 + 음소거

- 사용자가 페이지를 방문할 때 autoplay=1을 YouTube URL에 추가하여 동영상을 자동으로 재생하도록 할 수 있습니다. 그러나 동영상을 자동으로 시작하는 것은 방문자에게 성가신 일입니다!
- 동영상이 자동으로 재생되도록 하려면(음소거됨) autoplay=1뒤에 mute=1을 추가 하세요.

YouTube - 자동 재생 + 음소거

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">
</iframe>
```

◆ 유튜브 재생목록

- 재생할 비디오의 쉼표로 구분된 목록(원래 URL 외에).

◆ YouTube Loop

- 비디오가 영원히 반복되도록 할려면 loop=1을 추가하십시오 .

  - 값 0(기본값): 비디오가 한 번만 재생됩니다.
  - 값 1: 비디오가 (영원히) 반복됩니다.

YouTube - 루프

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1">
</iframe>
```

◆ YouTube 컨트롤

- 비디오 플레이어에서 컨트롤을 표시하지 않으려면 controls=0를 추가하십시오 .

  - 값 0: 플레이어 컨트롤이 표시되지 않습니다.
  - 값 1(기본값): 플레이어가 디스플레이를 제어합니다.

YouTube - 컨트롤

```
<iframe width="420" height="315"
```

```
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
```

이후 내용은

JavaScript를 어느 정도 이해 해야만 가능한 학습과정입니다.

따라서,

JavaScript 과정을 이수 한 후 학습하도록 하겠습니다.

7. Drag and Drop API

- HTML에서는 모든 요소를 끌어다 놓을 수 있습니다.

◆ Drag and Drop

- 드래그 앤 드롭은 매우 일반적인 기능입니다. 개체를 "잡아" 다른 위치로 끌어다 놓습니다.
- 아래 예는 간단한 드래그 앤 드롭 예입니다.

예시

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1 {
  width: 350px;
  height: 70px;
  padding: 10px;
  border: 1px solid #aaaaaa;
}
</style>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}
```

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

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<p>Drag the image into the rectangle:</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<img id="drag1" src="images/naver.png" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>
</html>
```

- 복잡해 보일 수 있지만 끌어서 놓기 이벤트의 모든 다른 부분을 살펴보겠습니다.

◆ 요소를 드래그 가능하게 만들기

우선: 요소를 드래그 가능하게 만들려면 draggable 속성을 true로 설정하십시오 .

```
<img draggable="true">
```

◆ 드래그할 대상 - ondragstart 및 setData()

- 그런 다음 요소를 끌 때 발생해야 하는 작업을 지정합니다.
- 위의 예에서 ondragstart 속성은 드래그할 데이터를 지정하는 함수 drag(event)를 호출합니다.
- dataTransfer.setData()메서드는 끌어온 데이터의 데이터 유형과 값을 설정합니다.

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

이 경우 데이터 유형은 "text"이고 값은 드래그 가능한 요소의 id("drag1")입니다.

◆ 드롭 위치 - ondragover

- ondragover 이벤트는 드래그된 데이터를 drop할 수 있는 위치를 지정합니다.
- 기본적으로 data/elements는 다른 요소에 drop할 수 없습니다. drop을 허용하려면 요소의 기본 처리(default handling)을 방지(prevent)해야 합니다.

- 이것은 ondragover에 대해 event.preventDefault() 이벤트에 대한 메서드를 호출하여 수행됩니다 .
-

```
event.preventDefault()
```

◆ 드롭 하기 - ondrop

- 드래그한 데이터를 드롭하면 드롭 이벤트가 발생합니다.
- 위의 예에서 ondrop 속성은 함수 drop(event)을 호출합니다.

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

코드 설명:

- 데이터의 브라우저 기본 처리를 방지하려면 preventDefault()를 호출하십시오(기본값은 드롭 시 링크로 열려 있음).
- dataTransfer.getData() 메서드로 드래그한 데이터를 가져옵니다. 이 메서드는 setData() 메서드에서 동일한 유형으로 설정된 모든 데이터를 반환합니다.
- 끌어온 데이터는 끌어온 요소의 ID("drag1")입니다.
- 드래그한 요소를 드롭 요소에 추가

더 많은 예

예시

두 요소 사이에서 이미지를 앞뒤로 드래그(앤드롭)하는 방법:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1, #div2 {
  float: left;
  width: 100px;
  height: 35px;
  margin: 10px;
  padding: 10px;
  border: 1px solid black;
}
</style>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

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

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<h2>Drag and Drop</h2>
<p>Drag the image back and forth between the two div elements.</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">
  <img  src="images/naver_01.png"  draggable="true"  ondragstart="drag(event)"  id="drag1"  width="88"
height="31">
</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

</body>
</html>
```

8. HTML Web Storage API

- HTML 웹 스토리지; 쿠키보다 낫다.

◆ HTML 웹 스토리지란 무엇입니까?

- 웹 스토리지를 사용하면 웹 애플리케이션이 사용자의 브라우저 내에서 로컬로 데이터를 저장할 수 있습니다.
- HTML5 이전에 애플리케이션 데이터는 쿠키에 저장되고, 모든 server 요청에 포함되었습니다. 웹 스토리지는 더 안전하며 웹 사이트 성능에 영향을 미치지 않고 대량의 데이터를 로컬에 저장할 수 있습니다.
- 쿠키와 달리 저장 제한은 훨씬 더 크며(최소 5MB) 정보는 서버로 전송되지 않습니다.
- 웹 스토리지는 origin 당(도메인 및 프로토콜 당)입니다. 하나의 origin에서 모든 페이지는 동일한 데이터를 저장하고 액세스할 수 있습니다.

◆ HTML 웹 스토리지 객체

- HTML 웹 스토리지는 클라이언트에 데이터를 저장하기 위한 두 가지 객체를 제공합니다.

  ✓ window.localStorage - 만료 날짜 없이 데이터를 저장합니다.
  ✓ window.sessionStorage - 한 세션에 대한 데이터를 저장합니다(브라우저 탭을 닫으면 데이터가 손실됨)

- 웹 저장소를 사용하기 전에 localStorage 및 sessionStorage에 대한 브라우저 지원을 확인하십시오.

```
if (typeof(Storage) !== "undefined") {
  // Code for localStorage/sessionStorage.
```

```
} else {
  // Sorry! No Web Storage support..
}
```

◆ localStorage 객체

- localStorage 객체는 만료 날짜 없이 데이터를 저장합니다. 데이터는 브라우저를 닫아도 삭제되지 않으며 다음 날, 주 또는 연도에 사용할 수 있습니다.

예시

```
<!DOCTYPE html>
<html>
<body>

<div id="result"></div>

<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
  // Store
  localStorage.setItem("lastname", "Smith");
  // Retrieve
  document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

```
} else {
  document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage...";
}
</script>

</body>
</html>
```

예 설명:

- name="lastname" 및 value="Smith"를 사용하여 localStorage 이름/값 쌍을 만듭니다.
- "lastname"의 값을 검색하고 id="result"인 요소에 삽입합니다.

위의 예는 다음과 같이 작성할 수도 있습니다.

```
// Store
localStorage.lastname = "Smith";
// Retrieve
document.getElementById("result").innerHTML = localStorage.lastname;
```

"lastname" localStorage 항목을 제거하는 구문은 다음과 같습니다.

```
localStorage.removeItem("lastname");
```

- 참고: 이름/값 쌍은 항상 문자열로 저장됩니다. 필요할 때 다른 형식으로 변환하는 것을 잊지 마십시오!

- 다음 예제에서는 사용자가 버튼을 클릭한 횟수를 계산합니다. 이 코드에서 값 문자열은 카운터를 늘릴 수 있도록 숫자로 변환됩니다.

예시

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
  if (typeof(Storage) !== "undefined") {
    if (localStorage.clickcount) {
      localStorage.clickcount = Number(localStorage.clickcount)+1;
    } else {
      localStorage.clickcount = 1;
    }
    document.getElementById("result").innerHTML = "You have clicked the button " +
localStorage.clickcount + " time(s).";
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support web
storage...";
  }
```

```
}
</script>
</head>
<body>

<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter will continue to count (is not reset).</p>

</body>
</html>
```

◆ sessionStorage 객체

- sessionStorage 개체는 한 세션에 대한 데이터만 저장한다는 점을 제외하고 localStorage 개체와 동일합니다. 사용자가 특정 브라우저 탭을 닫으면 데이터가 삭제됩니다.
- 다음은 사용자가 현재 세션에서 단추를 클릭한 횟수를 카운트하는 예제입니다.

예시

```
<!DOCTYPE html>
<html>
<head>
<script>
```

```
function clickCounter() {
  if (typeof(Storage) !== "undefined") {
    if (sessionStorage.clickcount) {
      sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
    } else {
      sessionStorage.clickcount = 1;
    }
    document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support web
storage...";
  }
}
</script>
</head>
<body>

<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter is reset.</p>

</body>
</html>
```

9. HTML Web Workers API

- Web Worker는 페이지 성능에 영향을 주지 않고 백그라운드에서 실행되는 JavaScript입니다.

◆ Web Worker 란?

- HTML 페이지에서 스크립트를 실행할 때 스크립트가 완료될 때까지 페이지가 응답하지 않습니다.
- Web Worker는 페이지 성능에 영향을 주지 않고 다른 스크립트와 독립적으로 백그라운드에서 실행되는 JavaScript입니다.
- Web Worker가 백그라운드에서 실행되는 동안 클릭, 항목 선택 등 원하는 작업을 계속할 수 있습니다.

◆ HTML Web Worker 예제

- 아래 예는 백그라운드에서 숫자를 세는 간단한 Web Worker를 만듭니다.

예시

```
<!DOCTYPE html>
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support Web Workers.</p>
```

```
<script>
var w;

function startWorker() {
  if(typeof(Worker) !== "undefined") {
    if(typeof(w) == "undefined") {
      w = new Worker("demo_workers.js");
    }
    w.onmessage = function(event) {
      document.getElementById("result").innerHTML = event.data;
    };
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support Web
Workers...";
  }
}

function stopWorker() {
  w.terminate();
  w = undefined;
}
</script>

</body>
</html>
```

◆ Web Worker 지원 확인

- Web Worker를 생성하기 전에 사용자의 브라우저가 Web Worker를 지원하는지 확인하십시오.

```
if (typeof(Worker) !== "undefined") {
  // Yes! Web worker support!
  // Some code.....
} else {
  // Sorry! No Web Worker support..
}
```

◆ Web Worker 파일 만들기

- 이제 외부 JavaScript에서 Web Worker를 만들어 보겠습니다.

- 여기에서 계산하는 스크립트를 만듭니다. 스크립트는 "demo_workers.js" 파일에 저장됩니다.

```
var i = 0;

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

```
}

timedCount();
```

- 위 코드의 중요한 부분은 postMessage() 메서드입니다. 이는 HTML 페이지에 메시지를 다시 게시하는 데 사용되는 메서드입니다.
- 참고: 일반적으로 Web Worker는 이러한 간단한 스크립트가 아니라 CPU를 많이 사용하는 작업에 사용됩니다.

◆ Web Worker 개체 만들기

이제 Web Worker 파일이 있으므로 HTML 페이지에서 호출해야 합니다.

다음 줄은 작업자가 이미 존재하는지 확인합니다. 없으면 새 Web Worker 개체를 만들고 "demo_workers.js"에서 코드를 실행합니다.

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

- 그런 다음 Web Worker로부터 메시지를 보내고 받을 수 있습니다.
- Web Worker에 "onmessage" 이벤트 리스너를 추가합니다.

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

```
};
```

- Web Worker가 메시지를 게시하면 event listener 내의 코드가 실행됩니다. Web Worker의 데이터는 event.data에 저장됩니다.

◆ Web Worker 종료

- Web Worker 개체가 생성되면 종료될 때까지 (외부 스크립트가 완료된 후에도) 메시지를 계속 수신합니다.
- Web Worker를 종료하고 브라우저/컴퓨터 리소스를 해제하려면 다음 terminate() 메서드를 사용하세요 .

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

◆ 웹 워커 재사용

작업자 변수를 undefined로 설정하면 종료된 후 코드를 재사용할 수 있습니다.

```
w = undefined;
```

전체 Web Worker 예제 코드

우리는 이미 .js 파일에서 작업자 코드를 보았습니다. 다음은 HTML 페이지의 코드입니다.

예시

```
<!DOCTYPE html>
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support Web Workers.</p>

<script>
var w;

function startWorker() {
  if(typeof(Worker) !== "undefined") {
    if(typeof(w) == "undefined") {
      w = new Worker("demo_workers.js");
    }
    w.onmessage = function(event) {
      document.getElementById("result").innerHTML = event.data;
    };
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Workers...";
```

```
  }
}

function stopWorker() {
  w.terminate();
  w = undefined;
}
</script>

</body>
</html>
```

◆ Web Worker와 DOM

- Web Worker는 외부 파일에 있으므로 다음 JavaScript 개체에 액세스할 수 없습니다.

  - ✓ 창 개체
  - ✓ 문서 객체
  - ✓ 상위 개체

10. HTML SSE API

- SSE(Server-Sent Events)를 사용하면 웹 페이지가 서버에서 업데이트를 가져올 수 있습니다.

◆ Server-Sent Events - One Way Messaging

- 서버에서 보낸 이벤트는 웹 페이지가 서버에서 자동으로 업데이트를 받는 경우입니다.
- 이것은 이전에도 가능했지만 웹 페이지에서 사용 가능한 업데이트가 있는지 물어야 합니다. 서버에서 보낸 이벤트를 사용하면 업데이트가 자동으로 제공됩니다.

- 예: Facebook/Twitter 업데이트, 주가 업데이트, 뉴스 피드, 스포츠 결과 등

◆ Receive Server-Sent Event Notifications

- EventSource 개체는 서버에서 보낸 이벤트 알림을 수신하는 데 사용됩니다.

예시

```
<!DOCTYPE html>
<html>
<body>

<h1>Getting server updates</h1>
<div id="result"></div>
```

```
<script>
if(typeof(EventSource) !== "undefined") {
  var source = new EventSource("demo_sse.php");
  source.onmessage = function(event) {
    document.getElementById("result").innerHTML += event.data + "<br>";
  };
} else {
  document.getElementById("result").innerHTML = "Sorry, your browser does not support server-sent
events...";
}
</script>

</body>
</html>
```

예 설명 :

- 새 EventSource 개체를 만들고 업데이트를 보내는 페이지의 URL을 지정합니다(이 예에서는 "demo_sse.html").
- 업데이트가 수신될 때마다 onmessage 이벤트가 발생합니다.
- onmessage 이벤트가 발생하면 수신된 데이터를 id="result"인 요소에 넣습니다.

◆ 서버 전송 이벤트 지원 확인

- 위의 tryit 예제에는 서버에서 보낸 이벤트에 대한 브라우저 지원을 확인하기 위한 몇 가지 추가 코드 줄이 있습니다.

```
if(typeof(EventSource) !== "undefined") {
  // Yes! Server-sent events support!
  // Some code.....
} else {
  // Sorry! No server-sent events support..
}
```

◆ 서버 측 코드 예

- 위의 예가 작동하려면 데이터 업데이트를 보낼 수 있는 서버가 필요합니다.
- 여기서는 php나 asp를 예를 듭니다 (서버측 프로그램은 Web 개발 프로젝트 과정에서 배울 수 있습니다.)

- 서버측 이벤트 스트림 구문은 간단합니다. "Content-Type" 헤더를 "text/event-stream"으로 설정합니다. 이제 이벤트 스트림 전송을 시작할 수 있습니다.

PHP 코드(demo_sse.php):

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

ASP(VB)의 코드(demo_sse.asp):

```
<%
Response.ContentType = "text/event-stream"
Response.Expires = -1
Response.Write("data: The server time is: " & now())
```

```
Response.Flush()
%>
```

코드 설명:

- "Content-Type" 헤더를 "text/event-stream"으로 설정합니다.
- 페이지가 캐시되지 않도록 지정
- 보낼 데이터 출력(항상 "data: "로 시작)
- 출력 데이터를 웹 페이지로 다시 플러시

◆ The EventSource Object

- 위의 예에서는 onmessage 이벤트를 사용하여 메시지를 가져왔습니다. 그러나 다른 이벤트도 사용할 수 있습니다.

Events	Description
onopen	When a connection to the server is opened
onmessage	When a message is received
onerror	When an error occurs