

Лекція 8. МУЛЬТИМЕДІА

8.1. Графіка HTML

https://www.w3schools.com/html/html5_canvas.asp

8.1.1. HTML Canvas Graphics

Елемент HTML `<canvas>` використовується для малювання простих графічних об'єктів та додання їх на сторінки сайтів.








Ця графіка створена за допомогою елемента `<canvas>`. Показано чотири елементи: Червоний прямокутник, градієнтний прямокутник, прямокутник з градієнтним переходом кількох кольорів та текст зі спеціальним накресленням та багатьма кольорами.

Елемент HTML `<canvas>` використовується для малювання графіки безпосередньо у HTML документі за допомогою JavaScript.

Елемент `<canvas>` - це лише контейнер для графічних об'єктів. Для того, щоб малювати графіку, потрібно використовувати JavaScript.

Canvas має декілька методів для малювання напрямків (ліній), прямокутників, кіл, тексту, а також додавання зображень.

Підтримка браузерами даного елемента показана у таблиці. Номера в таблиці вказують першу версію браузера, яка повністю підтримує елемент `<canvas>`.

Element					
<code><canvas></code>	4.0	9.0	2.0	3.1	9.0

Приклади застосування елемента `<canvas>`

Елемент `<canvas>` є прямокутною областю на HTML-сторінці. Без додання інших елементів коду такий об'єкт не має ані рамки ані вмісту.

HTML код в даному випадку має такий вигляд:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Зауваження: Завжди вказуйте атрибут `id` (який потім слід вказати у JavaScript), а атрибути ширини `width` і висоти `height` визначають розмір елемента `<canvas>`. Щоб додати рамку, скористайтеся атрибутом стилю `style`.

Це приклад порожнього початкового елемента `<canvas>`:

```
<!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>
```



HTML код в даному випадку має такий вигляд:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
</canvas>
```

Додавання JavaScript

Після створення прямокутної області елемента `<canvas>`, для малювання потрібно додати JavaScript.

Наведемо декілька прикладів.

Малювання лінії

```

<!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>

```



Додамо ще одну лінію з іншими координатами початку.

```

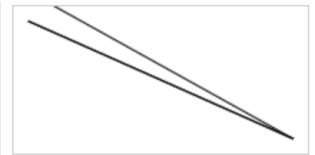
<!DOCTYPE html>
<html>
<body>

<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(10,10);
ctx.lineTo(190,90);
ctx.stroke();
ctx.moveTo(10,-10);
ctx.lineTo(190,90);
ctx.stroke();
</script>

</body>
</html>

```



Малювання кола

```

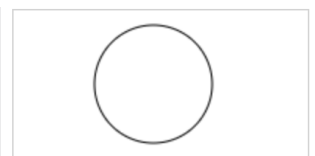
<!DOCTYPE html>
<html>
<body>

<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>

```



В коді застосовано такі параметри ctx.arc():

Зсув за горизонталлю = 95

Зсув за вертикаллю = 50

Діаметр = 40

Величина сектора, який візуалізується = 0

```
<!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(95,50,20,30,2*Math.PI);
ctx.stroke();
</script>

</body>
</html>
```



Написання тексту

```
<!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>
```

Hello World

Написання тексту особливим стилем

```
<!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>
```

Hello World

Градiєнтна заливка кольором

```
<!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>
```



Повторювана градиєнтна заливка кольором

```
<!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>
```



Додавання зображень до області елемента `<canvas>`

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

<p>Image to use:</p>


<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>

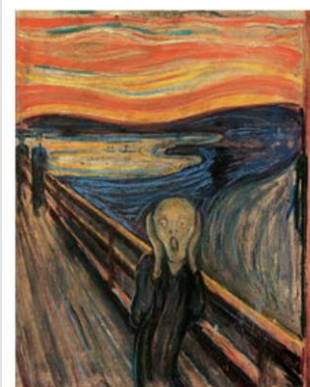
<p><button onclick="myCanvas()">Try it</button></p>

<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>
```



Canvas to fill:



Навчальні матеріали з HTML Canvas за посиланням

https://www.w3schools.com/graphics/canvas_intro.asp

8.1.2. HTML SVG графіка





SVG (Scalable Vector Graphics) – векторна графіка, яка передбачає можливості масштабування. SVG застосовується для завдання графіки у веб-застосунках. SVG є рекомендованою за нотацією W3C.

HTML елемент `<svg>`

Елемент HTML `<svg>` є контейнером для графічних об'єктів SVG.

SVG має декілька методів для малювання ліній, прямокутників, кіл, тексту та графічних зображень.

Підтримка браузером даного елемента показана у таблиці. Номера в таблиці вказують першу версію браузера, яка повністю підтримує елемент `<svg>`.

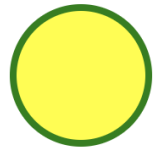
Element					
<code><svg></code>	4.0	9.0	3.0	3.2	10.1

Кола SVG

```
<!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>
```



Прямокутники SVG

```
<!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>
```

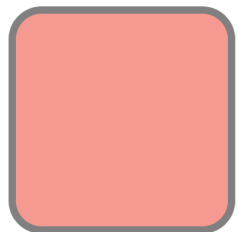


Прямокутники SVG з округленими кутами

```
<!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>
```

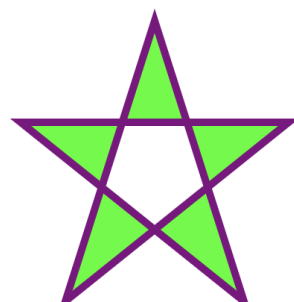


Зірка SVG

```
<!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>
```



Створення логотипу SVG

```
<!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>
```



Різниця між елементами SVG та Canvas

- SVG — мова для опису 2D-графіки в XML.
- Canvas малює 2D графіку, на льоту (з JavaScript).
- SVG заснований на XML, це означає, що кожен елемент доступний в рамках SVG DOM. Ви можете долучити до цього елемента обробники подій JavaScript.
- У SVG кожна намальована форма запам'ятовується як об'єкт. Якщо атрибути об'єкта SVG змінені, браузер може автоматично повторно відтворити форму (перемальовати її згідно зі змінами атрибутів).
- Canvas відтворюватиметься піксель за пікселем кожного разу під час візуалізації сторінки. Щойно графіка Canvas була відтворена браузер "забуває" її. Якщо положення такого графічного об'єкта має бути змінено, вся сторінка має бути перемальована, включаючи будь-які об'єкти, які могли бути покриті даним графічним об'єктом.

Наведена нижче таблиця містить важливі відмінності між елементами Canvas та SVG:

Canvas	SVG
<ul style="list-style-type: none">• Залежить від розрішення екрана• Відсутність підтримки обробників подій• Слабкі можливості відтворення тексту	<ul style="list-style-type: none">• Не залежить від розрішення екрана• Наявність підтримки обробників подій• Найкраще підходить для програм з великими областями

- Можна зберегти отримане зображення у форматі .png або .jpg
- Добре підходить для графічно-інтенсивних ігор
- візуалізації графіки (Google Maps)
- Через складність графічні об'єкти візуалізуються повільно
- Не підходить для ігрових застосунків

Навчальні матеріали з HTML Canvas за посиланням:

https://www.w3schools.com/graphics/svg_intro.asp

8.2. HTML Multimedia

Мультимедіа в Інтернеті — це звук, музика, відео, фільми та анімації.

Мультимедіа поставляється в багатьох різних форматах. Це може бути майже все, що ви можете почути або побачити, наприклад, зображення, музика, звук, відео, записи, фільми, анімації та інше.

Веб-сторінки часто містять мультимедійні елементи різних типів і форматів.

Перші веб-браузери мали підтримку лише тексту, обмежену єдиним за накресленням шрифтом одного кольору. Пізніше з'явилися браузері з підтримкою кольорів, шрифтів, зображень і мультимедіа.

Мультимедіа формати

Різні елементи мультимедіа (такі як аудіо або відео об'єкти) зберігаються в мультимедіа файлах. Найпоширенішим способом визначення типу файлу є його розширення. Мультимедійні файли мають різні формати та відповідно різні розширення: .wav, .mp3, .mp4, .mpg, .wmv або .avi.



Файли форматів MP4, WebM, та Ogg підтримуються HTML.

Формат MP4 є рекомендованим для відео YouTube.

Найрозповсюдженішими форматами зберігання відео-файлів наведені в таблиці нижче.

Формат	Розширення	Опис
MPEG	.mpg .mpeg	MPEG. Розробка Moving Pictures Expert Group. Перший популярний відеоформат в Інтернеті. Більше не підтримується у HTML.
AVI	.avi	AVI (Audio Video Interleave). Розробка Microsoft. Широко використовується у відеокамерах і телевізійній апаратній техніці. Добре відображується на комп'ютерах Windows, але не в веб-браузерах.
WMV	.wmv	WMV (Windows Media Video). Розробка Microsoft. Широко використовується у відеокамерах і телевізійній апаратній техніці. Добре відображується на комп'ютерах Windows, але не в веб-браузерах.
QuickTime	.mov	QuickTime. Розробка Apple. Широко використовується у відеокамерах і телевізійній апаратній техніці. Добре відображується на комп'ютерах Apple, але не в веб-браузерах.
RealVideo	.rm .ram	RealVideo. Розробка Real Media дозволяє потокове відео з низькою пропускнуою можливістю. Не відтворюється в веб-браузерах.

Flash	.swf .flv	Flash. Розробка Macromedia. Часто вимагає додаткового компонента (плагінів) для відтворення в веб-браузерах.
Ogg	.ogg	Theora Ogg. Розробка Xiph.Org Foundation. Підтримується HTML.
WebM	.webm	WebM. Розробка Mozilla, Opera, Adobe, and Google. Підтримується HTML.
MPEG-4 or MP4	.mp4	MP4. Розробка Moving Pictures Expert Group. Широко використовується у відеокамерах і телевізійній апаратній техніці. Підтримується всіма браузерами та рекомендована YouTube.

Зауважте, що лише відео-формати MP4, WebM та Ogg підтримуються стандартами HTML.

Популярні аудіо-формати

MP3 – це найкращий формат для стисненої записаної музики. Термін MP3 став синонімом цифрової музики.

Якщо веб-сайт має містити значну кількість записаної музики, то MP3 є найкращим вибором. Опис інших аудіо-форматів наведено в таблиці.

Формат	Розширення	Опис
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Основний формат для всіх електронних музичних пристроїв, таких як синтезатори та звукові карти комп'ютера. MIDI-файли замість звуку містять цифрові ноти, які

		можна відтворити за допомогою електроніки. Добре відтворюється на всіх комп'ютерах та музичному апаратному забезпеченні, але не в веб-браузерах.
RealAudio	.rm .ram	RealAudio. Розробка Real Media для забезпечення потокового відтворення аудіо за низької пропускнуої можливості. Не відтворюється в веб-браузерах.
WMA	.wma	WMA (Windows Media Audio). Розробка Microsoft. Добре відтворюється на комп'ютерах Windows, але не в веб-браузерах.
AAC	.aac	AAC (Advanced Audio Coding). Розроблений Apple в якості формату за умовчанням для iTunes. Добре відтворюється на комп'ютерах Apple, але не в веб-браузерах.
WAV	.wav	WAV. Розробка IBM та Microsoft. Добре відтворюється за управління операційними системами Windows, Macintosh та Linux. Підтримується стандартами HTML.
Ogg	.ogg	Ogg. Розробка Xiph.Org Foundation. Підтримується стандартами HTML.
MP3	.mp3	MP3 є звуковими частинами до файлів MPEG. MP3 є найпопулярнішим форматом для музичних програвачів. Поєднує гарне стиснення (файли невеликих розмірів) з високою якістю. Підтримується всіма браузерами.

MP4	.mp4	MP4 є форматом відео, але може також використовуватися для аудіо. Підтримується всіма браузерами.
-----	------	---

Зауважте, що лише аудіо-формати MP3, WAV та Ogg підтримуються стандартами HTML.

8.4. Відтворення відео HTML

Елемент HTML `<video>`

Елемент HTML `<video>` використовується для показу відео на веб-сторінці.

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

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

</body>
</html>
```



Атрибут `controls` додає елементи керування відео, такі як відтворення, пауза та гучність.

Слід завжди використовувати атрибути `width` та `height`. Якщо висота і ширина не встановлені, сторінка може мерехтіти під час завантаження відео.

Елемент `<source>` дозволяє вказати альтернативні відеофайли, які може вибрати браузер. Браузер буде використовувати перший формат, який зможе розпізнати.

Текст між тегами `<video>` та `</video>` tags відображатиметься лише у браузерах, які не підтримують елемент `<video>`.

Автоматичне відтворення елемента HTML `<video>`

Для того, щоб почати програвати відео-елемент автоматично, використовують атрибут **autoplay**:

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

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

</body>
</html>
```



Зверніть увагу, що відео-елемент в даному випадку не містить елементів управління відтворення або трекінгу часу.

Зауважте. Браузери родини Chrome не дозволяють здійснювати автовідтворення в більшості випадків. Однак, завжди дозволяється вимкнення відтворення.

Додавання атрибуту **muted** після атрибуту **autoplay** щоб відео було відтворюватися автоматично але без звуку:

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

Формати HTML відео

Існує три відео-формати, які підтримуються стандартами HTML: MP4, WebM і Ogg. Можливості підтримки браузерами цих форматів надана в таблиці

Browser	MP4	WebM	Ogg
Edge	YES	YES	YES

Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

Співвідношення між типами медіа-даних, що підтримуються стандартами HTML та форматами файлів наведено у таблиці.

Формат файлу	Тип медіа даних, які він містить
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

Методи, властивості та події відео-об'єктів HTML

Згідно з об'єктною моделлю HTML документів (DOM) визначаються методи, властивості та події для відео-об'єктів завданих елементом **<video>**.

Це дозволяє завантажувати, відтворювати та призупиняти відео, а також налаштовувати тривалість та гучність.

Існують також події DOM, які можуть повідомити вас про те, коли відео починає відтворюватися, призупинено тощо.

Приклад застосування JavaScript для відео-об'єктів

```
<!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="mov_bbb.mp4" type="video/mp4">
    <source src="mov_bbb.ogv" 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>
```

Результат має розмір: 625 x 535

Play/Pause

Big

Small

Normal



Video courtesy of [Big Buck Bunny](https://www.bigbuckbunny.org/).

Наведений код дозволяє управляти відтворенням та розмірами відео-об'єкта за допомогою відповідних кнопок.

Тег	Призначення
<u><video></u>	Визначає відео або фільм
<u><source></u>	Визначає декілька мультимедійних ресурсів для медіа-елементів, таких як <video> та <Audio>
<u><track></u>	Визначає текстові доріжки у програвачах мультимедійних даних

8.5.Відтворення аудіо в HTML

Елемент HTML *<audio>*

Елемент HTML *<audio>* елемент використовується для відтворення аудіо-файлів на веб-сторінці.

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

Атрибут **controls** додає елементи управління звуком, наприклад, відтворення, паузу та гучність.

Елемент *<source>* дозволяє вказати альтернативні аудіо-файли, які може вибрати браузер. Браузер буде використовувати перший файл, формат якого зможе розпізнати.

Текст між тегами *<audio>* та *</audio>* буде відображатися лише у браузерах, які не підтримують елемент *<audio>*.

Автоматичне відтворення елемента HTML *<audio>*

Для того, щоб почати програвати аудіо-елемент автоматично, використовують атрибут **autoplay**:

<audio controls autoplay>

Зауважте. Браузери родини Chrome не дозволяють здійснювати автовідтворення в більшості випадків. Однак, завжди дозволяється вимкнення відтворення.

Додавання атрибуту **muted** після атрибуту **autoplay** щоб аудіо було відтворюватися автоматично але без звуку:

<audio controls autoplay muted>

Формати аудіо в HTML

Існує три відео-формати, які підтримуються стандартами HTML: MP3, WAV і Ogg. Можливості підтримки браузерами цих форматів надана в таблиці

Browser	MP3	WAV	OGG
Edge/IE	YES	YES*	YES*
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

Співвідношення між типами медіа-даних, що підтримуються стандартами HTML та форматами файлів наведено у таблиці.

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

Методи, властивості та події аудіо-об'єктів HTML

Методи, властивості та події аудіо-об'єктів HTML `<audio>` визначаються Згідно з об'єктною моделлю HTML документів (DOM).

Це дозволяє завантажувати, відтворювати та призупиняти аудіо, а також налаштовувати тривалість та гучність.

Існують також події DOM, які можуть повідомити вас про те, коли аудіо починає відтворюватися, призупинено тощо.

Аудіо теги HTML наведені в таблиці.

Tag	Description
<code><audio></code>	Визначає звуковий об'єкт
<code><source></code>	Визначає декілька мультимедійних ресурсів для медіаелементів, таких як <code><video></code> та <code><audio></code>

Плагіни HTML

Плагіни – комп'ютерні програми, що розширюють стандартну функціональність браузера.

Плагіни призначені для використання в багатьох різних цілях:

- Для запуску Java Applets.
- Для запуску елементів керування Microsoft ActiveX.
- Для відтворення кіно у форматі Flash.
- Для відображення мап.
- Для перевірки наявності вірусів.
- Для перевірки банківського ідентифікаційного номера тощо

Зверніть увагу:

1. Більшість браузерів більше не підтримують Java-аплети (Java Applets) та Plug-ins.
2. Елементи управління ActiveX більше не підтримуються одним браузером.
3. Підтримка Shockwave Flash також була виключена з можливостей сучасних браузерів.

Елемент HTML *<object>*

Елемент *<object>* визначає вбудований об'єкт у HTML-документі. Його застосування підтримується можливостями всіх браузерів.

Він був розроблений для вбудовування плагінів (Java-аплетів, PDF-читачів та Flash програвачів) в веб-сторінки, але також може бути використаний для включення HTML-об'єктів до HTML-документів.

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

<object width="100%" height="500px" data="snippet.html"></object>

</body>
</html>
```

Alfreds Futterkiste	Berlin	Germany
Berglunds snabbköp	Luleå	Sweden
Centro comercial Moctezuma	México D.F.	Mexico
Ernst Handel	Graz	Austria
FISSA Fabrica Inter. Salchichas S.A.	Madrid	Spain
Galería del gastrónomo	Barcelona	Spain
Island Trading	Cowes	UK
Königlich Essen	Brandenburg	Germany
Laughing Bacchus Wine Cellars	Vancouver	Canada
Magazzini Alimentari Riuniti	Bergamo	Italy
North/South	London	UK
Paris spécialités	Paris	France
Rattlesnake Canyon Grocery	Albuquerque	USA
Simons bistro	København	Denmark
The Big Cheese	Portland	USA
Vaffeljernet	Århus	Denmark
Wolski Zajazd	Warszawa	Poland

Можна також вбудовувати зображення:

```
<object data="audi.jpeg"></object>
```

Елемент HTML *<embed>*

Елемент *<embed>* також визначає вбудований об'єкт в рамках HTML-документа. Його застосування підтримується можливостями всіх браузерів.

Веб-браузери тривалий час підтримували елемент *<embed>*. Однак, він не входив до специфікації HTML до HTML5.

```
<embed src="audi.jpeg">
```

Зауважте, що елемент *<embed>* не має завершального тегу. Він не може містити альтернативного тексту

Елемент *<embed>* також може бути використаний для включення HTML файлів до HTML документів:

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

<embed width="100%" height="500px" src="snippet.html">

</body>
</html>
```

Alfreds Futterkiste	Berlin	Germany
Berglunds snabbköp	Luleå	Sweden
Centro comercial Moctezuma	México D.F.	Mexico
Ernst Handel	Graz	Austria
FISSA Fabrica Inter. Salchichas S.A.	Madrid	Spain
Galería del gastrónomo	Barcelona	Spain
Island Trading	Cowes	UK
Königlich Essen	Brandenburg	Germany
Laughing Bacchus Wine Cellars	Vancouver	Canada
Magazzini Alimentari Riuniti	Bergamo	Italy
North/South	London	UK
Paris spécialités	Paris	France
Rattlesnake Canyon Grocery	Albuquerque	USA
Simons bistro	København	Denmark
The Big Cheese	Portland	USA
Vaffeljernet	Århus	Denmark
Wolski Zajazd	Warszawa	Poland

8.6. Відео YouTube в HTML

Найпростішим способом відтворення відео в HTML є використання YouTube.

Перетворення відео в різні формати може бути складним і трудомістким.

Простіше рішення – дозволити YouTube відтворювати відео на своїй веб-сторінці.

На YouTube відображатиметься ідентифікатор (наприклад tgbNymZ7vqY), під час збереження або відтворення відео. Можна використати цей ідентифікатор для звернення до відповідного відео у HTML-коді.

Щоб відтворити відео на веб-сторінці слід виконати наступні дії:

- Завантажити відео до YouTube
- Запам'ятати ідентифікатор цього відео
- Визначити елемент `<iframe>` на веб-сторінці
- Визначити атрибутом `src` URL цього відео
- За допомогою атрибутів `width` та `height` визначити розміри об'єкта, який буде відтворювати відео
- Додати до URL додаткові параметри (розглянуті нижче)

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

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

</body>
</html>
```



Автоматичне відтворення YouTube з зупинкою

Можна налаштувати автоматичне відтворення відео щойно користувач потрапляє на сторінку, додавши до URL YouTube параметр `autoplay=1`. Проте, автоматичний початок відео дратує відвідувачів.

Зауважте. Браузери родини Chrome не дозволяють здійснювати автовідтворення в більшості випадків. Однак, завжди дозволяється вимкнення відтворення.

Якщо після атрибуту `autoplay=1` додати атрибут `mute=1` відео буде відтворюватися автоматично але без звуку.

```

<!DOCTYPE html>
<html>
<body>

<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">
</iframe>

</body>
</html>

```



Можна додати список відео, відокремлених комами. Такий список слід додати до початкової адреси URL.

Параметр *loop*

Якщо вказати параметр **loop=1** відео буде відтворюватися знову й знову нескінченно. Якщо **loop=0** відео буде відтворюватися лише один раз (як за умовчанням).

```

<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7v
qY&loop=1">
</iframe>

```

Параметр *controls*

Якщо вказати параметр **controls=0** під час відтворення не будуть відображені елементи контролю програвача. З значення **controls=0** елементи контролю відображаються (як за умовчанням).

```

<!DOCTYPE html>
<html>
<body>

<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>

</body>
</html>

```








8.7. Визначення геолокації API в HTML

Для знаходження позиції користувача використовується API геопозиціювання HTML. API геопозиціювання HTML використовується для отримання географічного положення користувача.

Оскільки це може порушити конфіденційність, позиція недоступна, якщо користувач не схвалить її.

Зауваження: Геопозиціювання є найбільш точним для пристроїв з GPS, таких як, наприклад, смартфони.

Підтримка браузерми даної можливості показана у таблиці. Номера в таблиці вказують першу версію браузера, яка повністю підтримує функцію Geolocation.

API					
Geolocation	5.0 - 49.0 (http) 50.0 (https)	9.0	3.5	5.0	16.0

Починаючи з версії Chrome 50, API для геопозиціювання працюватиме лише над безпечними контекстами, такими як HTTPS. Якщо ваш сайт розміщений на незахищеному сервері (наприклад HTTP), запити на отримання адреси користувачів функціонувати не будуть.

Метод `getCurrentPosition()` метод використовується для отримання позиції користувача.

Приклад нижче повертає широту і довготу позиції користувача:

```
<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
function showPosition(position) {
```

```

x.innerHTML = "Latitude: " + position.coords.latitude +
"<br>Longitude: " + position.coords.longitude;
}
</script>

```

Цей приклад пояснює наступні моменти:

- Слід перевірити, чи підтримується геопозиціювання.
- Якщо підтримується використати метод `getCurrentPosition()`, інакше надати користувачеві відповідне повідомлення з поясненнями.
- Якщо метод `getCurrentPosition()` спрацював успішно, він повертає об'єкт координат до функції, вказаної у параметрі `function (showPosition())`.
- Функція `showPosition()` визначає широту та довготу.

Приклад, наведений вище, є базовим простим скриптом геопозиціювання, без обробки помилок.

Обробка помилок та відмов

Другий параметр метода `getCurrentPosition()` використовується для обробки помилок. Він визначає функцію, яку слід запустити, якщо не вдалося отримати координати користувача:

```

function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML = "Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML = "The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML = "An unknown error occurred."
      break;
  }
}

```

```
}  
}
```

Геолокація також дуже корисна для інформації, що стосується місця розташування, наприклад:

- Актуальна інформація щодо місця знаходження користувача (налаштування мови, формату часу та дати тощо)
- Можливість відображення соціальних об'єктів навколо користувача (магазинів, кафе тощо)
- Можливості надання навігаційних послуг (GPS)

Властивості метода `getCurrentPosition()`

У разі успішного виконання метод `getCurrentPosition ()` повертає об'єкт, який містить значення широти, довготи та точності завжди за умовчанням. Але можна отримати й інші дані за зазначення відповідних властивостей, вказаних в таблиці:

Властивість	Дані, що повертаються
<code>coords.latitude</code>	Широта як десяткове число (завжди повертається)
<code>coords.longitude</code>	Довгота як десяткове число (завжди повертається)
<code>coords.accuracy</code>	Точність положення (завжди повертається)
<code>coords.altitude</code>	Висота в метрах над середнім рівнем моря (повернена за наявності)
<code>coords.altitudeAccuracy</code>	Точність висоти положення (повертається за наявності)

coords.heading	Напрямок руху: визначається в градусах відхилення від півночі – 0 градусів, за годинниковою стрілкою (повертається, якщо є)
coords.speed	Швидкість у метрах за секунду (повертається за наявності)
timestamp	Дата/час відповіді (повертається за наявності)

Геолокаційні об'єкти також мають інші корисні методи:

- **watchPosition()** – повертає поточну позицію користувача і продовжує повертати оновлену позицію під час руху користувача (наприклад, GPS у автомобілі).
- **clearWatch()** – зупиняє роботу метода **watchPosition()**.

Наведений нижче приклад демонструє роботу метода **watchPosition()**. Для цього потрібен пристрій з точним GPS (такий, наприклад, як смартфон):

```
<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.watchPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

8.8. Стисла історія та технології вбудовування об'єктів

Здавна в Інтернеті було популярним використання фреймів для створення веб-сайтів — невеликих частин веб-сайту, що зберігаються в окремих HTML-сторінках. Ці параметри було вбудовано у головний документ, який називається *frameset*, який дозволяв вказати область на екрані, яку заповнював кожен кадр, а не розмір стовпчиків і рядків таблиці (як зараз). Це вважалося висотою майстерності розробки в середині-кінці 90-х. Було визначено, що коли веб-сторінка розділялася на менші елементи, це було краще з точки зору швидкості завантаження, що було особливо помітно, оскільки мережеві з'єднання були доволі повільними в той час. Однак одночасно було багато проблем, які значно переважали будь-які позитивні результати, оскільки швидкість мережі стала швидшою, а тому користувач не помічав різниці.

Трохи пізніше (кінець 90-х, початок 2000-х років), технології додатків стали дуже популярними, такі як Java-аплети і Flash. Це дозволило веб-розробникам вбудовувати різноманітний контент у веб-сторінки, такі як відео та анімація, які не були доступні лише засобами HTML. Вбудовування цих технологій було досягнуто за допомогою таких елементів, як `<object>`, і менш використовуваних `<embed>`, які були дуже корисними в той час. З тих пір ці елементи встигли втратити популярність через численні проблеми, включаючи доступність, безпеку, розмір файлу та багато інших. Сьогодні провідні браузері перестали підтримувати плагіни, такі як Flash.

Нарешті, з'явився елемент `<iframe>` (поряд з іншими способами вбудовування вмісту, такими як `<canvas>`, `<video>` та іншими). Поява цього елемента забезпечила спосіб вбудовування цілого веб-документа всередину іншого таким чином, як би він був елементом `` або іншим елементом подібного типу. Сьогодні такий спосіб використовується регулярно.

Розглянемо спосіб, який дозволяє вбудовувати YouTube відео на будь-яку сторінку за використання елемента `<iframe>`.

1. Обираємо відео на YouTube.

2. Під відео натискаємо кнопку *Share* для відображення параметрів спільного доступу.
3. Натискаємо кнопку *Embed* та отримуємо код для елемента `<iframe>` який слід скопіювати та додати до коду.

За цим самим алгоритмом можна вбудувати мапу Google Maps:

1. В Google Maps знайти потрібну мапу.
2. Натиснути на елемент "Hamburger Menu" (три горизонтальні лінії) в верхньому лівому куті вікна.
3. Обрати пункт *Share or embed map*.
4. Обрати пункт *Embed map*, за натискання якого отримуєте код елемента `<iframe>`, який слід скопіювати та додати до коду.

Такий метод є ефективним для того, щоб включити зовнішні об'єкти на свій веб-сайт. Ви маєте безпосереднього контролю над такими об'єктами і не бажаєте розробляти власну версію таких об'єктів (наприклад, відео від онлайн-провайдерів, коментування систем, таких як Disqus, мапи від провайдерів онлайн-карт, рекламні банери тощо).

Існують певні серйозні проблеми безпеки, на які слід зважати під час роботи з елементами `<iframe>`. Це не означає, що слід відмовлятися від їх використання на веб-сайтах, проте це потребує певних знань та ретельного обмислення заходів безпеки. Розглянемо приклад коду, яким включаємо глосарій MDN до веб-сторінки:

```
<head>
  <style>
    iframe {
      border: none;
    }
  </style>
</head>
<body>
  <iframe
    src="https://developer.mozilla.org/en-US/docs/Glossary"
    width="100%"
    height="500"
    allowfullscreen
    sandbox>
    <p>
      <a href="/en-US/docs/Glossary">
        Fallback link for browsers that don't support iframes
      </a>
    </p>
  </iframe>
</body>
```

Цей приклад включає основні засоби необхідні для використання елемента `<iframe>`:

[border: none](#)

Якщо використовується цей атрибут, то `<iframe>` відображається без зовнішньої рамки. В іншому випадку, за замовчуванням, браузері відображають `<iframe>` з рамкою (що, як правило, небажано).

[allowfullscreen](#)

Якщо встановлено цей атрибут, то відображення `<iframe>` можна перевести у повноекранний режим за допомогою API Fullscreen.

src

Містить шлях, що вказує на URL документа, який буде вбудовано для таких елементів як . `<video>/`

width and height

Ці атрибути визначають ширину і висоту, якими ви бажаєте бачити `<iframe>`.

Резервний вміст

Так само, як і інші подібні елементи, наприклад, `<video>`, можна включати альтернативний текст між мітками відкриття і закриття `<iframe></iframe>`, які з'являться, якщо браузер не підтримує використання елемента `<iframe>`. У цьому прикладі замість альтернативного тексту додане посилання на сторінку. Навряд чи сьогодні можна зустріти будь-який браузер, який не підтримує `<iframe>`.

sandbox

Цей атрибут, який працює в трохи більш сучасних браузерах, ніж решта функцій `<iframe>` (напр. IE 10 і вище) вимагає підвищеного налаштування безпеки. Обговоримо це питання в наступному пункті.

Зауваження: Для того, щоб збільшити швидкість, потрібно встановити атрибут `src ifrc iframe` за допомогою JavaScript після того, як основний контент буде завантажено. Це робить вашу сторінку придатнішою до використання (підвищить юзабіліті) та зменшує час завантаження офіційної сторінки (важлива метрика для SEO).

Питання безпеки

Розробники браузерів та веб-розробники дізналися на власному гіркому досвіді, що iframes є популярною ціллю атаки (офіційний термін: "Attack vector") для поганих людей в Інтернеті (часто їх називають хакерами, або точніше, крєкерами). Під час таких атак зловмисники намагаються змінити вашу веб-сторінку, або змусити людину робити щось, чого вони не мають

робити, наприклад, надавати конфіденційну інформацію: імена користувачів і паролі. Через це інженери зі специфікацій та розробники браузерів створили різноманітні механізми безпеки для того, щоб <iframe> був більш безпечним.

Зауваження: Clickjacking – це один з видів атак на iframe, за яких хакери вбудовують невидимий iframe у ваш документ (або вбудовують ваш документ у власний зловмисний веб-сайт) і використовують його для захоплення взаємодій користувачів. Це розповсюджений спосіб ввести в оману користувачів або вкрасти конфіденційні дані.

Розглянемо приклад. Спробуйте завантажити попередній приклад у браузер. Замість очікуваної сторінки, ви, ймовірно, побачите якесь повідомлення на кшталт *"Я не можу відкрити цю сторінку"*, і якщо ви подивитеся на консоль в інструментарій розробника браузера [browser developer tools](#), ви побачите повідомлення, яке повідомляє вам чому. У Firefox вам скажуть щось на зразок *«Завантаження "https://developer.mozilla.org/en-US/docs/Glossary" в кадрі відмовлено директивою "X-Frame-Options", яка має значення "ВІДМОВИТИ"»* (The loading of "[https://developer.mozilla.org/en-US/docs/Glossary](#)" in a frame is denied by "X-Frame-Options" directive set to "DENY"). Це відбувається тому, що розробники, які побудували MDN, включили налаштування на сервер, що обслуговує сторінки веб-сайту, яке не допускає вбудовування їх всередину елементів <iframe>s (розглянемо в пункті Директиви налаштування CSP, нижче). Дійсно, нема сенсу вбудовувати цілу сторінку MDN до іншої сторінки, якщо лише ви не прагнете зробити щось на зразок вбудовування цих сторінок на свій сайт, щоб потім видавати їх як свої власні, або спробувати вкрасти дані за допомогою clickjacking – в обох випадках це є зловмисними намірами.

Іноді має сенс вбудовувати сторонній контент (відео з YouTube або мапи), проте можна уникнути багатьох проблем, якщо вбудувати сторонній контент лише за крайньої необхідності. Загальне правило веб-безпеки: *«Ви ніколи не можете бути занадто обережними. Якщо ти щось зробив, двічі перевір. Якщо хтось інший зробив щось, стверджуємо, що цей контент є небезпечним, поки не доведено протилежне»*.

Під час вбудовування чужого контенту крім безпеки, ви маєте також враховувати питання інтелектуальної власності. Більшість контенту захищена авторським правом, оффлайн та онлайн, навіть іноді неочікуваний вміст (наприклад, більшість зображень на Wikimedia Commons). Ніколи не показувати вміст на веб-сторінці, якщо ви не є його розробником, або власники не надали вам письмового, однозначного дозволу. Штрафи за порушення авторських прав суворі. Знову ж таки, в цьому питанні ви теж ніколи не можете бути занадто обережними.

Якщо вміст ліцензовано, необхідно дотримуватися умов ліцензії. Наприклад, контент на MDN ліцензований під CC-BY-SA. Це означає, що ви повинні належним чином враховувати інтереси власника контенту, під час цитування вмісту, навіть якщо ви робите суттєві зміни.

Використання HTTPS

HTTPS – це зашифрована версія HTTP. Потрібно використовувати HTTPS під час створення та впровадження веб-сайтів, за можливості:

1. HTTPS зменшує ймовірність того, що віддалений вміст було спотворено під час пересилання,
2. 2. HTTPS запобігає доступу вбудованого контенту до вмісту вашого батьківського документа і навпаки.

Для долучення HTTPS до сайту вимагає встановлення спеціального сертифіката безпеки. Багато хостинг-провайдерів пропонують хостинг, що підтримує HTTPS, без необхідності робити будь-які дії самостійно для налаштування сертифікату. Але якщо вам потрібно самостійно налаштувати підтримку HTTPS для вашого сайту, *Let's Encrypt* надає інструменти та інструкції, які ви можете використовувати для автоматичного створення і встановлення необхідного сертифіката. Для автоматичного вбудовування та підтримки необхідного сертифіката безпеки найчастіше використовують популярні веб-сервери, включаючи веб-сервер *Apache*, *Nginx* та інші. Інструмент *Let's Encrypt* призначений для того, щоб зробити процес додавання сертифіката якомога простішим, тому немає дійсно ніяких поважних причин,

щоб уникнути використання його або інших доступних засобів для включення HTTPS вашого сайту.

Зауваження: Сторінки GitHub дозволяють за замовчуванням візуалізувати контент через HTTPS, тому він є прийнятним для хостинг-контенту. Якщо ви користуєтеся іншим хостинг-провайдером і не впевнені в його безпечній роботі, запитайте про це.

Використання атрибута `sandbox`

Завжди використовуйте атрибут `sandbox`. Щоб обмежити контроль над сайтом з боку злоумисників слід надати вбудованому вмісту лише дозволи, необхідні для виконання лише їх власних функцій. Звичайно, це стосується і власного контенту. Контейнер для коду, в якому програма може бути використана або протестована належним чином без загрози завдання шкоди решті кодової бази (випадково або злоумисно) називається `sandbox`.

Контент, який не розміщено в `sandbox` надає доступ до багатьох функцій (виконання JavaScript, подання форм, вигулькні вікна тощо) За умовчанням, можна встановити всі наявні обмеження за допомогою атрибута `sandbox` без параметрів, як це показано у нашому попередньому прикладі.

Якщо це абсолютно необхідно, можна додавати права доступу до кожної можливості окремо (всередині атрибута `sandbox = "attribute value"`):

Застосування додаткових обмежень вмісту за допомогою `sandbox`. Значення атрибута може бути порожнім для застосування всіх обмежень, або відокремлених пробілами значень для зняття певних обмежень:

- `allow-downloads-without-user-activation` Experimental: Дозволяє завантажувати файли без втручання користувача.
- `allow-downloads`: Дозволяє завантаження відбуватися за вказівкою користувача.
- `allow-forms`: Дозволяє ресурсу подавати форми. Якщо це ключове слово не використовується, подача форми блокується.
- `allow-modals`: Дозволяє ресурс [open modal windows](#).
- `allow-orientation-lock`: Дозволяє ресурс [lock the screen orientation](#).

- allow-pointer-lock: Дозволяє ресурсу використовувати [Pointer Lock API](#).
- allow-popups: Дозволяє спливаючі вікна (window.open(), target="_blank", або showModalDialog()). Якщо це ключове слово не було використано, вікно не буде відкрито.
- allow-popups-to-escape-sandbox: Дозволяє документу sandboxed відкривати нові вікна без збереження можливостей, які вказані в документі sandboxed. Наприклад, безпечно завантажувати рекламу за обмежень на сторінку, на яку зроблені посилання.
- allow-presentation: Дозволяє ресурсу розпочати презентацію ([presentation session](#)).
- allow-same-origin: Якщо це значення не вказано, то будь-який ресурс трактується як такий, що завжди є небезпечного походження (потенційно для такого ресурса не надається доступу до сховища даних/cookies і деяких API JavaScript).
- allow-scripts: Надає змогу запускати скрипти ресурсів (але не створювати контекстні вікна).
- allow-storage-access-by-user-activation Experimental: Дозволяє на запит зовнішніх ресурсів отримати доступ до батьківського сховища даних за допомогою API Storage Access.
- allow-top-navigation: Надає змогу ресурсу здійснювати переміщення у контексті верхнього рівня (який має назву _top).
- allow-top-navigation-by-user-activation: Надає змогу ресурсу здійснювати переміщення у контексті верхнього рівня, але лише за ініціативи користувача.

Зауваження:

- Коли вбудований документ має те ж саме походження, що і сторінка в яку такий документ вбудований, не бажаним є використання як *allow-scripts*, так і *allow-same-origin*, оскільки це дозволяє вбудованому документу видалити атрибут *sandbox* , тобто зробити використання ресурсу таким саме небезпечним, ніби атрибут *sandbox* взагалі не було використано.
- Використання атрибута *sandbox* не має сенсу, якщо зловмисник може відображати контент поза середовищем *sandboxed* так, ніби користувач відкриває документ в новій вкладці. Такий випадок повинен оброблятися окремо, щоб обмежити потенційні загрози з боку шкідливого програмного забезпечення.
- Атрибут *sandbox* не підтримується Internet Explorer до версії 9.

Важливе зауваження: ніколи не додавайте значень *allow-scripts* та *allow-same-origin* до вашого атрибуту *sandbox* – в цьому випадку вбудований контент може обійти політику безпеки, і використовувати JavaScript для того, щоб взагалі вимкнути *sandbox*.

Примітка: Пісочниці не захищають, якщо зловмисники можуть обдурити людей, які відвідують зловмисний контент безпосередньо (поза пісочницею). Якщо є ймовірність того, що певний вміст може бути зловмисним (наприклад, створений користувачем контент), будь ласка, подайте його з іншого домену на ваш головний сайт.

Директиви налаштування CSP

CSP відповідає політиці безпеки вмісту і надає набір HTTP-заголовків (метадані, які надсилаються разом з вашими веб-сторінками, коли вони обслуговуються з веб-сервера), призначених для поліпшення безпеки вашого HTML-документа. Якщо мова йде про використання елементів `<iframe>`, можна налаштувати сервер на надсилення відповідного заголовка X-Frame-Options. Це може перешкодити іншим веб-сайтам вбудовувати ваш контент в свої веб-сторінки (що перешкоджає *sickjacking* та іншим видам атак).

Елементи [<embed>](#) та [<object>](#)

Елементи [<embed>](#) та [<object>](#) виконують багато функцій в елементі [<iframe>](#) - ці елементи є засобами вбудовування загального призначення для вбудовування зовнішнього контенту, наприклад, PDFs.

Проте ці елементи використовуються не дуже часто. Для демонстрації PDFs, краще використати посилання на контент, ніж вбудовувати його на сторінку.

Історично ці елементи також використовувалися для вбудовування контенту, що обробляється плагінами браузера, такими як Adobe Flash, але ця технологія зараз застаріла і не підтримується сучасними браузерами.

За необхідності вбудовування плагіну слід зауважити дані наступної таблиці:

	<embed>	<object>
URL контенту, що вбудовується	src	data
Точний тип медіа (властивість media type або content type) того контенту, що вбудовується	type	type
Висота та ширина (в пікселях CSS) контейнера, який контролюється плагіном	height width	height width
назви і значення, для підключення плагінів як параметрів	спеціальні атрибути з цими іменами і значеннями	одиничний тег елемента <param> , що міститься в елементі <object>
Незалежний HTML-вміст як запасний варіант у випадку, якщо певний ресурс є недоступним	не підтримується (<noembed> є застарілим)	що міститься в елементі <object>, після елементу <param>

Розглянемо приклад використання <object> за вбудовування PDF-файлу до сторінки:

```
<object data="mypdf.pdf" type="application/pdf" width="800"
height="1200">
  <p>
    You don't have a PDF plugin, but you can
    <a href="mypdf.pdf">download the PDF file. </a>
  </p>
</object>
```

PDF файли були необхідними під час переходу від паперових документів до цифрових, але вони створюють багато проблем доступності і мають низьку

читабельність на невеликих екранах. Вони все ще користуються популярністю в деяких колах, але набагато краще робити на них посилання, щоб документи можна було завантажити або прочитати на окремій сторінці, а не вбудовувати в веб-сторінку.