

2.1 2.1 Teoria

Step 1

Przykłady znajdują się w repozytorium pod

adresem: https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_2
https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_2

Jeśli tworzymy stronę internetową, prawdopodobnie zechcemy połączyć się z innymi stronami internetowymi i zasobami, niezależnie od tego, czy znajduje się w witrynie własnej, czy innej. W końcu łączenie jest tym, o co chodzi w sieci. Jest jeden element, który umożliwia powiązanie: jest to tzw. **kotwica** czyli znacznik `a`.

```
<a>...</a>
```

Aby zaznaczyć tekst jako łącze, po prostu należy umieścić go w otwierające i zamkajające znaczniki `<a>...` i użyć atrybutu `href`, aby podać adres URL strony docelowej. Zawartość elementu kotwicy staje się **łączem hipertekstowym**.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro</title>
</head>
<body>
    <div id="main">
        <a href="https://mmiotk.gitlab.io">Mateusz Miotk Website</a>
    </div>
</body>
</html>
```

[Mateusz Miotk Website](https://mmiotk.gitlab.io)

Aby obraz stał się linkiem, po prostu należy umieścić element `img` w elemencie zakotwiczenia:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro</title>
</head>
<body>
    <div id="main">
        <a href="https://mmiotk.gitlab.io"></a>
    </div>
</body>
</html>
```

Nawiasem mówiąc, możemy umieścić dowolny element treści `HTML` w zakotwiczeniu, aby stał się linkiem, a nie tylko obraz. Prawie wszystkie przeglądarki graficzne wyświetlają połączony tekst jako niebieski i domyślnie podkreślony. Niektóre starsze przeglądarki umieszczały niebieskie obramowanie wokół połączonych obrazów, ale większość obecnych już tego nie robi. Odwiedzone linki są zazwyczaj wyświetlane w kolorze fioletowym. Użytkownicy mogą zmieniać te kolory w preferencjach przeglądarki i oczywiście możemy zmienić wygląd linków do swoich witryn za pomocą arkuszy stylów. Gdy użytkownik kliknie lub dotknie połączonego tekstu lub obrazu, strona określona w elemencie zakotwiczenia ładuje się w oknie przeglądarki.

Musimy poinformować przeglądarkę, do którego dokumentu ma prowadzić link. Atrybut `href` (**hypertext reference**) zapewnia przeglądarce adres strony lub zasobu (jej adres `URL`). Adres `URL` musi zawsze znajdować się w cudzysłowie. W większości przypadków będziemy wskazywać inne dokumenty `HTML`; możemy jednak również wskazać inne zasoby internetowe, takie jak obrazy, pliki audio i wideo. Ponieważ nie ma zbyt wiele do umieszczania tagów kotwczących wokół niektórych treści, prawdziwą sztuczką z linkowaniem jest uzyskanie poprawnego adresu `URL`. Istnieją dwa sposoby określenia adresu `URL`:

- **Bezwzględne adresy** `URL` zawierają pełny adres `URL` dokumentu, w tym protokół (`http://` lub `https://`), nazwę domeny i ścieżkę, jeśli to konieczne. Musimy użyć bezwzględnego adresu `URL`, gdy wskazujemy dokument w sieci (tzn. nie na własnym serwerze). Czasami, gdy strona, do której linkujemy, ma długą ścieżkę URL, link może wyglądać na dość mylący. Należy jednak pamiętać, że struktura nadal jest prostym elementem kontenera z jednym atrybutem.
- **Względne adresy** `URL` opisują ścieżkę do pliku względem bieżącego dokumentu. Względnych adresów `URL` można używać, gdy łączymy się z innym dokumentem we własnej witrynie (tj. na tym samym serwerze). Nie wymaga protokołu ani nazwy domeny – wystarczy sama nazwa ścieżki:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro</title>
</head>
<body>
    <div id="main">
        <a href="about.html">About page</a>
    </div>
</body>
</html>
```

Wiele razy będziemy chcieli utworzyć link do strony, którą znaleźliśmy w sieci. Jest to znane jako **link zewnętrzny**, ponieważ prowadzi do strony poza własnym serwerem lub witryną. Aby utworzyć łącze zewnętrzne, należy podać bezwzględny adres URL, zaczynając od http:// (protokół). To mówi przeglądarce: „Wyjdź do sieci i pobierz następujący dokument”.

Duża część linków, które wykonujemy, znajduje się między stronami własnej witryny: od strony głównej do stron sekcji, od stron sekcji do stron z treścią i tak dalej. W takich przypadkach możemy użyć względnego adresu URL – takiego, który wywołuje stronę na własnym serwerze. Bez „http://” przeglądarka szuka na aktualnym serwerze linkowanego dokumentu. Ścieżka, notacja używana do wskazania konkretnego pliku lub katalogu mówi przeglądarce, gdzie znaleźć plik. Ścieżki internetowe są zgodne z konwencją Uniksa polegającą na oddzielaniu nazw katalogów i plików za pomocą ukośników (/). Ścieżka względna opisuje, jak dostać się do połączonego dokumentu, zaczynając od lokalizacji bieżącego dokumentu. Względne nazwy ścieżek mogą być nieco skomplikowane. W moim doświadczeniu w nauczaniu nic tak nie zniechęca początkujących, jak pisanie względnych nazw ścieżek, więc zrobimy to krok po kroku. Najprostszy względny adres URL wskazuje na inny plik w tym samym katalogu. W przypadku linkowania do pliku w tym samym katalogu wystarczy podać tylko nazwę pliku (jego nazwę pliku). Gdy adres URL jest tylko nazwą pliku, serwer szuka pliku w bieżącym katalogu (tj. katalogu zawierającym dokument z łączem). Łącze do nazwy pliku wskazuje, że połączony plik znajduje się w tym samym katalogu, co bieżący dokument.

Ale co, jeśli pliki nie znajdują się w tym samym katalogu? Musimy dać przeglądarce wskazówki, dołączając nazwę ścieżki w adresie URL. Zobaczmy, jak to działa. Wracając do naszego przykładu, pliki są przechowywane w podkatalogu o nazwie pages. Chcemy utworzyć link z index.html do pliku w katalogu pages o nazwie about.html. Ścieżka w adresie URL mówi przeglądarce, aby szukała w bieżącym katalogu katalogu o nazwie pages, a następnie szukała pliku about.html.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro</title>
</head>
<body>
    <div id="main">
        <a href="pages/about.html">About page</a>
    </div>
</body>
</html>
```

Przejdzmy teraz do pliku o nazwie `sale.html`, który znajduje się w podkatalogu `sales`. Wszystko, co musimy zrobić, to podać ścieżkę przez dwa podkatalogi (`pages`, a następnie `sales`) do `couscous.html`.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro</title>
</head>
<body>
    <div id="main">
        <a href="pages/about.html">About page</a>
        <a href="pages/sales/couscous.html">Couscous page</a>
    </div>
</body>
</html>
```

Tym razem pójdźmy w innym kierunku i utwórzmy link ze strony z `about.html` z powrotem do strony głównej, która znajduje się jeden poziom wyżej w katalogu. W Uniksie istnieje konwencja nazw ścieżek tylko do tego celu, „kropka-kropka-ukośnik” (`..`). Kiedy zaczynamy ścieżkę od `..`, jest to to samo, co mówienie przeglądarkę „utwórz kopię zapasową jednego poziomu katalogu”, a następnie podążaj ścieżką do określonego pliku. Jeśli znamy się na przeglądaniu plików na pulpicie, warto wiedzieć, że `..` działa tak samo, jak kliknięcie przycisku W góre w Eksploratorze Windows lub przycisku strzałki w lewo w Finderze w systemie macOS. Zaczniemy od stworzenia linku z `about.html` z powrotem do strony głównej (`index.html`). Ponieważ `about.html` znajduje się w podkatalogu `pages`, musimy wrócić do katalogu głównego, aby znaleźć `index.html`. Ta ścieżka mówi przeglądarkę, aby „utworzyła kopię zapasową o jeden poziom”, a następnie poszukała w tym katalogu `index.html`.

```
<!--about.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>About page</title>
</head>
<body>
    <p>This is about page</p>
    <p><a href="..../index.html">Home page</a></p>
</body>
</html>
```

Ale jak wrzucić link z powrotem do strony głównej z pliku `couscous.htm`! Należy po prostu użyć kropki-kropki-ukośnika dwa razy.

```
<!--couscous.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Couscous page</title>
</head>
<body>
    <p>This is a Couscous page.</p>
    <p><a href=".../index.html">Home page</a></p>
</body>
</html>
```

Pierwszy `../` tworzy kopię zapasową w katalogu `pages`; drugi `.../` tworzy kopię zapasową do katalogu najwyższego poziomu, gdzie można znaleźć `index.html`. Ponownie, nie ma potrzeby wpisywania nazw katalogów; `.../` robi to wszystko.

Wszystkie witryny mają **katalog główny**, katalog zawierający wszystkie katalogi i pliki witryny. Do tej pory wszystkie przeglądane przez nas nazwy ścieżek odnoszą się do dokumentu z linkiem. Innym sposobem napisania względnej nazwy ścieżki jest rozpoczęcie od katalogu głównego i wyświetlenie listy nazw podkatalogów do pliku, do którego chcemy utworzyć link. Ten typ ścieżki jest nazywany **względną katalogu głównego witryny**. W konwencji uniksowej nazw ścieżek ukośnik (`/`) na początku nazwy ścieżki wskazuje, że ścieżka zaczyna się w katalogu głównym. Względna ścieżka katalogu głównego witryny w poniższym łączu brzmi: „Przejdź do katalogu najwyższego poziomu dla tej witryny, otwórz katalog `pages`, a następnie znajdź plik `about.html`”.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro</title>
</head>
<body>
<div id="main">
    <a href="/pages/about.html">About page</a>
    <a href="/pages/sales/couscous.html">Couscous page</a>
</div>
</body>
</html>
```

Zauważmy, że nie musimy (i nie powinniśmy) wpisywać nazwy katalogu głównego w ścieżce – ukośnik (/) na początku reprezentuje katalog najwyższego poziomu w ścieżce. Stamtąd po prostu określ katalogi, w których przeglądarka powinna szukać. Ponieważ ten typ łącza zaczyna się od katalogu głównego, aby opisać nazwę ścieżki, działa z dowolnego dokumentu na serwerze, niezależnie od tego, w którym podkatalogu może się znajdować. Łącza względne katalogu głównego witryny są przydatne w przypadku treści, które nie zawsze znajdują się w tym samym katalogu, lub materiałów generowanych dynamicznie. Ułatwiają również kopowanie i wklejanie łączy między dokumentami. Z drugiej strony linki nie będą działać na lokalnym komputerze, ponieważ będą one powiązane z dyskiem twardym. Będziemy musieli poczekać, aż witryna znajdzie się na końcowym serwerze, aby sprawdzić, czy linki działają.

```
<!--couscous.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Couscous page</title>
</head>
<body>
    <p>This is a Couscous page.</p>
    <p><a href="/index.html">Home page</a></p>
</body>
</html>
```

Atrybut `src` w elemencie `img` działa tak samo jak atrybut `href` w kotwicach. Ponieważ najprawdopodobniej będziemy używać obrazów z własnego serwera, atrybuty `src` w elementach obrazu zostaną ustalone na względne adresy `URL`.

Step 2

Możemy połączyć się z określonym punktem na stronie internetowej. Przydaje się to do udostępniania skrótów do informacji na dole długiej, przewijanej strony lub do powrotu na góre strony jednym kliknięciem lub dotknięciem. Łączenie do określonego miejsca na stronie jest również nazywane **łączem do fragmentu dokumentu**. Tworzenie linków do konkretnego miejsca na stronie to proces dwuczęściowy. Najpierw należy zidentyfikować miejsce docelowe, a następnie utworzyć do niego link.

Identyfikacja miejsca docelowego jest jak umieszczenie flagi w dokumencie, aby można było łatwo do niego wrócić. Aby utworzyć miejsce docelowe, należy użyć atrybutu `id`, aby nadać elementowi docelowemu w dokumencie unikalną nazwę (jest to „unikalna”, ponieważ nazwa może pojawić się tylko raz w dokumencie. W żargonie internetowym jest to **identyfikator fragmentu**.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Black Goose Bistro</title>
</head>
<body>
<div id="main">
<a href="pages/about.html">About page</a>
<a href="pages/sales/couscous.html">Couscous page</a>
<h3 id="startH">H</h3>
<ul>
<li>H1</li>
<li>H2</li>
<li>H3</li>
</ul>
<h3 id="startI">I</h3>
<ul>
<li>I1</li>
<li>I2</li>
<li>I3</li>
</ul>
<h3 id="startJ">J</h3>
<ul>
<li>J1</li>
<li>J2</li>
<li>J3</li>
</ul>
</div>
</body>
</html>
```

Mając identyfikator na swoim miejscu, możemy teraz utworzyć do niego link. Na górze strony utworzymy link do fragmentu „`startH`”. Jeśli chodzi o każdy link, do podania lokalizacji linku używamy elementu `a` z atrybutem `href`. Aby wskazać, że łączymy się z fragmentem, należy użyć symbolu (#), przed nazwą fragmentu. Teraz, gdy ktoś kliknie literę `H` na liście u góry strony, przeglądarka zeskoczy w dół i wyświetli sekcję zaczynającą się od nagłówka `H`.

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Black Goose Bistro</title>
</head>
<body>
<div id="main">
<a href="pages/about.html">About page</a>
<a href="pages/sales/couscous.html">Couscous page</a>
<br>
<a href="#startH">H</a>
<a href="#startI">I</a>
<a href="#startJ">J</a>
<h3 id="startH">H</h3>
<ul>
<li>H1</li>
<li>H2</li>
<li>H3</li>
</ul>
<h3 id="startI">I</h3>
<ul>
<li>I1</li>
<li>I2</li>
<li>I3</li>
</ul>
<h3 id="startJ">J</h3>
<ul>
<li>J1</li>
<li>J2</li>
<li>J3</li>
</ul>
</div>
</body>
</html>

```

[About page](#) [Couscous page](#)
[H](#) [I](#) [J](#)

H

- H1
- H2
- H3

1

I

- I1
- I2
- I3

J

Możemy utworzyć link do fragmentu w innym dokumencie, dodając nazwę fragmentu na końcu adresu URL (bez względnego lub względnego). Na przykład, aby utworzyć łącze do nagłówka „H” strony glosariusza z innego dokumentu w tym katalogu, adres URL wyglądałby tak:

```
<a href="glossary.html#startH">See the Glossary, letter H</a>
```

Możemy nawet tworzyć linki do określonych miejsc docelowych na stronach w innych witrynach, umieszczając identyfikator fragmentu na końcu bezwzględnego adresu URL, na przykład:

```
<a href="http://www.example.com/glossary.html#startH">See the Glossary, letter H</a>
```

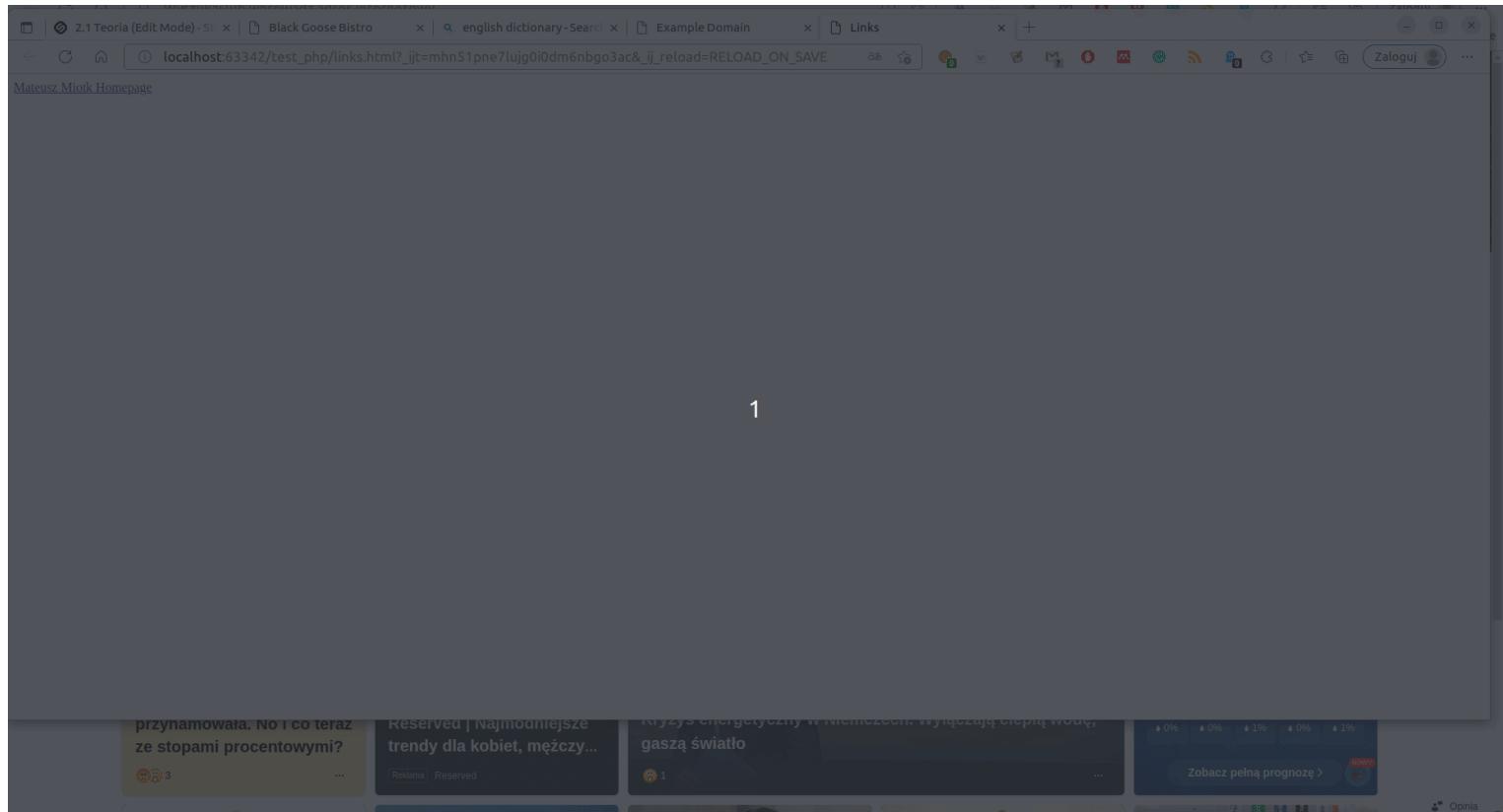
Oczywiście nie mamy żadnej kontroli nad nazwanymi fragmentami na stronach internetowych innych osób. Punkty docelowe muszą być wprowadzone przez autora tych dokumentów, aby były dostępne dla nas. Jednym sposobem, aby dowiedzieć się, czy tam są i gdzie się znajdują, jest „Wyświetl źródło” strony i poszukanie ich w znacznikach. Jeśli fragmenty w zewnętrznych dokumentach przesuną się lub znikną, strona nadal będzie się ładować; przeglądarka po prostu przejdzie na góre strony, tak jak w przypadku zwykłych linków.

Step 3

Jednym z problemów związanych z umieszczaniem linków na stronie jest to, że gdy ludzie je klikną, mogą nigdy nie wrócić do naszej treści. Tradycyjne rozwiązywanie tego dylematu polegało na otwarciu linkowanej strony w nowym oknie przeglądarki. W ten sposób odwiedzający mogą sprawdzić link i nadal mieć dostęp do treści tam, gdzie je zostawili. Pamiętajmy, że otwieranie nowych okien przeglądarki może powodować czkawkę w zadowoleniu użytkowników witryny. Otwieranie nowych okien jest problematyczne z punktu widzenia ułatwień dostępu i może być mylące dla niektórych użytkowników. Mogą nie być w stanie stwierdzić, że otworzyło się nowe okno lub mogą nigdy nie wrócić do oryginalnej strony. Przynajmniej nowe okna mogą być postrzegane jako utrapienie, a nie udogodnienie. Należy się więc zastanowić, czy potrzebujemy nowego okna i czy korzyści przeważają nad potencjalnymi wadami. Sposób, w jaki otwieramy łącze w nowym oknie przeglądarki, zależy od tego, czy chcemy kontrolować jego rozmiar. Jeśli rozmiar okna nie ma znaczenia, możemy użyć samych znaczników HTML. Jeśli jednak chcemy otworzyć nowe okno o określonych wymiarach w pikselach, musimy użyć skryptu JavaScript.

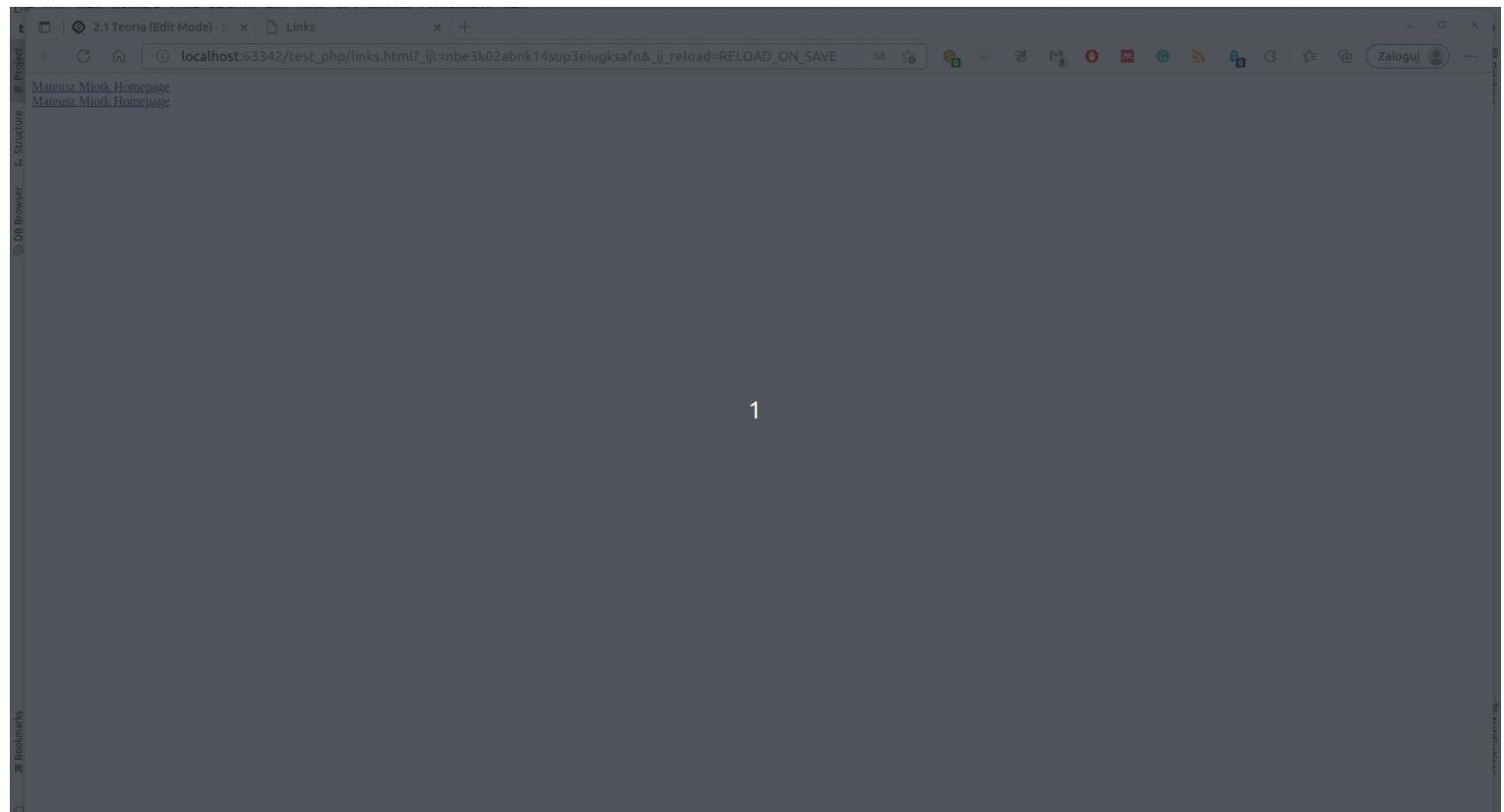
Aby otworzyć nowe okno ze znacznikami, należy użyć atrybutu target w elemencie zakotwiczenia (), aby poinformować przeglądarkę o nazwie okna, w którym chcemy otworzyć połączony dokument. Wartość target możemy ustawić na _blank lub dowolną wybraną nazwę. Pamiętajmy, że dzięki tej metodzie nie mamy kontroli nad rozmiarem okna, ale zazwyczaj otworzy się ono jako nowa karta lub w nowym oknie o takim samym rozmiarze, jak ostatnio otwarte okno w przeglądarce użytkownika. Nowe okno może, ale nie musi, zostać przeniesione na wierzch w zależności od używanej przeglądarki i urządzenia. Ustawienie target="_blank" zawsze powoduje, że przeglądarka otwiera nowe okno.

```
<!--links.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Links</title>
</head>
<body>
    <div id="main">
        <a href="https://mmiotk.gitlab.io" target="_blank">Mateusz Miotk Homepage</a>
    </div>
</body>
</html>
```



Jeśli ustawimy wartość `target="_blank"` dla każdego linku, każdy link uruchomi nowe okno, potencjalnie pozostawiając użytkownika z bałaganem otwartych okien. Nie ma w tym nic złego per se, o ile nie jest nadużywany. Inną metodą jest nadanie oknu docelowemu określonej nazwy, która może być następnie użyta przez kolejne linki. Możemy nadać oknu dowolną nazwę („new”, „sample”, cokolwiek), o ile nie zaczyna się od podkreślenia. Poniższy link otworzy nowe okno o nazwie `display`.

```
<!--links.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Links</title>
</head>
<body>
    <div id="main">
        <a href="https://mmiotk.gitlab.io" target="_blank">Mateusz Miotk Homepage</a><br>
        <a href="https://mmiotk.gitlab.io" target="display">Mateusz Miotk Homepage</a>
    </div>
</body>
</html>
```



Jeśli `target` będzie ustawione na wartość `display` do każdego łącza na stronie, każdy połączony dokument zostanie otwarty w tym samym drugim oknie. Niestety, jeśli drugie okno pozostaje ukryte za bieżącym oknem użytkownika, może się wydawać, że link po prostu nie zadziała. Możemy zdecydować, która metoda (nowe okno dla każdego łącza lub ponowne użycie nazwanych okien) jest najbardziej odpowiednia dla naszej zawartości i interfejsu.

Step 4

Oto sprytna sztuczka z linkowaniem: link `mailto`. Korzystając z protokołu `mailto` w łączu, możemy połączyć się z adresem e-mail. Gdy użytkownik kliknie łącze `mailto`, przeglądarka otwiera nową wiadomość pocztową zaadresowaną na ten adres w wyznaczonym programie pocztowym.

```
<!--links.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Links</title>
</head>
<body>
    <div id="main">
        <a href="https://mmiotk.gitlab.io" target="_blank">Mateusz Miotk Homepage</a><br>
        <a href="https://mmiotk.gitlab.io" target="display">Mateusz Miotk Homepage</a><br>
        <a href="mailto:mmiotk@pjwstk.edu.pl">Mail to Mateusz Miotk</a>
    </div>
</body>
</html>
```

Jak widać, jest to standardowy element z atrybutem `href`. Ale wartość jest ustawiona na `mailto:nazwa@adres.com`. Przeglądarka musi być skonfigurowana do uruchamiania programu pocztowego, więc efekt nie zadziała na 100% przeglądarek. Należy pamiętać, że umieszczenie adresu e-mail w źródle dokumentu powoduje, że jest on podatny na otrzymywanie niechcianych wiadomości-śmieci (znanych jako spam). Osoby, które tworzą listy spamowe, czasami używają automatycznych programów do wyszukiwania (zwanych botami), aby przeszukiwać sieć w poszukiwaniu adresów e-mail. Jeśli chcemy, aby adres e-mail był wyświetlany na stronie, aby ludzie mogli go rozszyfrować, ale roboty nie, możesz zdekonstruować adres w sposób, który jest nadal zrozumiały dla ludzi – na przykład `mmiotk[@].pjwstk.edu.pl`. Jednym z rozwiązań jest zaszyfrowanie adresu e-mail za pomocą `JavaScript`.

Smartfony są używane do uzyskiwania dostępu do witryny, więc mogą być również używane do wykonywania połączeń telefonicznych!

Dlaczego nie oszczędzić odwiedzającym kroku, pozwalając im wybrać numer telefonu w witrynie, po prostu dotykając go na stronie?

Składnia wykorzystuje protokół tel: i jest bardzo prosta:

```
<!--links.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Links</title>
</head>
<body>
    <div id="main">
        <a href="https://mmiotk.gitlab.io" target="_blank">Mateusz Miotk Homepage</a><br>
        <a href="https://mmiotk.gitlab.io" target="display">Mateusz Miotk Homepage</a><br>
        <a href="mailto:mmiotk@pjwstk.edu.pl">Mail to Mateusz Miotk</a><br>
        <a href="tel:+48-111-222-333">Call us</a>
    </div>
</body>
</html>
```

Gdy użytkownicy mobilni dotkną łącza, to, co się stanie, zależy od urządzenia: **Android** uruchamia aplikację telefonu; **BlackBerry** i **IE11 Mobile** natychmiast inicują połaczenie; a **iOS** uruchamia okno dialogowe z opcją połączenia, wiadomości lub dodania numeru do Kontaktów. Przeglądarki komputerowe mogą uruchamiać okno dialogowe w celu przełączenia aplikacji (na przykład na **FaceTime** w **Safari**) lub mogą zignorować łącze. Jeśli nie chcemy żadnych przerw w przeglądarkach komputerowych, możemy użyć reguły **CSS**, która ukrywa link dla urządzeń niemobilnych. Istnieje kilka dobrych praktyk dotyczących korzystania z łączów telefonicznych:

- Zaleca się podanie pełnego międzynarodowego numeru telefonu, w tym numeru kierunkowego kraju, dla wartości **tel:**, ponieważ nie ma możliwości sprawdzenia, gdzie użytkownik będzie miał dostęp do witryny.
- Uwzględnić należy również numer telefonu w treści linku, aby jeśli link nie działa, numer telefonu był nadal dostępny.
- **Android** i **iPhone** mają funkcję, która wykrywa numery telefonów i automatycznie zamienia je w linki. Niestety, niektóre 10-cyfrowe numery, które nie są numerami telefonów, również mogą zostać zamienione w linki. Jeśli dokument zawiera ciągi cyfr, które mogą zostać pomyłone z numerami telefonów, możemy wyłączyć automatyczne wykrywanie, umieszczając następujący metaelement w nagłówku dokumentu. Zapobiegnie to również zastępowaniu stylów zastosowanych do łączów telefonicznych.

```
<meta name="format-detection" content="telephone=no">
```

Step 5

Masowa popularność sieci wynikała częściowo z faktu, że na stronie znajdowały się **obrazy**. Zanim pojawiły się obrazy, internet był tylko tekstoną tundrą. Obrazy pojawiają się na stronach internetowych na dwa sposoby: osadzone w treści wbudowanej lub jako obrazy tła. Jeśli obraz jest częścią treści redakcyjnych, takich jak zdjęcia produktów, obrazy z galerii, reklamy, ilustracje itd., należy go umieścić w przepływie dokumentu **HTML**. Jeśli obraz jest czysto dekoracyjny, na przykład zachwycający obraz w tle nagłówka lub wzorzyste obramowanie wokół elementu, należy go dodać za pomocą kaskadowych arkuszy stylów. Umieszczanie obrazów wpływających na prezentację w arkuszu stylów nie tylko ma sens, ale sprawia, że dokument jest czystszy i bardziej dostępny, a projekt jest znacznie łatwiejszy do późniejszej aktualizacji. Najpierw przyjrzymy się wypróbowanemu i prawdziwemu elementowi **img** do dodawania podstawowych obrazów do strony, tak jak robimy to od 1992 roku. Działa dobrze przez ponad 25 lat. Następnie przedstawimy niektóre z dostępnych metod osadzania obrazów **SVG** (Scalable Vector Graphics) w dokumentach **HTML**. Pliki **SVG** są szczególnym przypadkiem i wymagają szczególnej uwagi. Na koniec przyjrzymy się, w jaki sposób znaczniki obrazu musiały dostosować się do szerokiej gamy urządzeń mobilnych, wprowadzając nowe responsywne elementy obrazu (obraz i źródło) oraz atrybuty (zestaw plików i rozmiary). Ponieważ liczba typów urządzeń używanych do oglądania sieci zaczęła gwałtownie rosnąć, zdaliśmy sobie sprawę, że pojedynczy obraz może nie spełniać potrzeb wszystkich środowisk oglądania, od ekranów wielkości dłoni w wolnych sieciach komórkowych po kinowe ekrany o dużej gęstości. Potrzebowaliśmy sposobu, aby obrazy były „responsywne”, to znaczy obsługiwały obrazy odpowiednie dla ich środowisk przeglądania. Po kilku latach współpracy między **W3C** a społecznością programistów, do

specyfikacji [HTML 5.1](#) dodano funkcje responsywnego obrazu i zaczynają być szeroko rozpowszechnione wsparcie przeglądarek. Chcę z góry zaznaczyć, że responsywne oznaczanie obrazów nie jest tak proste. Opiera się na bardziej zaawansowanych koncepcjach tworzenia stron internetowych, a składnia może być trudna dla kogoś, kto dopiero zaczyna pisać [HTML](#).

Za chwilę przejdziemy do elementu [img](#) i innych przykładów znaczników, ale najpierw ważne jest, aby wiedzieć, że nie możemy umieścić dowolnego obrazu na stronie internetowej; musi być w jednym z formatów obsługiwanych przez Internet. Ogólnie rzecz biorąc, obrazy składające się z siatki kolorowych pikseli (zwanych **obrazami bitmapowymi** lub rastrowymi, muszą być zapisane w formatach plików [PNG](#), [JPEG](#) lub [GIF](#), aby można je było umieszczać na stronie. Nowsze, bardziej zoptymalizowane formaty obrazów bitmapowych [WebP](#) i [JPEG-XR](#) powoli zyskują na popularności, szczególnie teraz, gdy mamy znaczniki, aby udostępnić je przeglądarkom, które je obsługują. W przypadku **obrazów wektorowych**, takich jak ikony i ilustracje tworzone za pomocą narzędzi do rysowania, takich jak [Adobe Illustrator](#), mamy format [SVG](#). Jeśli mamy obraz źródłowy w innym popularnym formacie, takim jak [TIFF](#), [BMP](#) lub [EPS](#), przed dodaniem go do strony musimy przekonwertować go na format internetowy. Jeśli z jakiegoś powodu musimy zachować plik graficzny w jego oryginalnym formacie (na przykład plik do programu CAD), możemy udostępnić go jako obraz zewnętrzny, tworząc link bezpośrednio do pliku obrazu, na przykład:

```
<a href="architecture.eps">Get the drawing</a>
```

Pliki obrazów należy nazywać odpowiednimi przyrostkami – odpowiednio [.png](#), [.jpg](#) (lub [.jpeg](#)), [.gif](#), [.webp](#) i [.jxr](#). Ponadto serwer musi być skonfigurowany, aby prawidłowo rozpoznawać i obsługiwać te różne typy obrazów. Wszystkie dzisiejsze oprogramowanie serwera WWW jest skonfigurowane do obsługi [PNG](#), [JPEG](#) i [GIF](#), ale jeśli używamy [SVG](#) lub jednego z nowszych formatów, może być konieczne celowe dodanie obsługi tego typu nośnika do oficjalnej listy na serwerze. Przydatne może być tutaj trochę informacji ogólnych. Pliki obrazów, a właściwie wszystkie pliki multimedialne, które mogą znajdować się na serwerze, mają oficjalny typ multimediiów (zwany również typem [MIME](#)) i sufiksy. Na przykład [SVG](#) ma typ [MIME image/svg+xml](#) oraz przyrostki [.svg](#) i [.svgz](#). Pakiety serwera mają różne sposoby obsługi informacji [MIME](#). Popularne oprogramowanie serwera [Apache](#) używa pliku w katalogu głównym o nazwie [htaccess](#), który zawiera listę wszystkich typów plików i ich dopuszczalnych przyrostków. Należy pamiętać, aby dodać (lub poprosić administratora serwera o dodanie) typów [MIME](#) nowych formatów obrazów, aby mogły być obsługiwane poprawnie. Serwer wyszukuje sufiks (np. [.webp](#)) żądanych plików na liście i dopasowuje go do [Content-Type](#) ([image/webp](#)), który zawiera w odpowiedzi [HTTP](#) do przeglądarki. To mówi przeglądarce, jakie dane nadchodzą i jak je przeanalizować. Przeglądarki używają aplikacji pomocniczych do wyświetlania multimediiów, z którymi nie mogą sobie poradzić samodzielnie. Przeglądarka dopasowuje sufiks pliku w linku do odpowiedniej aplikacji pomocniczej. Obraz zewnętrzny może otwierać się w osobnym oknie aplikacji lub w oknie przeglądarki, jeśli aplikacja pomocnicza jest wtyczką przeglądarki. Przeglądarka może również poprosić użytkownika o zapisanie pliku lub ręczne otwarcie aplikacji. Możliwe też, że w ogóle nie da się go otworzyć.

Element [img](#) mówi przeglądarce „Umieść tutaj obraz”. Możemy również umieścić element obrazu bezpośrednio w tekście w miejscu, w którym ma się pojawić obraz, jak w poniższym przykładzie. Obrazy pozostają w przepływie tekstu, wyrównane do linii bazowej tekstu i nie powodują żadnych podziałów wierszy ([HTML5](#) nazywa to elementem frazującym).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Using images</title>
</head>
<body>
    <div id="main">
        This is a PJATK  logo.
    </div>
</body>
</html>
```



This is a PJATK

logo.

Gdy przeglądarka widzi element `img`, wysyła żądanie do serwera i pobiera plik obrazu przed wyświetleniem go na stronie. W szybkiej sieci z szybkim komputerem lub urządzeniem, nawet jeśli dla każdego pliku obrazu jest wysyłane osobne żądanie, strona zwykle pojawia się natychmiast. Na urządzeniach mobilnych z wolnymi połączeniami sieciowymi możemy dobrze zdawać sobie sprawę z oczekiwania na pobranie obrazów pojedynczo. To samo dotyczy użytkowników korzystających z połączeń dial-up lub innych wolnych sieci, takich jak drogie Wi-Fi w luksusowych hotelach. Atrybuty `src` i `alt` pokazane w przykładzie **są wymagane**. Atrybut `src` (source) podaje lokalizację pliku obrazu (jego adres `URL`). Atrybut `alt` zapewnia alternatywny tekst, który jest wyświetlany, jeśli obraz jest niedostępny. Warto zwrócić uwagę na kilka innych rzeczy dotyczących elementu `img`:

- Jest to pusty element, co oznacza, że nie zawiera żadnej treści. Po prostu umieszczamy go w przepływie tekstu, w którym powinien znajdować się obraz.
- Jest to element śródliniowy, więc zachowuje się jak każdy inny element śródliniowy w przepływie tekstu. Po zmianie rozmiaru okna przeglądarki linia obrazów zmienia się, aby wypełnić nową szerokość.
- Element `img` to tak zwany element zastępowany, ponieważ podczas wyświetlania strony jest zastępowany przez plik zewnętrzny. Różni się to od elementów tekstowych, które mają swoją treść bezpośrednio w źródle (a więc nie są zastępowane).
- Domyślnie dolna krawędź obrazu jest wyrównana z linią bazową tekstu. Korzystając z CSS, możemy przesunąć obraz do prawego lub lewego marginesu i umożliwić przepływ tekstu wokół niego, przyciąć go do kształtu, kontrolować przestrzeń i obramowanie wokół obrazu oraz zmienić jego wyrównanie w pionie.

Wartość atrybutu `src` to adres `URL` pliku obrazu. W większości przypadków obrazy, których używamy na swoich stronach, będą znajdująć się na własnym serwerze, więc będziemy używać względnych adresów `URL`, aby na nie wskazywać. Deweloperzy zazwyczaj organizują obrazy dla witryny w katalogu o nazwie `images` lub `img` (w rzeczywistości pomaga to wyszukiwarem, gdy robimy to w ten sposób). Dla każdej sekcji witryny mogą istnieć nawet oddzielne katalogi obrazów. Jeśli obraz nie znajduje się w tym samym katalogu co dokument, musimy podać ścieżkę do pliku obrazu (tak jak powyżej). Oczywiście możemy umieszczać obrazy z innych witryn, używając pełnego adresu `URL`, ale nie jest to zalecane.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Using images</title>
</head>
<body>
    <div id="main">
        This is a PJATK  logo.
        <p>Now this is a  logo</p>
    </div>
</body>
</html>
```



This is a PJATK

logo.



Now this is a

logo

Gdy przeglądarka pobiera obraz, przechowuje plik w **pamięci podręcznej dysku** (miejscu do tymczasowego przechowywania plików na dysku twardym). W ten sposób, jeśli musimy ponownie wyświetlić stronę, może po prostu pobrać lokalną kopię obrazu bez tworzenia nowego żądania serwera. Jeśli wielokrotnie używamy tego samego obrazu, należy upewnić się, że atrybut `src` dla każdego elementu `img` wskazuje na ten sam adres `URL` na serwerze. Obraz jest pobierany raz, a następnie wywoływany z pamięci podręcznej do kolejnych zastosowań. Oznacza to mniejszy ruch dla serwera i szybsze wyświetlanie dla użytkownika.

Każdy element `img` **musi również zawierać atrybut `alt`**, który zapewnia tekstową alternatywę dla obrazu dla tych, którzy nie są w stanie go zobaczyć. Tekst alternatywny (zwany także tekstem alternatywnym) powinien służyć jako substytut treści obrazu – przenoszący te same informacje i funkcję. Tekst alternatywny jest używany przez czytniki ekranu, wyszukiwarki i przeglądarki graficzne, gdy obraz się nie ładuje. W przykładzie ikona `PDF` wskazuje, że połączony tekst pobiera plik w formacie `PDF`. W tym przypadku obraz przekazuje wartościową treść, której brakowałoby, gdyby nie można było zobaczyć obrazu. Dostarczenie tekstu alternatywnego „`PDF FILE`” powieła przeznaczenie obrazu.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Using images</title>
</head>
<body>
    <div id="main">
        This is a PJATK  logo.
        Now this is a  logo</p>
        <p>
            <a href="document.pdf">PDF Document</a>
            
        </p>
    </div>
</body>
</html>
```



This is a PJATK

logo.



Now this is a

logo

[PDF Document](#)

Jeśli obraz nie wnosi nic znaczącego do treści tekstu strony, zaleca się pozostawienie wartości atrybutu `alt` pustej (null). W poniższym przykładzie nie wpływa na zawartość strony, więc jego wartość `alt` jest równa `null`. W przypadku każdego wbudowanego obrazu na stronie należy zastanowić się, jak brzmiałby tekst alternatywny czytany na głos i czy poprawia to wrażenia, czy może przeszkadzać użytkownikowi korzystającemu z technologii wspomagającej. Tekst alternatywny może również przynieść korzyści użytkownikom przeglądarek graficznych. Jeśli użytkownik zdecydował się wyłączyć obrazy w preferencjach przeglądarki lub jeśli obraz po prostu się nie załaduje, przeglądarka może wyświetlić tekst alternatywny, aby dać użytkownikowi wyobrażenie o tym, czego brakuje. Obsługa tekstu alternatywnego jest jednak niespójna w nowoczesnych przeglądarkach, to znaczy każda z nich inaczej może wyświetlać tekst alternatywny.



If you're  and you know it clap your hands.

Firefox

If you're happy and you know it clap your hands.

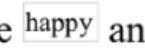
Chrome (Mac & Windows)

If you're  and you know it clap your hands.

MS Edge (Windows)

If you're  happy and you know it clap your hands.

Safari (iOS)

If you're  and you know it clap your hands.

Safari (Mac)

If you're  and you know it clap your hands.

Atrybuty `width` i `height` wskazują wymiary obrazu w liczbie pikseli. Przeglądarki używają określonych wymiarów, aby zachować odpowiednią ilość miejsca w układzie podczas ładowania obrazów, zamiast odtwarzać stronę za każdym razem, gdy pojawia się nowy obraz, co powoduje szybsze wyświetlanie strony. Jeśli ustawiony jest tylko jeden wymiar, obraz będzie skalowany proporcjonalnie. `width="liczba"` to szerokość obrazu w pikselach `height="liczba"` to wysokość obrazu w pikselach. Te atrybuty stały się mniej przydatne w dobie współczesnego tworzenia stron internetowych. Nigdy nie powinny być używane do zmiany rozmiaru obrazu (należy skorzystać z programów do edycji obrazów lub `CSS`) i powinny być całkowicie pominięte, gdy używamy jednej z technik responsywnych obrazów. Mogą być używane z obrazami, które będą wyświetlane w tym samym stałym rozmiarze na wszystkich urządzeniach, takimi jak logo lub ikona, aby dać przeglądarce wskazówkę dotyczącą układu. Należy upewnić się, że podane wymiary w pikselach są rzeczywistymi wymiarami obrazu. Jeśli wartości pikseli różnią się od rzeczywistych wymiarów obrazu, przeglądarka zmienia rozmiar obrazu, aby odpowiadał określonym wartościom. Jeśli używamy atrybutów `width` i `height`, a obraz wygląda na zniekształcony lub nawet lekko rozmazany, należy upewnić się, czy wartości są zsynchronizowane.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Using images</title>
</head>
<body>
    <div id="main">
        This is a PJATK  logo.
        <p>Now this is a  logo</p>
        <p>
            <a href="document.pdf">PDF Document</a>
            
        </p>
    </div>
</body>
</html>

```

This is a PJATK  logo.

Now this is a  logo
[PDF Document](#)

Step 6

Żadna lekcja na temat dodawania obrazów do stron internetowych nie byłaby kompletna bez wprowadzenia do dodawania **SVG (Scalable Vector Graphics)**. W końcu popularność obrazów **SVG** nabrała rozpędu dzięki niemal wszechobecnej obsłudze przeglądarek i zapotrzebowaniu na obrazy, których rozmiar można zmieniać bez utraty jakości. **SVG** są odpowiednim formatem do przechowywania obrazów wektorowych. Zamiast siatki pikseli wektory składają się z kształtów i ścieżek zdefiniowanych matematycznie. Co ciekawsze, w **SVG** te kształty i ścieżki są określane przez instrukcje zapisane w pliku tekstowym. Są to obrazy zapisane w tekście! Wszystkie kształty i ścieżki oraz ich właściwości są napisane w standardowym języku znaczników **SVG**. Ponieważ **HTML** zawiera elementy dla akapitów (**p**) i tabel (**table**), **SVG** zawiera elementy definiujące kształty, takie jak prostokąt (**rect**), okrąg (**circle**) i ścieżki (**path**). Oto kod **SVG** opisujący prostokąt (**rect**) z zaokrąglonymi rogami (**rx** i **ry**, dla **x-radius** i **y-radius**) oraz słowem **hello** ustawionym jako tekst z atrybutami dla czcionki i koloru. Przeglądarki obsługujące **SVG** czytają instrukcje i rysują obraz dokładnie tak, jak go zaprojektowaliśmy.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>SVG Example</title>
</head>
<body>
    <div id="main">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 350 250">
            <rect rx="20" ry="20" width="300" height="180" fill="purple"/>
            <text x="40" y="114" fill="yellow" font-family="Verdana-Bold" font-size="72">
                hello!
            </text>
        </svg>
    </div>
</body>
</html>

```

hello!

Pliki **SVG** oferują pewne znaczące zalety w porównaniu z ich odpowiednikami w postaci map bitowych w przypadku niektórych typów obrazów:

- Ponieważ zapisują tylko instrukcje dotyczące rysowania, zazwyczaj wymagają mniej danych niż obraz zapisany w formacie mapy bitowej. Oznacza to szybsze pobieranie i lepszą wydajność.
- Ponieważ są wektorami, można je zmieniać w razie potrzeby w responsywnym układzie bez utraty jakości. **SVG** jest zawsze ładne i ostre. Brak rozmytych krawędzi.
- Ponieważ są tekstowe, dobrze integrują się z **HTML / XML** i mogą być kompresowane za pomocą narzędzi takich jak **Gzip** i **Brotli**, podobnie jak pliki **HTML**.
- Mogą być animowane.
- Można zmienić ich wygląd za pomocą kaskadowych arkuszy stylów.
- Można dodać interaktywność z **JavaScript**, aby coś się działo, gdy użytkownicy najechali myszą lub kliknęli obraz.

Pliki tekstowe **SVG** zapisane z sufiksem **.svg** (czasami określane jako samodzielny plik **SVG**) można traktować jak każdy inny obraz, włączając go w dokument za pomocą elementu **img**.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>SVG Example</title>
</head>
<body>
  <div id="main">
    
  </div>
</body>
</html>
```

Zaletą osadzania **SVG** z **img** jest to, że jest on powszechnie obsługiwany w przeglądarkach obsługujących **SVG**. To podejście działa dobrze, gdy używamy samodzielnego **SVG** jako prostego substytutu **GIF** lub **PNG**, ale osadzanie **SVG** ma kilka wad:

- Nie można zastosować stylów do elementów w pliku **SVG** za pomocą zewnętrznego arkusza stylów, takiego jak plik **.css** zastosowany do całej strony. Plik **.svg** może zawierać własny wewnętrzny arkusz stylów używający elementu stylu, jednak do stylizowania zawartych w nim elementów. Możemy również zastosować **style** do samego elementu **img**.
- Nie możemy manipulować elementami w **SVG** za pomocą **JavaScript**, więc tracimy opcję interaktywności. Skrypty w dokumencie internetowym nie widzą zawartości **SVG**, a skrypty w pliku **SVG** w ogóle nie działają. Inne efekty interaktywne, takie jak linki lub **style :hover**, również nigdy nie są uruchamiane w **SVG** osadzonym w **img**.
- W **SVG** nie można używać żadnych plików zewnętrznych, takich jak osadzone obrazy lub czcionki internetowe.

Innymi słowy, samodzielne pliki **SVG** zachowują się tak, jakby znajdowały się we własnej małej, samodzielnej bańce.

Inną opcją umieszczenia `SVG` na stronie internetowej jest skopiowanie zawartości pliku `SVG` i wklejenie jej bezpośrednio do dokumentu `HTML`. Nazywa się to używaniem wbudowanego `SVG`. Każdy element okręgu ma atrybuty opisujące kolor wypełnienia, położenie jego punktu środkowego (`cx` i `cy`) oraz długość jego promienia (`r`).

```
<!--svg_example_2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>SVG Example 2</title>
</head>
<body>
    <div id="main">
        <p>This summer, try making
            <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 72 72" width="100" height="100">
                <circle fill="#D4AB00" cx="36" cy="36" r="36"/>
                <circle opacity=".7" fill="#FFF" stroke="#8A291C" cx="36.1" cy="35.9" r="31.2"/>
                <circle fill="#A52C1B" cx="38.8" cy="13.5" r="4.8"/>
                <circle fill="#A52C1B" cx="22.4" cy="20.9" r="4.8"/>
                <circle fill="#A52C1B" cx="32" cy="37.2" r="4.8"/>
                <circle fill="#A52C1B" cx="16.6" cy="39.9" r="4.8"/>
                <circle fill="#A52C1B" cx="26.2" cy="53.3" r="4.8"/>
                <circle fill="#A52C1B" cx="42.5" cy="27.3" r="4.8"/>
                <circle fill="#A52C1B" cx="44.3" cy="55.2" r="4.8"/>
                <circle fill="#A52C1B" cx="54.7" cy="42.9" r="4.8"/>
                <circle fill="#A52C1B" cx="56" cy="28.3" r="4.8"/>
            </svg>
            your own pizza</p>
        </div>
    </body>
</html>
```



This summer, try making your own pizza

Wbudowane pliki `SVG` umożliwiają programistom pełne wykorzystanie funkcji `SVG`. Gdy znacznik `SVG` znajduje się obok znacznika `HTML`, wszystkie jego elementy są częścią głównego drzewa `DOM`. Oznacza to, że możemy uzyskiwać dostęp do obiektów `SVG` i manipulować nimi za pomocą `JavaScript`, dzięki czemu reagują na interakcję użytkownika lub dane wejściowe. Podobne zalety mają arkusze stylów, ponieważ elementy w `SVG` mogą dziedziczyć style z elementów `HTML`. Ułatwia to stosowanie tych samych stylów do elementów na stronie i w grafice `SVG`. Z drugiej strony kod do ilustracji `SVG` może być bardzo długi i nieporęczny, co skutkuje rozdzielonymi dokumentami `HTML`, które są trudne do odczytania. Nawet ta mała pizza pepperoni wymaga poważnego bloku kodu. Utrudnia to również utrzymanie obrazów witryny, ponieważ są one schowane w dokumentach `HTML`. Inną wadą jest to, że wbudowane pliki `SVG` nie są buforowane przez przeglądarkę oddziennie od pliku `HTML`, więc należy unikać tej metody w przypadku dużych obrazów, które są ponownie używane na wielu stronach `HTML`. Dobra wiadomością jest to, że wszystkie nowoczesne przeglądarki obsługują obrazy `SVG` umieszczone w linii z elementem `svg`. Następujące starsze wersje przeglądarek nie są obsługiwane: Safari w wersji 5 i starsze, przeglądarka mobilna Android do wersji 3 oraz iOS do wersji 5.

`HTML` ma uniwersalny element osadzania mediów o nazwie `object`. Jest to dobry kompromis między `img` a wbudowanym `SVG`, pozwalający na pełni funkcjonalny `SVG`, który nadal jest zamknięty w osobnym pliku, który można buforować. Znacznik otwierający `object` określa typ nośnika (`svg+xml`) i wskazuje plik, który ma być użyty z atrybutem `data`. Element `object` ma swój własny mechanizm awaryjny – dowolna zawartość w obiekcie jest renderowana, jeśli nie można wyświetlić nośnika określonego z danymi. W takim przypadku wersja `PNG` obrazu zostanie umieszczona z obrazem, jeśli plik `.svg` nie jest obsługiwany lub nie można go załadować.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Object Example</title>
</head>
<body>
    <div id="main">
        <object type="image/svg+xml" data="/images/hello.svg">
            </object>
        </div>
    </body>
</html>

```

Główną zaletą osadzania `SVG` w elemencie `object` jest to, że można je oskryptować i ładować pliki zewnętrzne. Mogą również używać skryptów, aby uzyskać dostęp do nadzięnego dokumentu `HTML` (z pewnymi ograniczeniami bezpieczeństwa). Jednak ponieważ są to osobne pliki i nie są częścią `DOM` strony, nie można użyć arkusza stylów w dokumencie `HTML` do stylizowania elementów w `SVG`. Wbudowane pliki `SVG` mogą również wykazywać błędy w przeglądarkach, więc należy je dokładnie przetestować.

Jak wspomniano wcześniej, wszystkie nowoczesne przeglądarki obsługują pliki `SVG` osadzone jako obrazek, osadzone jako `object` lub osadzone w dokumencie, co jest bardzo dobrą wiadomością. Jeśli jednak dzienniki serwera pokazują znaczny ruch z Internet Explorera 8 i wcześniejszych, Androida w wersji 3 i wcześniejszych lub Safari 5 i wcześniejszych albo jeśli klient wymaga tylko obsługi tych przeglądarek, może być konieczne zastosowanie techniki awaryjnej. Jedną z opcji jest użycie elementu `object` do osadzenia `SVG` na stronie i skorzystanie z przedstawionej wcześniej funkcji zawartości zastępczej. Jeśli używamy `SVG` jako obrazu z elementem `img`, inną opcją jest użycie elementu `picture`. Element `picture` może służyć do udostępniania kilku wersji obrazu w różnych formatach. Każda wersja jest sugerowana z elementem `source`, który w poniższym przykładzie wskazuje na obraz `hello.svg` i definiuje jego typ nośnika. Element `picture` ma również wbudowany mechanizm awaryjny. Jeśli przeglądarka nie obsługuje sugerowanych plików źródłowych lub nie obsługuje elementu `picture`, użytkownicy zobaczą zamiast tego obraz `PNG` dostarczony ze starym dobrym elementem `img`.

```

<!--picture_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Picture example</title>
</head>
<body>
    <div id="main">
        <picture>
            <source type="image/svg+xml" srcset="/images/hello.svg">
                <img srcset="images/R.jpeg" alt="No SVG support">
        </picture>
    </div>
</body>
</html>

```

Step 7

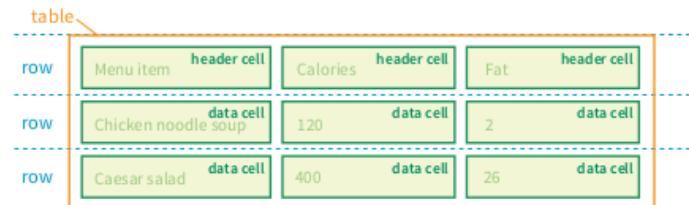
Tabele `HTML` zostały stworzone na wypadek, gdy trzeba dodać do strony internetowej materiał tabelaryczny (dane ułożone w rzędy i kolumny). Tabele mogą być używane do organizowania harmonogramów, porównań produktów, statystyk lub innych rodzajów informacji. Należy pamiętać, że „dane” niekoniecznie oznaczają liczby. Komórka tabeli może zawierać wszelkiego rodzaju informacje, w tym liczby, elementy tekstowe, a nawet obrazy i obiekty multimedialne. W przeglądarkach wizualnych rozmieszczenie danych w wierszach i kolumnach zapewnia czytelnikom natychmiastowe zrozumienie relacji między komórkami danych a ich odpowiednimi etykietami nagłówków. Podczas tworzenia tabel niektórzy czytelnicy będą słyszeć nasze dane odczytywane na głos za pomocą czytnika ekranu lub odczytywania wyników brailowskich. W czasach przed arkuszami stylów tabele były jedyną opcją tworzenia układów wielokolumnowych lub kontrolowania wyrównania i białych znaków. Jeśli potrzebujemy wierszy i kolumn do celów prezentacji, istnieją alternatywy, które wykorzystują `CSS`, aby osiągnąć pożądany efekt. W jednym podejściu, znanym jako tabele `CSS`,

zagnieżdżone elementy `div` zapewniają znacznik, a właściwości tabeli `CSS` sprawiają, że zachowują się one jak wiersze i komórki w przeglądarce. Możemy także osiągnąć wiele efektów, które wcześniej wymagały znaczników tabeli, używając technik `Flexbox` i `Grid Layout`.

Rzućmy okiem na prostą tabelę, aby zobaczyć, z czego jest ona zrobiona. Oto mała tabela z trzema wierszami i trzema kolumnami, która zawiera informacje o wartościach odżywczych.

Menu item	Calories	Fat (g)
Chicken noodle soup	120	2
Caesar salad	400	26

Możemy rozłożyć to na następujące elementy.



Teraz powyższą strukturę możemy przetłumaczyć za pomocą znaczników `HTML`.

```
<table>
<tr> <th>Menu item</th> <th>Calories</th> <th>Fat</th> </tr>
<tr> <td>Chicken noodle soup</td> <td>120</td> <td>2</td> </tr>
<tr> <td>Caesar salad</td> <td>400</td> <td>26</td> </tr>
</table>
```

Powyżej, znajdują się elementy, które identyfikują tabelę (`table`), wiersze (`tr`) i komórki (`th`, dla „nagłówków tabeli” i `td`, dla „danych tabeli”). Komórki są sercem tabeli, ponieważ tam właśnie trafia właściwa zawartość. Inne elementy po prostu trzymają rzeczy razem. To, czego nie widzimy, to elementy kolumn. Liczba kolumn w tabeli wynika z liczby komórek w każdym wierszu. Jest to jedna z rzeczy, które sprawiają, że tabele `HTML` są potencjalnie trudne. Wiersze są łatwe – jeśli chcemy, aby tabela miała trzy wiersze, po prostu należy użyć trzech elementów `tr`. Kolumny są różne. W przypadku tabeli z czterema kolumnami należy upewnić się, że każdy wiersz ma cztery elementy `td` lub `th`. Powszechnie jest układanie w stos elementów `th` i `td`, aby ułatwić ich znalezienie w źródle. Nie ma to wpływu na sposób ich renderowania przez przeglądarkę.

```

<!--table_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Table example</title>
</head>
<body>
    <div id="main">
        <table>
            <tr>
                <th>Menu item</th>
                <th>Calories</th>
                <th>Fat (g)</th>
            </tr>
            <tr>
                <td>Chicken noodle soup</td>
                <td>120</td>
                <td>2</td>
            </tr>
            <tr>
                <td>Caesar salad</td>
                <td>400</td>
                <td>26</td>
            </tr>
        </table>
    </div>
</body>
</html>

```

Menu item	Calories	Fat (g)
Chicken noodle soup	120	2
Caesar salad	400	26

Należy pamiętać, że cała zawartość musi znajdować się w komórkach – to znaczy w elementach `td` lub `th`. W komórce możemy umieścić dowolną zawartość: tekst, grafikę, a nawet inną tabelę. Tagi początku i końca tabeli identyfikują początek i koniec materiału tabelarycznego. Element tabeli może bezpośrednio zawierać tylko pewną liczbę elementów `tr` (wiersz), podpis (`caption`) i opcjonalnie elementy grupy wierszy i kolumn. Jedyne, co może się znaleźć w elemencie `tr`, to pewna liczba elementów `td` lub `th`. Innymi słowy, w elementach `table` i `tr` może nie być zawartości tekstuowej, która nie jest zawarta w `td` lub `th`. Warto zauważyć, że w przeglądarkach tabele domyślnie zawsze zaczynają się od nowej linii. Jak widać powyżej, tekst oznaczony jako nagłówki (elementy `th`) jest wyświetlany inaczej niż pozostałe komórki w tabeli (elementy `td`). Różnica nie jest jednak czysto kosmetyczna. Nagłówki tabeli są ważne, ponieważ dostarczają informacji lub kontekstu o komórkach w wierszu lub kolumnie, którą poprzedzają. Ten element może być obsługiwany inaczej niż `td` przez alternatywne urządzenia do przeglądania. Na przykład czytniki ekranu mogą odczytywać na głos nagłówki przed każdą komórką danych. W ten sposób nagłówki są kluczowym narzędziem do udostępniania zawartości tabeli. Nie próbujmy ich podrabiać, formatując wiersz elementów `td` inaczej niż reszta tabeli. I odwrotnie, nie unikajmy używania `th` elementów z powodu ich domyślnego renderowania (pogrubienie i wyśrodkowanie). Zamiast tego zaznaczmy semantycznie nagłówki i zmieńmy prezentację później za pomocą reguły stylu.

Jedną z podstawowych cech struktury tabeli jest **rozpinanie komórek**, czyli rozciąganie komórki na kilka wierszy lub kolumn. Łączenie komórek umożliwia tworzenie złożonych struktur tabel, ale powoduje to efekt uboczny polegający na utrudnieniu śledzenia znaczników. Może to również utrudnić śledzenie użytkownikom korzystającym z czytników ekranu. Rozpiętość nagłówka lub komórki danych tworzy się, dodając atrybuty `colspan` lub `rowspan`. Rozpiętości kolumn, utworzone za pomocą atrybutu `colspan` w elemencie `td` lub `th`, rozciągają komórkę w prawo, aby objąć kolejne kolumny. Tutaj rozpiętość kolumn służy do zastosowania nagłówka do dwóch kolumn.

```

<!--table_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Table example</title>
</head>
<body>
    <div id="main">
        <table>
            <tr>
                <th>Menu item</th>
                <th>Calories</th>
                <th>Fat (g)</th>
            </tr>
            <tr>
                <td>Chicken noodle soup</td>
                <td>120</td>
                <td>2</td>
            </tr>
            <tr>
                <td>Caesar salad</td>
                <td>400</td>
                <td>26</td>
            </tr>
        </table>
        <table>
            <tr>
                <th colspan="2">Fat</th>
            </tr>
            <tr>
                <td>Saturated Fat (g)</td>
                <td>Unsaturated Fat (g)</td>
            </tr>
        </table>
    </div>
</body>
</html>

```

Menu item	Calories	Fat (g)
Chicken noodle soup	120	2
Caesar salad	400	26
Fat		
Saturated Fat (g)		Unsaturated Fat (g)

Zauważmy, że w pierwszym wierszu (`tr`) jest tylko jeden element, podczas gdy drugi wiersz ma dwa elementy `td`. `th` dla kolumny, która została rozciągnięta, nie znajduje się już w źródle; zastępuje ją komórka z `colspan`. Każdy wiersz powinien mieć taką samą liczbę komórek lub równoważne wartości `colspan`. Na przykład istnieją dwa elementy `td`, a wartość `colspan` wynosi 2, więc domniemana liczba kolumn w każdym wierszu jest równa.

Rozpiętości wierszy, utworzone za pomocą atrybutu `rowspan`, działają tak samo jak zakresy kolumn, ale powodują, że komórka rozciąga się w dół na kilka wierszy.

```

<!--table_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Table example</title>
</head>
<body>
    <div id="main">
        <table>
            <tr>
                <th>Menu item</th>
                <th>Calories</th>
                <th>Fat (g)</th>
            </tr>
            <tr>
                <td>Chicken noodle soup</td>
                <td>120</td>
                <td>2</td>
            </tr>
            <tr>
                <td>Caesar salad</td>
                <td>400</td>
                <td>26</td>
            </tr>
        </table>
        <hr>
        <table>
            <tr>
                <th colspan="2">Fat</th>
            </tr>
            <tr>
                <td>Saturated Fat (g)</td>
                <td>Unsaturated Fat (g)</td>
            </tr>
        </table>
        <hr>
        <table>
            <tr>
                <th rowspan="3">Serving Size</th>
                <td>Small (8oz.)</td>
            </tr>
            <tr>
                <td>Medium (16oz.)</td>
            </tr>
            <tr>
                <td>Large (24oz.)</td>
            </tr>
        </table>
    </div>
</body>
</html>

```

	Menu item	Calories	Fat (g)
	Chicken noodle soup	120	2
	Caesar salad	400	26

Fat	
Saturated Fat (g)	Unsaturated Fat (g)

Small (8oz.)
Serving Size Medium (16oz.)
Large (24oz.)

Elementy `td` dla komórek, które zostały rozciągnięte (pierwsze komórki w pozostałych wierszach) nie pojawiają się w źródle. `Rowspan="3"` implikuje komórki dla kolejnych dwóch wierszy, więc nie są potrzebne żadne elementy `td`.

Domyślnie tabele rozszerzają się na tyle, aby dopasować się do zawartości komórek, które mogą wyglądać na trochę ciasne. Stare wersje HTML zawierały atrybuty cellpadding i cellspacing w celu dodawania miejsca w komórkach i między komórkami, ale zostały one wyrzucone z HTML5, ponieważ są przestarzałymi, prezentacyjnymi znacznikami. Właściwym sposobem dostosowania odstępów między komórkami tabeli jest oczywiście użycie arkuszy stylów.

Jako projektant stron internetowych ważne jest, abyśmy zawsze pamiętały, w jaki sposób zawartość witryny będzie wykorzystywana przez odwiedzających z wadami wzroku. Szczególnie trudne jest zrozumienie materiału tabelarycznego za pomocą czytnika ekranu, ale specyfikacja HTML zapewnia środki poprawiające wrażenia i sprawiające, że treść jest bardziej zrozumiałą. Najsłutecznieszym sposobem, aby użytkownicy z dysfunkcją wzroku mogli zapoznać się z tabelą, jest nadanie jej tytułu lub opisu z elementem caption. Napisy są wyświetlane obok tabeli (na ogół nad nią) i mogą służyć do opisywania zawartości tabeli lub udzielania wskazówek dotyczących jej struktury. W przypadku użycia, element caption musi być pierwszą rzeczą w elemencie tabeli.

```
<!--table_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Table example</title>
</head>
<body>
    <div id="main">
        <table>
            <caption>Nutritional Information</caption>
            <tr>
                <th>Menu item</th>
                <th>Calories</th>
                <th>Fat (g)</th>
            </tr>
            <tr>
                <td>Chicken noodle soup</td>
                <td>120</td>
                <td>2</td>
            </tr>
            <tr>
                <td>Caesar salad</td>
                <td>400</td>
                <td>26</td>
            </tr>
        </table>
        <hr>
        <table>
            <tr>
                <th colspan="2">Fat</th>
            </tr>
            <tr>
                <td>Saturated Fat (g)</td>
                <td>Unsaturated Fat (g)</td>
            </tr>
        </table>
        <hr>
        <table>
            <tr>
                <th rowspan="3">Serving Size</th>
                <td>Small (8oz.)</td>
            </tr>
            <tr>
                <td>Medium (16oz.)</td>
            </tr>
            <tr>
                <td>Large (24oz.)</td>
            </tr>
        </table>
    </div>
</body>
</html>
```

Nutritional Information		
Menu item	Calories	Fat (g)
Chicken noodle soup	120	2
Caesar salad	400	26
Fat		
Saturated Fat (g) Unsaturated Fat (g)		
Small (8oz.)		
Serving Size Medium (16oz.)		
Large (24oz.)		

Podpis jest domyślnie wyświetlany nad tabelą, chociaż można użyć właściwości arkusza stylów, aby przenieść go pod tabelę (`caption-side: bottom`). W przypadku dłuższych opisów można rozważyć umieszczenie tabeli w elemencie `figure` i użycie elementu `figcaption` do opisu. Specyfikacja HTML5 zawiera szereg sugestii dotyczących dostarczania opisów tabel, dostępnych pod adresem www.w3.org/TR/html5/tabular-data.html#table-descriptions-techniques (www.w3.org/TR/html5/tabular-data.html#table-descriptions-techniques).

Step 8

Krótko omówiliśmy nagłówki jako prostą metodę poprawy dostępności zawartości tabeli, ale czasami może być trudno określić, który nagłówek dotyczy jakich komórek. Na przykład nagłówki mogą znajdować się na lewej lub prawej krawędzi wiersza, a nie na górze kolumny. I chociaż widzący użytkownicy mogą łatwo zrozumieć strukturę tabeli na pierwszy rzut oka, dla użytkowników słyszących dane jako tekst, ogólna organizacja nie jest tak jasna. Atrybuty `scope` i `headers` umożliwiają autorom jawne skojarzenie nagłówków i ich odpowiedniej zawartości. Atrybut `scope` kojarzy nagłówek tabeli z wierszem, kolumną, grupą wierszy (na przykład `tbody`) lub grupą kolumn, w której występuje, przy użyciu odpowiednio wartości `row`, `col`, `rowgroup` lub `colgroup`.

```
<!--table_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Table example 2</title>
</head>
<body>
  <div id="main">
    <table>
      <tr>
        <th scope="row">Mars</th>
        <td>.95</td>
        <td>.62</td>
        <td>0</td>
      </tr>
    </table>
  </div>
</body>
</html>
```

Mars .95 .62 0

Eksperci ds. ułatwień dostępu zalecają, aby każdy element `th` zawierał atrybut `scope`, aby skojarzone z nim dane były wyraźnie jasne.

W przypadku naprawdę skomplikowanych tabel, w których `scope` nie jest wystarczający do powiązania komórki danych tabeli z odpowiednim nagłówkiem (np. gdy tabela zawiera wiele połączonych komórek), atrybut `headers` jest używany w elemencie `td`, aby jawnie powiązać ją z nagłówkiem wartości identyfikatora.

```

<!--table_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Table example 2</title>
</head>
<body>
    <div id="main">
        <table>
            <tr>
                <th scope="row">Mars</th>
                <td>.95</td>
                <td>.62</td>
                <td>0</td>
            </tr>
        </table>
        <hr>
        <table>
            <th id="diameter">One</th>
            <th id="diameter2">Two</th>
            <th id="diameter3">Three</th>
            <tr>
                <td headers="diameter">.38</td>
                <td headers="diameter3">.78</td>
                <td headers="diameter2">.01</td>
            </tr>
        </table>
    </div>
</body>
</html>

```

Mars .95 .62 0

One Two Three

.38 .78 .01

Niestety obsługa funkcji identyfikatora/nagłówków jest zawodna. Zalecaną najlepszą praktyką jest tworzenie tabel w taki sposób, aby prosty atrybut `scope` wykonał to zadanie. Dogłębne instrukcje dotyczące tworzenia dostępnych tabel są na WebAIM pod adresem (webaim.org/techniques/tables/data (webaim.org/techniques/tables/data)) jako doskonały punkt wyjścia.

Przykładowe tabele, którym przyglądaliśmy się do tej pory, zostały zredukowane do podstawowych elementów, aby struktura była przezroczysta, gdy uczymy się, jak działają tabele. Ale tabele w prawdziwym świecie nie zawsze są takie proste. Poniżej mamy przykład takiej tabeli. Możemy zidentyfikować trzy grupy kolumn (jedna z nagłówkami, dwie z dwiema kolumnami każda) oraz trzy grupy wierszy (nagłówki, dane i przypis). Koncepcyjne grupy tabel, takie jak te, są oznaczone elementami **grup wierszy** i **grup kolumn**, które zapewniają dodatkową strukturę semantyczną i więcej „haczyków” do stylizacji lub tworzenia skryptów. Na przykład grupy wierszy i kolumn na poniższym przykładzie zostały stylizowane z grubszymi obramowaniami, aby wyróżniały się wizualnie.

Bidi control codes injected by <code>'unicode-bidi'</code> at the start/end of <code>'display: inline'</code> boxes				
'unicode-bidi' value	'ltr'		'rtl'	
	start	end	start	end
'normal'	—	—	—	—
'embed'	LRE (U+202A)	PDF (U+202C)	RLE (U+202B)	PDF (U+202C)
'isolate'	LRI (U+2066)	PDI (U+2069)	RLI (U+2067)	PDI (U+2069)
'bidi-override'*	LRO (U+202D)	PDF (U+202C)	RLO (U+202E)	PDF (U+202C)
'isolate-override'*	FSI,LRO (U+2068,U+202D)	PDF,PDI (U+202C,U+2069)	FSI,RLO (U+2068,U+202E)	PDF,PDI (U+202C,U+2069)
'plaintext'	FSI (U+2068)	PDI (U+2069)	FSI (U+2068)	PDI (U+2069)

* The LRO/RLO+PDF pairs are also applied to the root inline box of a block container if these values of `'unicode-bidi'` were specified on the block container.

Możemy opisywać wiersze lub grupy wierszy jako należące do nagłówka, stopki lub treści tabeli, używając odpowiednio elementów `thead`, `tfoot` i `tbody`. Niektóre programy użytkownika (inne słowo oznaczające przeglądarkę) mogą powtarzać wiersze nagłówka i stopki w tabelach obejmujących wiele stron. Na przykład wiersze nagłówka i stopy mogą być drukowane na każdej stronie tabeli wielostroanicowej. Autorzy mogą również używać tych elementów do stosowania stylów w różnych regionach tabeli. Elementy grupy wierszy mogą zawierać tylko jeden lub więcej elementów `tr`. Nie zawierają bezpośrednio treści tekstu. Element `thead` powinien pojawić się jako pierwszy, po nim dowolna liczba elementów `tbody`, a następnie opcjonalna `tfoot`.

```
<!--table_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Table example 2</title>
</head>
<body>
    <div id="main">
        <table>
            <tr>
                <th scope="row">Mars</th>
                <td>.95</td>
                <td>.62</td>
                <td>0</td>
            </tr>
        </table>
        <hr>
        <table>
            <th id="diameter">One</th>
            <th id="diameter2">Two</th>
            <th id="diameter3">Three</th>
            <tr>
                <td headers="diameter">.38</td>
                <td headers="diameter3">.78</td>
                <td headers="diameter2">.01</td>
            </tr>
        </table>
        <hr>
        <table>
            <thead>
                <th>One</th>
                <th>Two</th>
                <th>Three</th>
            </tr></thead>
            <tr></tr>
            <tr></tr>
            <tbody>
                <tr>
                    <td>1</td>
                    <td>2</td>
                    <td>3</td>
                </tr>
                <tr>
                    <td>5</td>
                    <td>6</td>
                    <td>7</td>
                </tr>
            </tbody>
            <tfoot>
                <tr>Footer</tr>
            </tfoot>
        </table>
    </div>
</body>
</html>
```

One Two Three

.38 .78 .01

Footer

One Two Three

1	2	3
5	6	7

Jak już wiemy, kolumny są implikowane przez liczbę komórek (`td` lub `th`) w każdym wierszu. Kolumny można grupować semantycznie (oraz przypisywać wartości `id` i `class`) za pomocą elementu `colgroup`. Grupy kolumn są identyfikowane na początku tabeli, zaraz po `caption`, jeśli taki istnieje, i dają przeglądarce trochę informacji na temat układu kolumn w tabeli. Liczba kolumn reprezentowanych przez `colgroup` jest określona w atrybutie `span`. Jeśli potrzebujemy uzyskać dostęp do poszczególnych kolumn w grupie `col` do tworzenia skryptów lub stylów, należy je zidentyfikować za pomocą elementów `col`.

```
<!--table_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Table example 2</title>
</head>
<body>
    <div id="main">
        <table>
            <tr>
                <th scope="row">Mars</th>
                <td>.95</td>
                <td>.62</td>
                <td>0</td>
            </tr>
        </table>
        <hr>
        <table>
            <th id="diameter">One</th>
            <th id="diameter2">Two</th>
            <th id="diameter3">Three</th>
            <tr>
                <td headers="diameter">.38</td>
                <td headers="diameter3">.78</td>
                <td headers="diameter2">.01</td>
            </tr>
        </table>
        <hr>
        <table>
            <thead>
                <th>One</th>
                <th>Two</th>
                <th>Three</th>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            </thead>
            <tbody>
                <tr>
                    <td>1</td>
                    <td>2</td>
                    <td>3</td>
                </tr>
                <tr>
                    <td>5</td>
                    <td>6</td>
                    <td>7</td>
                </tr>
            </tbody>
            <tfoot>
                <tr>Footer</tr>
            </tfoot>
        </table>
        <hr>
        <table>
            <colgroup>
                <col span="2">
                <col style="background-color:yellow">
            </colgroup>
            <tr>
                <th>ISBN</th>
                <th>Title</th>
                <th>Price</th>
            </tr>
            <tr>
                <td>3476896</td>
                <td>My first HTML</td>
                <td>$53</td>
            </tr>
            <tr>
                <td>5869207</td>
                <td>My first CSS</td>
            </tr>
        </table>
    </div>
</body>
```

```

<td>$49</td>
</tr>
</table>
</div>
</body>
</html>

```

Mars .95 .62 0

One Two Three

.38 .78 .01

Footer

One Two Three

1	2	3
5	6	7

ISBN	Title	Price
3476896	My first HTML	\$53
5869207	My first CSS	\$49

Zwróćmy uwagę, że elementy `colgroup` nie zawierają treści – dostarczają jedynie informacji o semantycznie istotnej strukturze kolumn. Puste elementy są używane jako uchwyty skryptów lub stylów, ale nie są wymagane.

Step 9

Nie minęło dużo czasu, zanim sieć przeniosła się z sieci stron do czytania do miejsca, do którego chodzimy, aby załatwić różne sprawy – robić zakupy, rezerwować bilety lotnicze, podpisywać petycje, przeszukiwać witryny, publikować tweety... lista jest długa! Formularze internetowe obsługują wszystkie te interakcje. W rzeczywistości, w odpowiedzi na to przejście od strony do aplikacji, `HTML5` wprowadził mnóstwo nowych kontrolek formularzy i atrybutów, które ułatwiają użytkownikom wypełnianie formularzy, a programistom ich tworzenie. Zadania, które tradycyjnie opierały się na `JavaScript`, mogą być obsługiwane wyłącznie przez znaczniki i zachowanie natywne przeglądarki. `HTML5` wprowadza wiele nowych elementów związanych z formularzami, 12 nowych typów danych wejściowych i wiele nowych atrybutów.

Formularz roboczy składa się z dwóch części. Pierwsza część to formularz, który widzimy na samej stronie, który jest tworzony przy użyciu znaczników `HTML`. Formularze składają się z przycisków, pól wejściowych i menu rozwijanych (zwanych łącznicami kontrolkami formularza) używanych do zbierania informacji od użytkownika. Formularze mogą również zawierać tekst i inne elementy. Drugim składnikiem formularza internetowego jest aplikacja lub skrypt na serwerze, który przetwarza informacje zebrane przez formularz i zwraca odpowiednią odpowiedź. To właśnie sprawia, że formularz działa. Innymi słowy, opublikowanie dokumentu `HTML` z elementami formularza nie wystarczy. Jeśli zamierzamy tworzyć formularze internetowe, dobrze jest zrozumieć, co dzieje się za kulissami. Ten przykład śledzi kroki transakcji przy użyciu prostego formularza, który zbiera nazwiska i adresy e-mail dla listy adresowej; jednak jest to typowe dla procesu dla wielu form.

1. Twój gość – nazwijmy ją Jan – otwiera stronę z formularzem internetowym w oknie przeglądarki. Przeglądarka widzi elementy sterujące formularza w znacznikach i renderuje je z odpowiednimi kontrolkami formularza na stronie, w tym dwoma polami do wprowadzania tekstu i przyciskiem Wyślij.
2. Jan chciałby zapisać się na tę listę mailingową, więc wpisuje swoje imię i nazwisko oraz adres e-mail w polach i przesyła formularz, naciskając przycisk Wyślij.
3. Przeglądarka zbiera wprowadzone przez nią informacje, koduje je i przesyła go do aplikacji internetowej na serwerze.
4. Aplikacja internetowa akceptuje informacje i przetwarza je (to znaczy robi z nimi wszystko, co jest zaprogramowane). W tym przykładzie imię i nazwisko oraz adres e-mail są dodawane do bazy danych listy adresowej.
5. Aplikacja internetowa również zwraca odpowiedź. Rodzaj odesłanej odpowiedzi zależy od treści i przeznaczenia formularza. Tutaj odpowiedzią jest prosta strona internetowa z podziękowaniem za zapisanie się na listę mailingową. Inne aplikacje mogą odpowiedzieć, przeładowując stronę formularza ze zaktualizowanymi informacjami, przenosząc użytkownika na inną powiązaną stronę formularza lub wyświetlając komunikat o błędzie, jeśli formularz nie jest wypełniony poprawnie.
6. Serwer odsyła odpowiedź aplikacji internetowej z powrotem do przeglądarki, gdzie jest wyświetlana. Jan widzi, że formularz zadziałał i że został dodany do listy mailingowej.

Formularze są dodawane do stron internetowych za pomocą elementu `form`. Element `form` jest pojemnikiem na całą zawartość formularza, w tym pewną liczbę kontrolek formularza, takich jak pola wprowadzania tekstu i przyciski. Może również zawierać elementy blokowe (na przykład `h1`, `p` i listy). **Nie może jednak zawierać innego elementu formularza.**

```
<!--form_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example</title>
</head>
<body>
    <div id="main">
        <h1>Mailing List Signup</h1>
        <form action="/mailinglist.php" method="POST">
            <fieldset>
                <legend>Join our email list</legend>
                <p>Get news about the band such as tour dates and special MP3 releases sent to your own in-box.</p>
                <ol>
                    <li><label for="firstlast">Name:</label>
                        <input type="text" name="fullname" id="firstlast"></li>
                    <li><label for="email">Email:</label>
                        <input type="text" name="email" id="email"></li>
                </ol>
                <input type="submit" value="Submit">
            </fieldset>
        </form>
    </div>
</body>
</html>
```

Mailing List Signup

Join our email list

Get news about the band such as tour dates and special MP3 releases sent to your own in-box.

1. Name:

2. Email:

Element `form` ma pewne atrybuty, które są niezbędne do interakcji z programem do przetwarzania formularzy na serwerze. Przyjrzymy się każdemu. Atrybut `action` podaje lokalizację (URL) aplikacji lub skryptu, który zostanie użyty do przetworzenia formularza. Atrybut `action` w tym przykładzie wysyła dane do skryptu o nazwie `mailinglist.php`. Sufiks `.php` wskazuje, że ten formularz jest przetwarzany przez skrypt napisany w języku skryptowym `PHP`, ale formularze internetowe mogą być przetwarzane za pomocą dowolnej z następujących technologii: `Microsoft ASP (.asp)`, `Microsoft's ASP.NET (.aspx)`, `Ruby on Rails`, `JavaServer Pages (.jsp)`, `Python` itd. Istnieją inne opcje przetwarzania formularzy, które mogą mieć własne sufiksy lub wcale (tak jak w przypadku platformy `Ruby on Rails`). Wówczas należy się skontaktować z programistą, administratorem serwera lub dokumentacją skryptu, aby uzyskać prawidłową nazwę i lokalizację programu, który ma być podany w atrybucie `action`. Czasami w pliku `HTML` jest osadzony kod przetwarzania formularzy, taki jak `PHP`. W takim przypadku należy pozostawić atrybut `action` z wartością pustą, czyli `action=""`, a formularz zostanie opublikowany na samej stronie.

Atrybut `method` określa sposób przesyłania informacji do serwera. Wykorzystajmy te dane zebrane z przykładowego formularza.

```
fullname = Mateusz Miotk
email = mmiotk@pjwstk.edu.pl
```

Kiedy przeglądarka koduje te informacje na potrzeby swojej podróży na serwer, wygląda to tak.

```
fullname=Mateusz+Miotk&email=mmiotk%40pjwstk.edu.pl
```

Istnieją tylko dwie metody przesyłania tych zaszyfrowanych danych na serwer: `POST` lub `GET`, wskazane przez atrybut `method` w elemencie formularza. Metoda jest opcjonalna i będzie domyślnie `GET`, jeśli zostanie pominięta. Dzięki metodzie `GET` zakodowane dane formularza są dołączane bezpośrednio do adresu `URL` wysyłanego do serwera. Znak zapytania oddziela adres `URL` od następujących danych.

```
<!--form_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example</title>
</head>
<body>
    <div id="main">
        <h1>Mailing List Signup</h1>
        <form action="/mailinglist.php" method="get">
            <fieldset>
                <legend>Join our email list</legend>
                <p>Get news about the band such as tour dates and special MP3 releases sent to your own in-box.</p>
                <ol>
                    <li><label for="firstlast">Name:</label>
                        <input type="text" name="fullname" id="firstlast"></li>
                    <li><label for="email">Email:</label>
                        <input type="text" name="email" id="email"></li>
                </ol>
                <input type="submit" value="Submit">
            </fieldset>
        </form>
    </div>
</body>
</html>
```

```
mailinglist.php?fullname=Mateusz+Miotk&email=mmiotk%40pjwstk.edu.pl
```

Gdy metoda formularza jest ustawiona na `POST`, przeglądarka wysyła osobne żądanie serwera zawierające kilka specjalnych nagłówków, po których następują dane. Teoretycznie tylko serwer widzi treść tego żądania, dlatego jest to najlepsza metoda przesyłania bezpiecznych informacji, takich jak adres domowy lub inne dane osobowe. W praktyce upewnijmy się, że na serwerze włączony jest protokół `HTTPS`, aby dane użytkownika były szyfrowane i niedostępne podczas przesyłania. Metoda `POST` jest również preferowana w przypadku wysyłania dużej ilości danych, na przykład długich wpisów tekstowych, ponieważ nie ma limitu znaków, jak w przypadku `GET`. Metoda `GET` jest odpowiednia, jeśli chcemy, aby użytkownicy mogli dodawać zakładki do wyników przesłania formularza (takich jak lista wyników wyszukiwania). Ponieważ treść formularza jest widoczna, `GET` nie jest odpowiedni dla formularzy zawierających prywatne dane osobowe lub finansowe. Ponadto `GET` nie może być używany, gdy formularz służy do przesyłania pliku. W `POST` i `GET` nie jest rozróżniana wielkość liter i zgodnie z konwencją są one zwykle pisane wielkimi literami. Jednak w dokumentach `XHTML` wartość atrybutu metody (`post` lub `get`) musi być podana małymi literami.

Step 10

Formularze internetowe wykorzystują różne kontrolki, które umożliwiają użytkownikom wprowadzanie informacji lub wybieranie opcji. Typy kontrolek obejmują różne pola wprowadzania tekstu, przyciski, menu i kilka kontrolek ze specjalnymi funkcjami. Są one dodawane do dokumentu wraz z kolekcją elementów kontroli formularzy, które będziemy analizować jeden po drugim. Jako projektant stron internetowych musimy znać opcje kontroli, aby formularze były łatwe i intuicyjne w użyciu. Warto również mieć pojęcie o tym, co kontrolki formularza robią za kulisami. Zadaniem każdej kontrolki formularza jest zebranie jednego bitu informacji od użytkownika. W poprzednim przykładzie pola do wprowadzania tekstu zbierają imię i nazwisko odwiedzającego oraz adres e-mail. Używając terminu technicznego, `fullname` i `email` to dwie zmienne zbierane przez formularz. Dane wprowadzone przez użytkownika (`Mateusz Miotk` i `mmiotk@pjwstk.edu.pl`) są wartością lub zawartością zmiennych. Atrybut `name` zapewnia nazwę zmiennej dla kontrolki. W tym przykładzie tekst zebrany przez element `textarea` jest zdefiniowany jako zmienna `comment`.

```

<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here"></textarea>
            <input type="submit" value="Send">
        </form>
    </div>
</body>
</html>

```

The screenshot shows a simple HTML form. It consists of a text area with the placeholder "Leave comment here" and a submit button labeled "Send".

Gdy użytkownik wprowadzi komentarz w polu (`This is the best band ever`), zostanie on przekazany do serwera jako para nazwa/wartość (zmienna/treść) w następujący sposób.

```
form_example2.html?comment=This+is+the+best+band+ever
```

Wszystkie elementy sterujące formularza muszą zawierać atrybut `name`, aby aplikacja przetwarzająca formularze mogła sortować informacje. Możemy dodać atrybut `name` dla elementów przycisku przesyłania i resetowania, ale nie są one wymagane, ponieważ mają specjalne funkcje (przesyłanie lub resetowanie formularza) niezwiązane z gromadzeniem danych.

Nie możemy po prostu nazwać kontrolek chcąc nie chcąc. Aplikacja internetowa przetwarzająca dane jest zaprogramowana do wyszukiwania określonych nazw zmiennych. Jeśli projektujemy formularz do pracy z istniejącą już aplikacją lub skryptem, musimy znaleźć konkretne nazwy zmiennych, które będą używane w formularzu, aby mówiły tym samym językiem. Nazwy zmiennych można uzyskać z instrukcji dostarczonych z gotowym do użycia skryptem na serwerze, administratorem systemu lub programistą, z którym pracujemy. Jeśli skrypt lub aplikacja zostanie utworzona później, pamiętajmy, aby nazwać zmienne w prosty i opisowy sposób oraz dobrze je udokumentować. Ponadto, aby uniknąć nieporozumień, zaleca się nadawanie niepowtarzalnej nazwy każdej zmiennej – to znaczy nie należy używać tej samej nazwy dla dwóch zmiennych (jednak mogą istnieć wyjątki, dla których jest to pożądane). Należy również unikać umieszczania spacji znakowych w nazwach zmiennych. Zamiast tego użyjmy podkreślenia lub łącznika.

Większość kontrolek dodawana jest do formularza poprzez element `input`. Funkcjonalność i wygląd elementu `input` zmienia się w zależności od wartości atrybutu `type` w znaczniku. W [HTML5.2](#) istnieją **dzwadzieścia dwa** rodzaje kontrolek wejściowych.

Jednym z najczęstszych zadań formularzy internetowych jest wprowadzanie informacji tekstowych. To, którego elementu używamy do zbierania danych wejściowych, zależy od tego, czy użytkownicy są proszeni o wprowadzenie pojedynczego wiersza tekstu (`input`) czy wielu wierszy (`textarea`). Należy pamiętać, że jeśli formularz zawiera pola do wprowadzania tekstu, musi korzystać z bezpiecznego protokołu `HTTPS` w celu ochrony treści wprowadzonych przez użytkownika podczas przesyłania ich danych na serwer. Jednym z najprostszych typów wprowadzania w formularzu jest **pole wprowadzania tekstu** służące do wprowadzania pojedynczego słowa lub wiersza tekstu. W rzeczywistości jest to domyślny typ wejściowy, co oznacza, że otrzymamy go, jeśli zapomnimy dodać atrybut `type` lub dodamy nierożpoznaną wartość. Dodajmy pole tekstowe do formularza, wstawiając element wejściowy z atrybutem `type` ustawionym na `text`.

```

<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">
                </textarea></label></li>
                <li><input type="submit" value="Send"></li>
            </ul>
        </form>
    </div>
</body>
</html>

```

- Favorite color:
- Comment:
-

Jest tam kilka atrybutów, na które należy zwrócić uwagę. Atrybut `name` jest wymagany do wskazania nazwy zmiennej. Atrybut `value` określa domyślny tekst, który pojawia się w polu podczas ładowania formularza. Po zresetowaniu formularza powraca on do tej wartości. Wartość atrybutu `value` zostaje przesłana do serwera, więc w tym przykładzie wartość `Red` zostanie wysłana wraz z formularzem, chyba że użytkownik ją zmieni. Alternatywnie możemy użyć atrybutu `placeholder`, aby podpowiedzieć, co wpisać w polu, na przykład `My favorite color`. Wartość `placeholder` nie jest przesyłana z formularzem i stanowi jedynie ulepszenie interfejsu użytkownika. Domyślnie użytkownicy mogą wpisywać nieograniczoną liczbę znaków w polu tekstowym, niezależnie od jego wielkości (wyświetlacz przewija się w prawo, jeśli tekst przekracza szerokość znaku w polu). Możemy ustawić maksymalny limit znaków za pomocą atrybutu `maxlength`, jeśli wymaga tego program do przetwarzania formularzy. Atrybut `minlength` określa minimalną liczbę znaków. Atrybut `size` określa długość pola wejściowego w liczbie widocznych znaków. Jednak bardziej powszechnie jest używanie arkuszy stylów do ustawiania rozmiaru obszaru wejściowego. Domyślnie widżet do wprowadzania tekstu jest wyświetlany w rozmiarze mieszczącym 20 znaków.

Czasami będziemy chcieli, aby użytkownicy mogli wpisać więcej niż jeden wiersz tekstu. W takich przypadkach należy użyć elementu `textarea`, który podczas wyświetlania w przeglądarce jest zastępowany wielowierszowym, przewijalnym polem wprowadzania tekstu. W przeciwieństwie do pustego elementu `input`, możemy umieścić zawartość między znacznikami otwierającymi i zamykającymi w elemencie `textarea`. Zawartość elementu `textarea` pojawia się w polu tekstowym, gdy formularz jest wyświetlany w przeglądarce. Jest również wysyłany na serwer po przesłaniu formularza. Atrybuty `rows` i `cols` umożliwiają określenie rozmiaru obszaru tekstowego za pomocą znaczników. `rows` określa liczbę wierszy, które powinien wyświetlić obszar tekstowy, a `cols` określa szerokość w liczbie znaków (chociaż częściej używa się CSS do określenia szerokości pola). Paski przewijania zostaną udostępnione, jeśli użytkownik wpisze więcej tekstu niż mieści się w przydzielonym miejscu. Istnieje również kilka atrybutów, których nie pokazano w przykładzie. Atrybut `wrap` określa, czy miękkie łamanie linii (gdzie tekst naturalnie zawija się na krawędzi pola) są zachowywane podczas przesyłania formularza. Wartość `soft` (domyślna) nie zachowuje łamania wierszy. Wartość `hard` zachowuje podziały wierszy, gdy atrybut `cols` jest używany do ustawienia szerokości znaku w polu. Atrybuty `maxLength` i `minLength` określają maksymalną i minimalną liczbę znaków, które można wpisać w pole. Często zdarza się, że programiści nie umieszczają niczego między znacznikami otwierającym i zamykającym, a zamiast tego podpowiadają, co powinno się tam znaleźć, za pomocą `placeholder`. Tekst zastępczy, w przeciwieństwie do zawartości obszaru tekstowego, nie jest wysyłany na serwer po przesłaniu formularza.

```

<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><input type="submit" value="Send"></li>
            </ul>
        </form>
    </div>
</body>
</html>

```

- Favorite color:
- Comment:
-

Oprócz ogólnych jednowierszowych wpisów tekstowych istnieje kilka typów danych wejściowych do wprowadzania określonych typów informacji, takich jak hasła, wyszukiwane słowa, adresy e-mail, numery telefonów i adresy URL. Pole `password` działa tak samo jak pole tekstowe, z wyjątkiem tego, że znaki są zasłonięte gwiazdką (*), punktorami (•) lub innym znakiem określonym przez przeglądarkę. Należy pamiętać, że chociaż znaki wpisane w polu hasła nie są widoczne dla przypadkowych osób, formularz nie szyfruje informacji, więc nie należy tego traktować jako prawdziwego środka bezpieczeństwa.

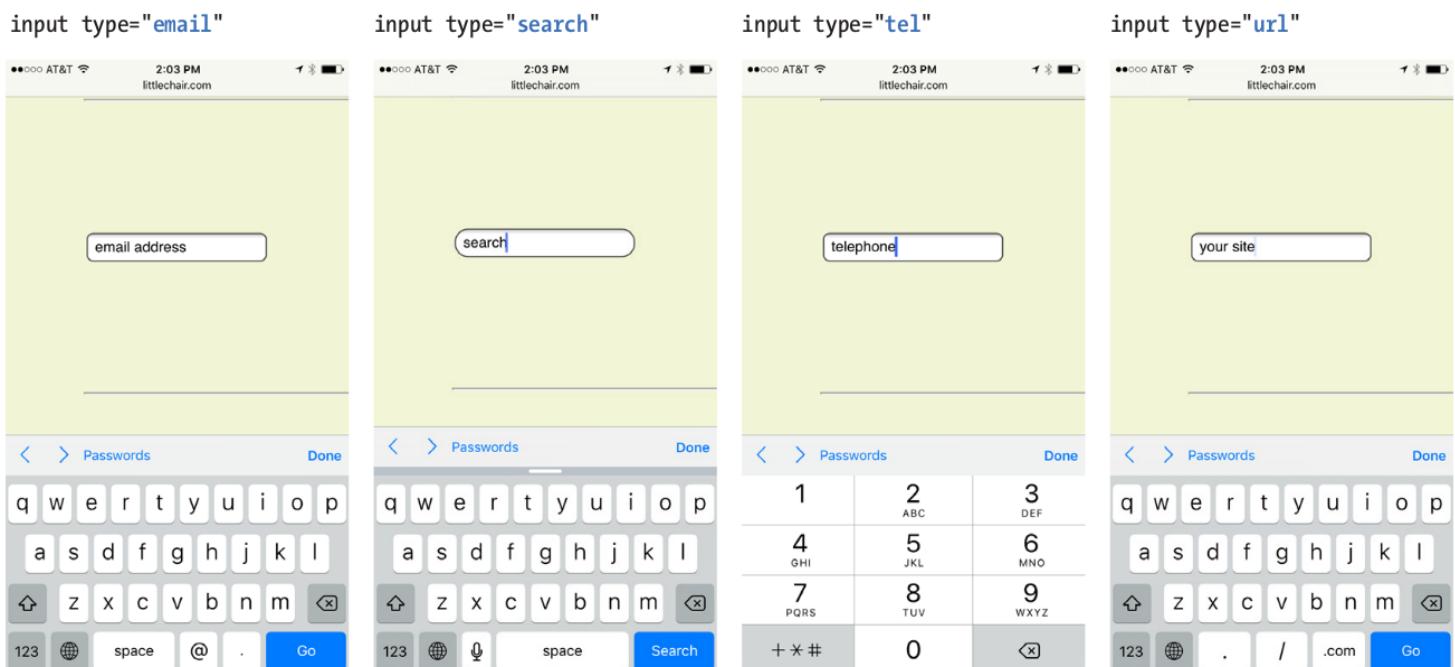
```

<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><input type="submit" value="Send"></li>
            </ul>
        </form>
    </div>
</body>
</html>

```

- Favorite color:
- Password:
- Comment:
-

Przed HTML5 jedynym sposobem zbierania adresów e-mail, numerów telefonów, adresów URL lub wyszukiwanych haseł było wstawianie ogólnego pola tekstowego. W HTML5 typy danych wejściowych `e-mail`, `tel`, `url` i `search` informują przeglądarkę, jakiego rodzaju informacji można się spodziewać w danej dziedzinie. Te typy danych wejściowych wykorzystują te same atrybuty, co ogólny tekstowy typ wejściowy opisany wcześniej (`name`, `maxlength`, `minlength`, `size` i `value`), a także szereg innych atrybutów. Wszystkie te typy danych wejściowych są zwykle wyświetlane jako jednowierszowe dane wejściowe tekstowe. Ale przeglądarki, które je obsługują, mogą robić ciekawe rzeczy z dodatkowymi informacjami semantycznymi. Na przykład Safari w systemie iOS używa typu `input`, aby zapewnić klawiaturę dobrze dopasowaną do zadania, taką jak klawiatura z przyciskiem Szukaj dla typu wprowadzania wyszukiwania lub przyciskiem `.com`, gdy typ wprowadzania jest ustawiony na adres `URL`. Przeglądarki zwykle dodają ikonę „wyczyść pole” jednym kliknięciem (zwykle mały `X`) w polach `search`. Obsługiwana przeglądarka może sprawdzić dane wprowadzone przez użytkownika, aby sprawdzić, czy są prawidłowe – na przykład poprzez upewnienie się, że tekst wprowadzony w wiadomości e-mail jest zgodny ze standardową strukturą adresu e-mail (w przeszłości do walidacji potrzebny był kod JavaScript). Na przykład przeglądarki Opera i Chrome wyświetlają ostrzeżenie, jeśli dane wejściowe nie są zgodne z oczekiwany formatem. Chociaż e-mail, `search`, `tel` i adres `URL` są dobrze obsługiwane przez aktualne przeglądarki, mogą występować niespójności w sposobie ich obsługi. Starsze przeglądarki, takie jak Opera Mini i wszelkie wersje Internet Explorera starsze niż 11, w ogóle ich nie rozpoznają, ale zamiast tego wyświetlają domyślne ogólne wprowadzanie tekstu, co działa idealnie.



```
<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"> </label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><input type="submit" value="Send"></li>
            </ul>
        </form>
    </div>
</body>
</html>
```

- Favorite color:
- Password:
This is a default comment
- Comment:
- Email:
- Telephone:
- URL site:
- Search:
-

Element `datalist` umożliwia autorowi udostępnienie rozwijanego menu sugerowanych wartości dla dowolnego typu wprowadzania tekstu. Daje użytkownikowi kilka skrótów do wyboru, ale jeśli żaden nie zostanie wybrany, użytkownik nadal może wpisać własny tekst. W elemencie `datalist` sugerowane wartości są oznaczone jako elementy `option`. Użycie atrybutu `list` w elemencie wejściowym, wiąże go z identyfikatorem odpowiedniej listy danych.

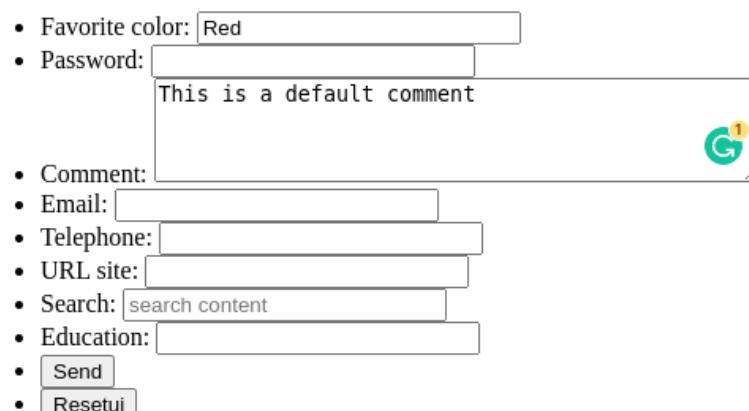
```
<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50"></label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password"></label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"> </label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><label>Education: <input type="text" list="edulevel" name="education"></label></li>
                <datalist id="edulevel">
                    <option value="High school">
                    <option value="Bachelors degree">
                    <option value="Masters degree">
                </datalist>
                <li><input type="submit" value="Send"></li>
            </ul>
        </form>
    </div>
</body>
</html>
```

- Favorite color:
- Password:
This is a default comment
- Comment:
- Email:
- Telephone:
- URL site:
- Search:
- Education:
-

W chwili pisania tego tekstu obsługa `datalist` przez przeglądarkę pozostaje nierówna. Chrome i Opera obsługują to, ale jest błąd, który uniemożliwia przewijanie list danych (tj. nieużyteczne), jeśli lista jest zbyt długa, więc najlepiej jest używać jej do krótkich list opcji. Edge ma wadliwą implementację, a Safari i iOS w ogóle ich nie obsługują. Dobrą wiadomością jest to, że jeśli nie jest obsługiwana, przeglądarki prezentują proste wprowadzanie tekstu, co jest całkowicie akceptowalnym rozwiązaniem awaryjnym. Możemy również użyć wypełniacza JavaScript, aby utworzyć funkcjonalność `datalist`.

Istnieje kilka rodzajów przycisków, które można dodawać do formularzy internetowych. Najbardziej podstawowym jest przycisk przesyłania (`submit`). Po kliknięciu przycisku `submit` natychmiast wysyła zebrane dane formularza na serwer w celu przetworzenia. Przycisk `reset` przywraca kontrolki formularza do stanu, w którym znajdowały się w momencie początkowego załadowania formularza. Innymi słowy, zresetowanie formularza nie powoduje po prostu wyczyszczenia wszystkich pól. Przyciski `submit` i `reset` są dodawane za pośrednictwem elementu wejściowego. Jak wspomniano wcześniej, ponieważ te przyciski mają określone funkcje, które nie obejmują wprowadzania danych, są jedynymi elementami sterującymi formularza, które nie wymagają atrybutu `name`, chociaż można go dodać, jeśli jest to potrzebne. Przyciski `submit` i `reset` są proste w użyciu. Wystarczy umieścić je w odpowiednim miejscu w formularzu, który w większości przypadków znajduje się na samym końcu. Możemy zmienić tekst na przycisku, używając atrybutu `value`.

```
<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
</label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password">
</label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"> </label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><label>Education: <input type="text" list="edulevel" name="education"></label></li>
                <datalist id="edulevel">
                    <option value="High school">
                    <option value="Bachelors degree">
                    <option value="Masters degree">
                </datalist>
                <li><input type="submit" value="Send"></li>
                <li><input type="reset"></li>
            </ul>
        </form>
    </div>
</body>
</html>
```



Przycisk `reset` nie jest używany w formularzach tak często, jak kiedyś. Dzieje się tak, ponieważ we współczesnym tworzeniu formularzy używamy `JavaScript` do sprawdzania poprawności danych wejściowych formularza po drodze, aby użytkownicy otrzymywali informacje zwrotne w miarę postępu. Dzięki przemyślanemu projektowi i pomocy mniej użytkowników powinno dotrzeć do końca formularza i musi zresetować całość.

Zarówno pole wyboru (`checkbox`), jak i przyciski radiowe (`radio`) ułatwiają odwiedzającym wybór spośród wielu dostępnych opcji. Są one podobne pod tym względem, że działają jak małe włączniki/wyłączniki, które mogą być przełączane przez użytkownika i dodawane za pomocą elementu `input`. Pełnią jednak różne funkcje. Kontrolka formularza składająca się ze zbioru przycisków opcji jest odpowiednia, gdy dozwolona jest tylko jedna opcja z grupy – innymi słowy, gdy wybory wzajemnie się wykluczają (takie jak „Tak lub Nie” lub „Odbiór lub Dostawa”). Gdy jeden przycisk opcji jest „włączony”, wszystkie inne muszą być „wyłączone”, podobnie jak przyciski używane na starych radiach: naciśnij jeden przycisk, a reszta wyskoczy. Jednak gdy pola wyboru są zgrupowane razem, można wybrać dowolną liczbę lub tylko kilka z grupy. To czyni je właściwym wyborem dla list, w których więcej niż jeden wybór jest prawidłowy. Przyciski `radio` są dodawane do formularza za pośrednictwem elementu `input` z atrybutem `type` ustawionym na `radio`.

```
<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"> </label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><label>Education: <input type="text" list="edulevel" name="education"></label></li>
                <datalist id="edulevel">
                    <option value="High school">
                    <option value="Bachelors degree">
                    <option value="Masters degree">
                </datalist>
                <li>Education (radio)</li>
                <ol>
                    <li><label>High school <input type="radio" name="radio" value="High" checked></label></li>
                    <li><label>Bachelors degree <input type="radio" name="radio" value="Bachelors"></label></li>
                    <li><label>Masters degree <input type="radio" name="radio" value="Master"></label></li>
                </ol>
                <li><input type="submit" value="Send"></li>
                <li><input type="reset"></li>
            </ul>
        </form>
    </div>
</body>
</html>
```

- Favorite color:
- Password:
This is a default comment
- Comment:
- Email:
- Telephone:
- URL site:
- Search:
- Education:
- Education (radio)
 - 1. High school
 - 2. Bachelors degree
 - 3. Masters degree
-
-

Atrybut `name` jest wymagany i odgrywa ważną rolę w wiązaniu wielu wejść `radio` w zestawie. Gdy kilka danych wejściowych przycisków opcji ma taką samą wartość nazwy (`radio` w przykładzie), tworzą one grupę wzajemnie wykluczających się opcji. W tym przykładzie przyciski opcji służą jako interfejs umożliwiający użytkownikom wprowadzanie swojej edukacji. Osoba nie może należeć do więcej niż jednej grupy edukacji, więc radio button to właściwy wybór. Wszystkie elementy wejściowe mają tę samą nazwę zmiennej (`radio`), ale ich wartości są różne. Ponieważ są to przyciski radiowe, w danym momencie można sprawdzić tylko jeden przycisk, dlatego tylko jedna wartość zostanie wysłana do serwera w celu przetworzenia podczas przesyłania formularza. Możemy zdecydować, który przycisk jest sprawdzany podczas ładowania formularza, dodając atrybut `checked` do elementu wejściowego.

Pola wyboru (`checkbox`) są dodawane za pośrednictwem elementu `input`, którego typ jest ustawiony na `checkbox`. Podobnie jak w przypadku przycisków radiowych, tworzymy grupy pól wyboru, przypisując im tę samą wartość nazwy. Różnica, jak już zauważyliśmy, polega na tym, że jednocześnie można zaznaczyć więcej niż jedno pole wyboru. Wartość każdego zaznaczonego przycisku zostanie wysłana na serwer po przesłaniu formularza.

```

<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50"></label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password"></label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"></label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><label>Education: <input type="text" list="edulevel" name="education"></label></li>
                <datalist id="edulevel">
                    <option value="High school">
                    <option value="Bachelors degree">
                    <option value="Masters degree">
                </datalist>
                <li>Education (radio)</li>
                <ol>
                    <li><label>High school <input type="radio" name="radio" value="High" checked></label></li>
                    <li><label>Bachelors degree <input type="radio" name="radio" value="Bachelors"></label></li>
                    <li><label>Masters degree <input type="radio" name="radio" value="Master"></label></li>
                </ol>
                <li>Languages (checkbox)</li>
                <ul>
                    <li><label>English: <input type="checkbox" name="languages" value="english"></label></li>
                    <li><label>Polish: <input type="checkbox" name="languages" value="polish"></label></li>
                    <li><label>German: <input type="checkbox" name="languages" value="german"></label></li>
                </ul>
                <li><input type="submit" value="Send"></li>
                <li><input type="reset"></li>
            </ul>
        </form>
    </div>
</body>
</html>

```

- Favorite color:
- Password:
- Comment:

This is a default comment
- Email:
- Telephone:
- URL site:
- Search:
- Education:
- Education (radio)
 - 1. High school
 - 2. Bachelors degree
 - 3. Masters degree
- Languages (checkbox)
 - English:
 - Polish:
 - German:
-
-

Innym sposobem udostępnienia listy wyborów jest umieszczenie ich w **rozwijanym** lub **przewijanym menu**. Menu wydają się być bardziej zwarte niż grupy przycisków i pól wyboru. Do formularza z wybranym elementem dodaje się zarówno menu rozwijane, jak i menu przewijane. To, czy menu jest rozwijane, czy przewijane, jest wynikiem tego, jak określmy jego rozmiar i czy zezwolimy na wybór więcej niż jednej opcji. Przyjrzyjmy się obu rodzajom menu.

Element `select` jest domyślnie wyświetlany jako menu rozwijane (nazywane również menu rozwijanym), gdy nie określono rozmiaru lub jeśli atrybut rozmiaru jest ustawiony na 1. W menu rozwijanych można wybrać tylko jeden element.

```
<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"> </label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><label>Education: <input type="text" list="edulevel" name="education"></label></li>
                <datalist id="edulevel">
                    <option value="High school">
                    <option value="Bachelors degree">
                    <option value="Masters degree">
                </datalist>
                <li>Education (radio)</li>
                <ol>
                    <li><label>High school <input type="radio" name="radio" value="High" checked></label></li>
                    <li><label>Bachelors degree <input type="radio" name="radio" value="Bachelors"></label></li>
                    <li><label>Masters degree <input type="radio" name="radio" value="Master"></label></li>
                </ol>
                <li>Languages (checkbox)</li>
                <ul>
                    <li><label>English: <input type="checkbox" name="languages" value="english"></label></li>
                    <li><label>Polish: <input type="checkbox" name="languages" value="polish"></label></li>
                    <li><label>German: <input type="checkbox" name="languages" value="german"></label></li>
                </ul>
                <li><label>Age: <select name="age">
                    <option>0-18</option>
                    <option>19-25</option>
                    <option>26-34</option>
                    <option>35+</option>
                </select></label></li>

                <li><input type="submit" value="Send"></li>
                <li><input type="reset"></li>
            </ul>
        </form>
    </div>
</body>
</html>
```

- Favorite color:
- Password:
This is a default comment
- Comment:
- Email:
- Telephone:
- URL site:
- Search:
- Education:
- Education (radio)
 - 1. High school
 - 2. Bachelors degree
 - 3. Masters degree
- Languages (checkbox)
 - English:
 - Polish:
 - German:
- Age:
-
-

Element `select` jest tylko pojemnikiem na wiele elementów opcji. Treść wybranego elementu opcji jest przekazywana do aplikacji internetowej po przesłaniu formularza. Jeśli z jakiegoś powodu chcemy wysłać inną wartość niż ta, która pojawia się w menu, należy użyć atrybutu `value`, aby podać wartość nadzczną. Aby menu wyświetlało się jako **przewijana lista**, po prostu należy określić liczbę wierszy, które mają być widoczne, za pomocą atrybutu `size`.

```

<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"> </label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><label>Education: <input type="text" list="edulevel" name="education"></label></li>
                <datalist id="edulevel">
                    <option value="High school">
                    <option value="Bachelors degree">
                    <option value="Masters degree">
                </datalist>
                <li>Education (radio)</li>
                <ol>
                    <li><label>High school <input type="radio" name="radio" value="High" checked></label></li>
                    <li><label>Bachelors degree <input type="radio" name="radio" value="Bachelors"></label></li>
                    <li><label>Masters degree <input type="radio" name="radio" value="Master"></label></li>
                </ol>
                <li>Languages (checkbox)</li>
                <ul>
                    <li><label>English: <input type="checkbox" name="languages" value="english"></label></li>
                    <li><label>Polish: <input type="checkbox" name="languages" value="polish"></label></li>
                    <li><label>German: <input type="checkbox" name="languages" value="german"></label></li>
                </ul>
                <li><label>Age: <select name="age" size="6" multiple>
                    <option>0-18</option>
                    <option>19-25</option>
                    <option>26-34</option>
                    <option>35+</option>
                </select></label></li>

                <li><input type="submit" value="Send"></li>
                <li><input type="reset"></li>
            </ul>
        </form>
    </div>
</body>
</html>

```

- Favorite color:
- Password:
This is a default comment
- Comment:
- Email:
- Telephone:
- URL site:
- Search:
- Education:
- Education (radio)
 - 1. High school
 - 2. Bachelors degree
 - 3. Masters degree
- Languages (checkbox)
 - English:
 - Polish:
 - German:
- Age:
- Send
- Resetuj

Możemy zauważyc kilka zminimalizowanych atrybutów, które są tam schowane. Atrybut `multiple` umożliwia użytkownikom dokonanie więcej niż jednego wyboru z przewijanej listy. Zauważmy, że menu rozwijane nie pozwalają na wielokrotny wybór; gdy przeglądarka wykryje wiele atrybutów, domyślnie automatycznie wyświetla małe menu przewijania. Użycie atrybutu `selected` w elemencie opcji, ustawia go jako wartość domyślną dla kontrolki menu. Wybrane opcje są podświetlane podczas ładowania formularza. Atrybut `selected` może być również używany w menu rozwijanych.

Możemy użyć elementu `optgroup` do tworzenia koncepcyjnych grup opcji. Wymagany atrybut `label` zawiera nagłówek grupy.

```

<!--form_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 2</title>
</head>
<body>
    <div id="main">
        <form action="" method="GET">
            <ul>
                <li><label>Favorite color: <input type="text" name="favoriteColor" value="Red" maxlength="50">
                </label></li>
                <li><label>Password: <input type="password" name="password" maxlength="12" id="form_password">
                </label></li>
                <li><label>Comment: <textarea name="comment" rows="4" cols="45" placeholder="Leave comment here">This is a default comment</textarea></label></li>
                <li><label>Email: <input type="email" name="email" id="form_email"></label></li>
                <li><label>Telephone: <input type="tel" name="tel"></label></li>
                <li><label>URL site: <input type="URL" name="url"> </label></li>
                <li><label>Search: <input type="search" name="search" placeholder="search content"></label></li>
                <li><label>Education: <input type="text" list="edulevel" name="education"></label></li>
                <datalist id="edulevel">
                    <option value="High school">
                    <option value="Bachelors degree">
                    <option value="Masters degree">
                </datalist>
                <li>Education (radio)</li>
                <ol>
                    <li><label>High school <input type="radio" name="radio" value="High" checked></label></li>
                    <li><label>Bachelors degree <input type="radio" name="radio" value="Bachelors"></label></li>
                    <li><label>Masters degree <input type="radio" name="radio" value="Master"></label></li>
                </ol>
                <li>Languages (checkbox)</li>
                <ul>
                    <li><label>English: <input type="checkbox" name="languages" value="english"></label></li>
                    <li><label>Polish: <input type="checkbox" name="languages" value="polish"></label></li>
                    <li><label>German: <input type="checkbox" name="languages" value="german"></label></li>
                </ul>
                <li><label>Age: <select name="age" size="6" multiple>
                    <option>0-18</option>
                    <option>19-25</option>
                    <option>26-34</option>
                    <option>35+</option>
                </select></label></li>
                <li><label>Ice creams: <select name="ice" size="7" multiple>
                    <optgroup label="traditional">
                        <option selected>Vanilla</option>
                        <option>Chocolate</option>
                    </optgroup>
                    <optgroup label="fancy">
                        <option>Super praline</option>
                        <option>Nut suprise</option>
                        <option>Candy corn</option>
                    </optgroup>
                </select> </label></li>
                <li><input type="submit" value="Send"></li>
                <li><input type="reset"></li>
            </ul>
        </form>
    </div>
</body>
</html>

```

- Favorite color:
- Password:
This is a default comment
- Comment:
- Email:
- Telephone:
- URL site:
- Search:
- Education:
- Education (radio)
 - 1. High school
 - 2. Bachelors degree
 - 3. Masters degree
- Languages (checkbox)
 - English:
 - Polish:
 - German:
- Age:

0-18
19-25
26-34
35+
- Ice creams:

traditional
Vanilla
Chocolate
fancy
Super praline
Nut surprise
Candy corn
- Send
- Resetuj

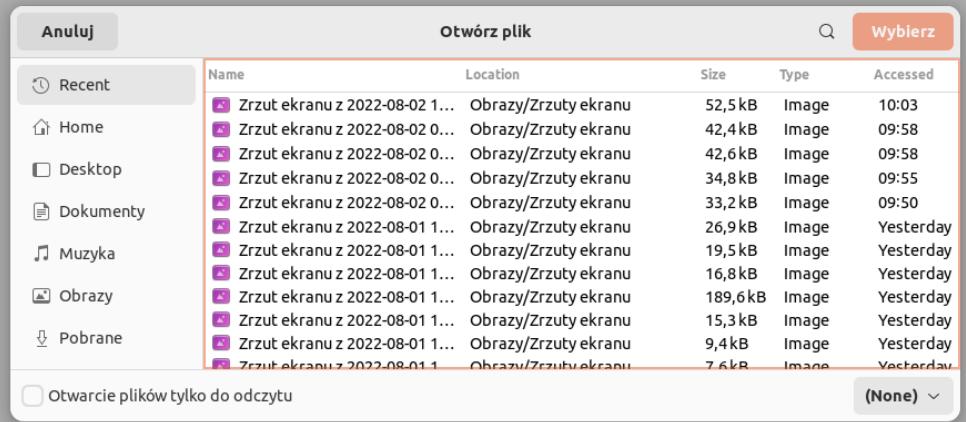
Step 11

Formularze internetowe mogą gromadzić więcej niż tylko dane. Mogą być również używane do przesyłania dokumentów zewnętrznych z dysku twardego użytkownika. Na przykład firma poligraficzna może użyć formularza internetowego, aby przesyłać grafikę do zamówienia wizytówki. Magazyn mógłby wykorzystać formularz do zbierania zdjęć cyfrowych na konkurs fotograficzny. Kontrolka wyboru pliku (`file`) umożliwia użytkownikom wybranie dokumentu z dysku twardego, który ma zostać przesłany z danymi formularza. Dodajemy go do formularza za pomocą naszego starego przyjaciela, elementu `input`, którego typ jest ustawiony na `file`.

```
<!--form_example3.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Form example 3</title>
</head>
<body>
  <div id="main">
    <form action="client.php" method="POST" enctype="multipart/form-data">
      <label>Send a photo to be used as your online icon <em>(optional)</em><br>
        <input type="file" name="photo"></label>
    </form>
  </div>
</body>
</html>
```

end a photo to be used as your online icon (optional)

Wybierz plik Nie wybrano pliku



Widżet przesyłania plików różni się nieznacznie w zależności od przeglądarki i systemu operacyjnego, ale zazwyczaj jest to przycisk, który umożliwia dostęp do systemu organizacji plików na komputerze. Należy zauważyć, że gdy formularz zawiera element wejściowy wyboru pliku, należy określić typ kodowania (`enctype`) jako `multipart/form-data` w elemencie formularza i użyć metody `POST`. Typ pliku wejściowego ma kilka atrybutów. Atrybut `accept` informuje przeglądarkę o tym, jakie typy plików mogą być akceptowane (audio, video, obraz lub inny format identyfikowany przez typ nośnika). Dodanie atrybutu `multiple` umożliwia wybranie wielu plików do przesłania. Atrybut `require`, mówi, że wymaga wybrania pliku.

Może się zdarzyć, że będziemy musieli wysłać do aplikacji przetwarzającej formularze informacje, które nie pochodzą od użytkownika. W takich przypadkach można użyć ukrytej kontrolki formularza (`hidden`), która wysyła dane po przesłaniu formularza, ale nie jest widoczna, gdy formularz jest wyświetlany w przeglądarce. Ukryte kontrolki (`hidden`) są dodawane za pośrednictwem elementu `input` z typem ustawionym na `hidden`. Jego jedynym celem jest przekazanie pary nazwa/wartość do serwera podczas przesyłania formularza.

```
<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>
```

SEND RESET

Jest to rodzaj informacji, które otrzymujemy od programisty aplikacji, administratora systemu lub innej osoby, która pomaga w przetwarzaniu formularzy.

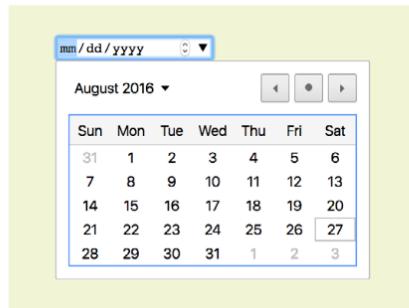
Jeśli kiedykolwiek rezerwowałyśmy hotel lub lot online, bez wątpienia używaliśmy małego widżetu kalendarza do wyboru daty. Są szanse, że mały kalendarz został stworzony w `JavaScript`. `HTML5` wprowadził sześć nowych typów danych wejściowych, które sprawiają, że widżety wyboru daty i godziny są częścią standardowych wbudowanych funkcji wyświetlania przeglądarki, tak jak dziś

mogą wyświetlać pola wyboru, wyskakujące menu i inne widżety. W chwili pisania tego tekstu selektory daty i godziny są zaimplementowane tylko w kilku przeglądarkach (Chrome , Microsoft Edge , Opera , Vivaldi i Android), ale w nieobsługujących przeglądarkach typy wprowadzania daty i godziny są wyświetlane jako do wpisania tekst.

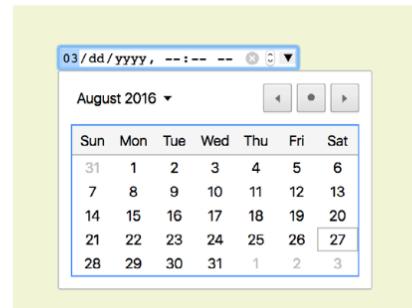
`input type="time"`



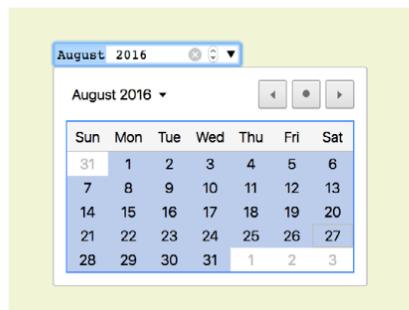
`input type="date"`



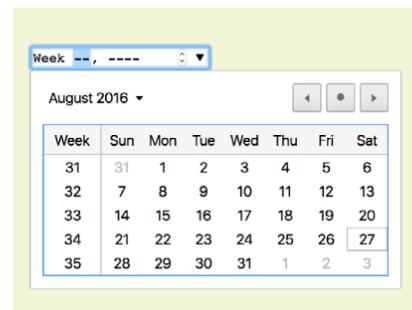
`input type="datetime-local"`



`input type="month"`



`input type="week"`



Typ `date` tworzy kontrolkę wprowadzania daty, taką jak kalendarz podręczny, do określania daty (rok, miesiąc, dzień). Wartość początkową należy podać w formacie daty ISO (`RRRR-MM-DD`).

```
<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>
```

Date:

Typ `time` tworzy kontrolkę wprowadzania czasu w celu określenia czasu (godzina, minuta, sekundy, części ułamkowe) bez wskazanej strefy czasowej. Wartość jest podana jako `gg:mm:ss` .

```

<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>

```

Date:

Time:

Typ `datetime-local` tworzy połączoną kontrolkę wprowadzania daty/godziny bez informacji o strefie czasowej w formacie (`RRRR-MM-DDThh:mm:ss`).

```

<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <label>Datetime: <input type="datetime-local" name="datetime" value="2022-08-02T10:19:00"></label>
<br>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>

```

Date:

Time:

Datetime:

Typ `month` tworzy kontrolkę wprowadzania daty, która określa określony miesiąc w roku (`RRRR-MM`), natomiast typ `week` tworzy kontrolkę wprowadzania daty do określonego tygodnia w roku przy użyciu formatu numerowania tygodni ISO (`YYYY-W#`).

```

<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <label>Datetime: <input type="datetime-local" name="datetime" value="2022-08-02T10:19:00"></label>
<br>
            <label>Month: <input type="month" name="month" value="2022-08"></label><br>
            <label>Week: <input type="week" name="week" value="2022-W32"></label><br>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>

```

Date:

Time:

Datetime:

Month:

Week:

Typy wejściowe liczby (`number`) i zakresu (`range`) zbierają dane liczbowe. W przypadku wprowadzania liczb przeglądarka może dostarczyć widżet pokrętła ze strzałkami w górę i w dół w celu wybrania określonej wartości liczbowej (wprowadzanie tekstu może być wyświetlane w programach użytkownika, które nie obsługują typu wprowadzania). Wprowadzany zakres jest zwykle wyświetlany jako suwak, który pozwala użytkownikowi wybrać wartość z określonego zakresu.

```

<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <label>Datetime: <input type="datetime-local" name="datetime" value="2022-08-02T10:19:00"></label>
<br>
            <label>Month: <input type="month" name="month" value="2022-08"></label><br>
            <label>Week: <input type="week" name="week" value="2022-W32"></label><br><hr>
            <label>Number of guests <input type="number" name="guests" min="1" max="6"></label><br>
            <label>Satisfaction (0 to 10) <input type="range" name="Satisfaction" min="0" max="10" step="1">
</label><br>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>

```

Date:

Time:

Datetime:

Month:

Week:

Number of guests

Satisfaction (0 to 10)

Zarówno `number`, jak i typ danych wejściowych `range` akceptują atrybuty `min` i `max` w celu określenia minimalnej i maksymalnej wartości dozwolonej dla danych wejściowych (ponownie przeglądarka może sprawdzić, czy dane wejściowe użytkownika są zgodne z ograniczeniem). Zarówno `min`, jak i `max` są opcjonalne, można też ustawić jedno bez drugiego. Dopuszczalne są wartości ujemne. Po wybraniu elementu wartość można zwiększać lub zmniejszać za pomocą klawiszy numerycznych na klawiaturze komputera, a także przesuwać myszką lub palcem. Atrybut `step` umożliwia programistom określenie akceptowalnych przyrostów dla danych liczbowych. Wartość domyślna to 1. Wartość „`.5`” zezwala na wartości 1, 1,5, 2, 2,5 itd.; wartość `100` pozwoliłaby na 100, 200, 300 i tak dalej. Możemy również ustawić atrybut `step` na `any`, aby jawnie zaakceptować dowolny przyrost wartości. Te dwa elementy pozwalają tylko na obliczone wartości kroków, a nie na określoną listę dozwolonych wartości (takich jak 1, 2, 3, 5, 8, 13, 21). Jeśli potrzebujemy niestandardowych wartości, musimy użyć `JavaScript` do zaprogramowania tego zachowania. Ponieważ są to nowsze elementy, obsługa przeglądarek jest niespójna. Niektóre widżety interfejsu użytkownika zawierają strzałki w góre i w dół do zwiększania lub zmniejszania kwoty, ale wiele z nich tego nie robi. Przeglądarki mobilne (iOS Safari, Android, Chrome na Androida) obecnie nie obsługują `min`, `max` i `step`. Przeglądarki, które nie obsługują tych nowych typów wprowadzania, wyświetlają zamiast tego standardowe pole wprowadzania tekstu, co jest dobrym rozwiązaniem.

Intencją typu sterowania kolorem (`color`) jest utworzenie wyskakującego próbnika kolorów do wizualnego wybierania wartości koloru podobnej do tej używanej w systemach operacyjnych lub programach do edycji obrazów. Wartości są podawane w szesnastkowych wartościach `RGB` (`#RRGGBB`).

```
<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <label>Datetime: <input type="datetime-local" name="datetime" value="2022-08-02T10:19:00"></label>
            <br>
            <label>Month: <input type="month" name="month" value="2022-08"></label><br>
            <label>Week: <input type="week" name="week" value="2022-W32"></label><br><hr>
            <label>Number of guests <input type="number" name="guests" min="1" max="6"></label><br>
            <label>Satisfaction (0 to 10) <input type="range" name="Satisfaction" min="0" max="10" step="1">
            </label><br><hr>
            <label>Choose color: <input type="color" name="color"></label><br><hr>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>
```

Date:

Time:

Datetime:

Month:

Week:

Number of guests

Satisfaction (0 to 10)

Choose color:

Dla kompletności przyjrzyjmy się pozostałym elementom formularza. Zostały one dodane w **HTML5** i, w chwili pisania tego tekstu, nadal mają nierówną obsługę przeglądarek. I tak są nieco ezoteryczne, więc możemy chwilę poczekać, aby dodać je do swojego zestawu narzędzi **HTML**. Omówiliśmy już element **datalist** do dostarczania sugerowanych wartości dla danych wejściowych tekstu. **HTML5** wprowadził również następujące elementy.

Element postępu (**progress**) zapewnia użytkownikom informacje zwrotne na temat stanu trwającego procesu, takiego jak pobieranie pliku. Może wskazywać określony procent ukończenia (**determinate**), jak pasek postępu (wówczas należy ustawić atrybut **value**), lub po prostu wskazywać stan „oczekiwania” (**indeterminate**), jak spinner. Element **progress** wymaga do działania skryptów.

```
<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <label>Datetime: <input type="datetime-local" name="datetime" value="2022-08-02T10:19:00"></label>
            <br>
            <label>Month: <input type="month" name="month" value="2022-08"></label><br>
            <label>Week: <input type="week" name="week" value="2022-W32"></label><br><hr>
            <label>Number of guests <input type="number" name="guests" min="1" max="6"></label><br>
            <label>Satisfaction (0 to 10) <input type="range" name="Satisfaction" min="0" max="10" step="1">
            </label><br><hr>
            <label>Choose color: <input type="color" name="color"></label><br><hr>
            <label>Percent download: <progress max="100" id="fave">0</progress> </label><br>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>
```

Date:

Time:

Datetime:

Month:

Week:

Number of guests

Satisfaction (0 to 10)

Choose color:

Percent download:

Miernik (meter) reprezentuje pomiar w znany zakresie wartości. Ma kilka atrybutów: `min` i `max` wskazują najwyższe i najniższe wartości dla zakresu (domyślnie 0 i 100); `low` i `high` mogą być wykorzystywane do wyzwalania ostrzeżeń na niepożądanych poziomach; a `optimum` określa preferowaną wartość.

```
<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <label>Datetime: <input type="datetime-local" name="datetime" value="2022-08-02T10:19:00"></label>
<br>
            <label>Month: <input type="month" name="month" value="2022-08"></label><br>
            <label>Week: <input type="week" name="week" value="2022-W32"></label><br><hr>
            <label>Number of guests <input type="number" name="guests" min="1" max="6"></label><br>
            <label>Satisfaction (0 to 10) <input type="range" name="Satisfaction" min="0" max="10" step="1">
</label><br><hr>
            <label>Choose color: <input type="color" name="color"></label><br><hr>
            <label>Percent download: <progress value="20" max="100" id="fave">20</progress> </label><br>
            <label>Meter example: <meter min="0" max="100" id="meter" optimum="90" value="60">20%</meter>
</label><br>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form>
    </div>
</body>
</html>
```

Date:

Time:

Datetime:

Month:

Week:

Number of guests

Satisfaction (0 to 10)

Choose color:

Percent download:

Meter example:

Mówiąc najprościej, element `output` wskazuje wynik obliczenia przez skrypt lub program. Ten przykład, zaczerpnięty ze specyfikacji HTML5.2, używa elementu `output` i JavaScript do wyświetlenia sumy liczb wprowadzonych do wejść `a` i `b`.

```
<!--form_example4.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form example 4</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label>Date: <input type="date" name="date" value="2022-08-02"></label><br>
            <label>Time: <input type="time" name="time" value="10:19:00"></label><br>
            <label>Datetime: <input type="datetime-local" name="datetime" value="2022-08-02T10:19:00"></label>
<br>
            <label>Month: <input type="month" name="month" value="2022-08"></label><br>
            <label>Week: <input type="week" name="week" value="2022-W32"></label><br><hr>
            <label>Number of guests <input type="number" name="guests" min="1" max="6"></label><br>
            <label>Satisfaction (0 to 10) <input type="range" name="Satisfaction" min="0" max="10" step="1">
</label><br><hr>
            <label>Choose color: <input type="color" name="color"></label><br><hr>
            <label>Percent download: <progress value="20" max="100" id="fave">20</progress> </label><br>
            <label>Meter example: <meter min="0" max="100" id="meter" optimum="90" value="60">20%</meter>
</label><br>
            <input type="hidden" name="success" value="http://example.com/thankYou.html">
            <input type="submit" value="SEND">
            <input type="reset" value="RESET">
        </form><hr>
        <p>Output example:</p>
        <form onsubmit="return false" oninput="o.value = a.valueAsNumber + b.valueAsNumber">
            <label>A: <input name="a" type="number" step="any"></label>
            <label><input name="b" type="number" step="any"></label>
            <output name="o" for="a b"></output>
        </form>
    </div>
</body>
</html>
```

The screenshot shows a web form with the following fields:

- Date: 02.08.2022
- Time: 10:19
- Datetime: 02.08.2022, 10:19
- Month: Sierpień 2022
- Week: Tydzień 32, 2022
- Number of guests: (input field)
- Satisfaction (0 to 10): (range slider, value 60)
- Choose color: (color picker, black)
- Percent download: (progress bar, value 20%)
- Meter example: (meter, value 60%)
- Output example: A: 4 + 5 = 9

Step 12

Ważne jest, aby zastanowić się, w jaki sposób użytkownicy bez przeglądarki wizualnych będą mogli zrozumieć i poruszać się po formularzach internetowych. Elementy formularza `label`, `fieldset` i `legend` poprawiają dostępność, zapewniając czytelność połączeń semantycznych między składnikami formularza. Wynikowy znacznik jest nie tylko bogatszy semantycznie, ale także

dostępnych jest więcej elementów, które działają jako „haczyki” dla reguł arkusza stylów. Chociaż możemy zobaczyć etykietę „Adres” tuż obok pola tekstu do wprowadzania adresu w przeglądarce wizualnej, w źródle etykieta i dane wejściowe mogą być rozdzielone. Element `label` wiąże tekst opisowy z odpowiednim polem formularza. Zapewnia to ważny kontekst dla użytkowników korzystających z przeglądarek opartych na mowie. Kolejną zaletą używania etykiet jest to, że użytkownicy mogą kliknąć lub dotknąć w dowolnym miejscu, aby wybrać lub ustawić kontrolkę formularza. **Każdy element etykiety jest powiązany z dokładnie jedną kontrolką formularza.** Można go wykorzystać na dwa sposoby. Jedna metoda, zwana **niewawnym powiązaniem**, zagnieźduje kontrolkę i jej opis w elemencie `label`. W poniższym przykładzie etykiety są przypisane do poszczególnych pól wyboru i powiązanych z nimi opisów tekstowych. (Nawiasem mówiąc, jest to sposób oznaczania przycisków radiowych i pól wyboru. Nie można przypisać etykiety do całej grupy.)

```
<li>Languages (checkbox)</li>
    <ul>
        <li><label>English: <input type="checkbox" name="languages" value="english"></label></li>
        <li><label>Polish: <input type="checkbox" name="languages" value="polish"></label></li>
        <li><label>German: <input type="checkbox" name="languages" value="german"></label></li>
    </ul>
```

Druga metoda, zwana **jawnym powiązaniem**, dopasowuje etykietę do referencji `id` kontrolki. Atrybut `for` mówi, do której kontrolki jest przeznaczona etykieta. To podejście jest przydatne, gdy formant nie znajduje się bezpośrednio obok tekstu opisowego w źródle. Oferuje również potencjalną korzyść polegającą na utrzymywaniu etykiety i kontrolki jako dwóch odrębnych elementów, które mogą być przydatne podczas wyrównywania ich z arkuszami stylów.

```
<!--label_Example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Label example</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <label for="form-login-username">Login account</label>
            <input type="text" name="login" id="form-login-username"><br>
            <label for="form-login-password">Password</label>
            <input type="password" name="password" id="form-login-password"><br>
            <input type="reset" value="RESET">
            <input type="submit" value="SEND">
        </form>
    </div>
</body>
</html>
```

The screenshot shows a simple login form. It consists of two text input fields: one for the login account and one for the password. Each input field is preceded by a label: "Login account" and "Password" respectively. Below the input fields are two buttons: "RESET" and "SEND".

Element zestawu pól (`fieldset`) wskazuje logiczną grupę kontrolek formularza. Zestaw pól może również zawierać element `legend`, który zapewnia podpis dla zamkniętych pól.

```

<!--fieldset_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Fieldset example</title>
</head>
<body>
    <div id="main">
        <form method="GET">
            <fieldset>
                <legend>Mailing List Sign-up</legend>
                <label>Add me to your mailing list <input type="radio" name="list" value="yes" checked>
</label><br>
                <label>No thanks <input type="radio" name="list" value="no"></label><br>
            </fieldset>
            <fieldset>
                <legend>Customer information</legend>
                <label for="full-name">Full name: </label>
                <input type="text" id="full-name" name="fullname"><br>
                <label for="email">Email</label>
                <input type="text" id="email" name="email"><br>
                <label for="state">State</label>
                <input type="text" id="state" name="state">
            </fieldset>
            <fieldset>
                <input type="reset" value="RESET">
                <input type="submit" value="SEND">
            </fieldset>
        </form>
    </div>
</body>
</html>

```

Mailing List Sign-up

Add me to your mailing list

No thanks

Customer information

Full name:

Email

State

Źle zaprojektowany formularz może zrujnować wrażenia użytkownika w witrynie i negatywnie wpływać na cele biznesowe. Źle zaprojektowane formularze oznaczają utratę klientów, dlatego tak ważne jest, aby zrobić to dobrze – zarówno na biurku, jak i w przypadku urządzeń z małym ekranem ze specjalnymi wymaganiami. Chcemy, aby droga do zakupu lub innego działania przebiegała jak najłatwiej. Temat dobrego projektowania formularzy internetowych to temat bogaty, który sam w sobie mógłby wypełnić książkę. W rzeczywistości istnieje taka książka: *Web Form Design* (Rosenfeld Media) - <https://rosenfeldmedia.com/books/web-form-design/> (<https://rosenfeldmedia.com/books/web-form-design/>). Kolejna książka, *Mobile First* (A Book Apart) - <https://abookapart.com/products/mobile-first> (<https://abookapart.com/products/mobile-first>), zawiera wskazówki dotyczące formatowania formularzy w kontekście mobilnym. Możemy przeglądać ponad sto artykułów o formularzach na stronie www.lukew.com/ff?tag=forms ([https://www.lukew.com/ff?tag=forms](http://www.lukew.com/ff?tag=forms)). Oto kilka porad podsumowujących prawidłowe tworzenie formularzy.

- 1. Unikaj niepotrzebnych pytań.** Pomóż użytkownikom przejść przez formularz tak łatwo, jak to możliwe, nie dodając pytań, które nie są absolutnie konieczne do wykonania zadania. Dodatkowe pytania, oprócz tego, że spowalniają pracę, mogą sprawić, że użytkownik będzie nieufnie podchodzić do naszych motywacji do zadawania pytań. Jeśli mamy inny sposób uzyskania informacji (np. rodzaj karty kredytowej można określić na podstawie pierwszych czterech cyfr rachunku), skorzystaj z alternatywnych sposobów i nie obciążaj użytkownika. Jeśli istnieją informacje, które mogą być miłe, ale nie są wymagane, zastanów się nad pytaniem później, po przesłaniu formularza i zbudowaniu relacji z użytkownikiem.

2. Rozważ wpływ umieszczenia etykiety. Pozycja etykiety względem danych wejściowych wpływa na czas potrzebny na wypełnienie formularza. Im mniejoko użytkownika musi odbijać się po stronie, tym szybsze wypełnianie formularza. Umieszczenie etykiet nad odpowiednimi polami tworzy jedno wyrównanie, co przyspiesza skanowanie i uzupełnianie, szczególnie gdy prosimy o znajome informacje (nazwisko, adres itp.). Etykiety umieszczone u góry mogą również pomieścić etykiety o różnej długości i działają najlepiej na wąskich urządzeniach o małym ekranie. Dają jednak dłuższą formę, więc jeśli problemem jest pionowa przestrzeń, możesz umieścić etykiety po lewej stronie danych wejściowych. Wyrównanie etykiet do lewej powoduje najwolniejsze wypełnianie formularzy, ale może być odpowiednie, jeśli chcesz, aby użytkownik spowolnił lub był w stanie skanować i rozważyć rodzaje wymaganych informacji.

3. Starannie wybieraj typy danych wejściowych. Jest sporo typów danych wejściowych do wyboru i czasami nie jest łatwo zdecydować, którego użyć. Na przykład lista opcji może być przedstawiona jako menu rozwijane lub kilka opcji z polami wyboru. Dokładnie zważ zalety i wady każdego typu kontroli i wykonaj testy z użytkownikami.

4. Grupuj powiązane rzeczy w formularzu. Łatwiej jest analizować wiele pól, menu i przycisków w formularzu, jeśli są one wizualnie pogrupowane według powiązanych tematów. Na przykład informacje kontaktowe użytkownika mogą być prezentowane w zwartej grupie, dzięki czemu pięć lub sześć wejść jest postrzeganych jako jedna jednostka. Zwykle wystarczy bardzo subtelna wskazówka, taka jak cienka linia pozioma i dodatkowa przestrzeń.

5. Wyjaśnij działania wysłania formularza i wszelkich błędów. Podstawową akcją na końcu formularza jest zwykle przycisk przesyłania („Kup”, „Zarejestruj się” itp.), który sygnalizuje wypełnienie formularza i gotowość do przejścia dalej. Chcemy, aby ten przycisk był wizualnie dominujący i łatwy do znalezienia (przydatne jest również wyrównanie go wzdłuż głównej osi formularza). Używając `JavaScript`, możemy wyszarzyć przycisk przesyłania jako niedziałający, dopóki wszystkie niezbędne dane nie zostaną wypełnione.

Step 13

Specyfikacja `HTML` definiuje **osadzoną zawartość** w następujący sposób: zawartość, która importuje inny zasób do dokumentu lub zawartość z innego słownika, która jest wstawiana do dokumentu. Elementy `img` i `picture` wskazują na zewnętrzny zasób obrazu za pomocą atrybutów `src` lub `srcset`, a element `svg` osadza plik obrazu zapisany w słowniku `SVG` bezpośrednio na stronie. Ale obrazy z pewnością nie są jedynymi rzeczami, które możesz umieścić na stronie internetowej. Istnieją inne typy osadzonej zawartości i ich odpowiednim znacznikom, w tym:

- Okno do przeglądania zewnętrznego źródła HTML (`iframe`)
- Uniwersalne elementy osadzone (`object` i `embed`)
- Odtwarzacze wideo i audio (`video` i `audio`)
- Skryptowy obszar rysowania, który można wykorzystać do animacji lub interaktywności podobnej do gry (`canvas`)

Element `iframe` (skrót od wbudowanej ramki) umożliwia osadzenie w dokumencie oddzielnego dokumentu `HTML` lub innego zasobu internetowego. Istnieje od wielu lat, ale ostatnio stał się jednym z najpopularniejszych sposobów udostępniania treści między witrynami. Na przykład, gdy poprosimy o kod, aby umieścić film z `YouTube` lub mapę z Map Google, udostępniają one kod oparty na `iframe` do skopiowania i wklejenia na stronie. Wiele innych witryn multimedialnych idzie w ich ślady, ponieważ pozwala im to kontrolować aspekty treści, które umieszczamy na swojej stronie. Ramki wbudowane stały się również standardowym narzędziem do osadzania treści reklam, które mogły być obsługiwane przez Flash w przeszłości. Witryny z samouczkami sieci Web mogą używać ramek wbudowanych do osadzania na stronach próbek kodu. Dodanie ramki `iframe` do strony tworzy małe okno w oknie (lub zagnieżdżony kontekst przeglądania, jak jest znany w specyfikacji), który wyświetla zasób zewnętrzny. Umieszczamy ramkę w wierszu na stronie podobnie do obrazu, określając źródło (źródło) jej zawartości. Atrybuty `width` i `height` określają wymiary ramki. Zawartość samego elementu `iframe` jest zawartością zastępczą dla przeglądarki, które nie obsługują tego elementu, chociaż praktycznie wszystkie przeglądarki obsługują w tym momencie elementy `iframe`.

```
<!--iframe_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Iframe example</title>
</head>
<body>
    <div id="main">
        <h3>Mateusz Miotk Homepage</h3>
        <iframe src="https://mmiotk.gitlab.io" width="400" height="400">Mateusz Miotk Homepage2</iframe>
    </div>
</body>
</html>
```

Mateusz Miotk Homepage

Adres

Instytut Informatyki
Zakład optymalizacji
kombinatorycznej
Uniwersytet Gdańskie
ul. Wita Stwosza 57
80-306 Gdańsk, Polska

Kontakt oraz
konsultacje

Email:

Korzystanie z ramek `iframe` wiąże się z pewnymi problemami dotyczącymi bezpieczeństwa, ponieważ mogą one zachowywać się jak otwarte okna, przez które hakerzy mogą się przemykać. Atrybut `sandbox` nakłada ograniczenia na to, co może zrobić treść w ramkach, na przykład nie zezwala na formularze, wyskakujące okienka, skrypty i tym podobne. Artykuł MDN Web Docs „From object to iframe: Other Embedding Technologies” (https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Other_embedding_technologies (https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Other_embedding_technologies)), zawiera dobry przegląd problemów związanych z bezpieczeństwem `iframe`.

Na początku przeglądarek internetowych miały bardzo ograniczone możliwości renderowania, więc polegały na wtyczkach, które pomagały im wyświetlać multimedia, z którymi nie radziły sobie natywnie. Aplety `Java`, filmy `Flash`, `RealMedia` (stary internetowy format wideo i audio) i inne multimedia wymagały wtyczek innych firm, aby mogły być odtwarzane w przeglądarce. `Heck`, nawet obrazy `JPEG` kiedyś wymagały wtyczki do wyświetlania. Aby osadzić te zasoby multimedialne na stronie, używamy elementów `object` i `embed`. Mają nieco inne zastosowania. Element `object` jest wielozadaniowy. Może być używany do umieszczania obrazu, tworzenia zagnieździonego kontekstu przeglądania (takiego jak `iframe`) lub osadzenia zasobu, który musi być obsługiwany przez wtyczkę. Element `embed` był przeznaczony wyłącznie do użytku z wtyczkami. Mówiąc szczerze, chociaż nadal jest w użyciu, `object` wyszedł z mody, a `embed` jest prawie wymarły. Media, takie jak aplety `Java` i filmy `Flash`, szybko znikają, a nowoczesne przeglądarki używają interfejsów API do wyświetlania wielu rodzajów multimediów natywnie. Ponadto przeglądarki mobilne oraz przeglądarka `Microsoft Edge` na komputery stacjonarne nie obsługują wtyczek. W swojej najbardziej minimalnej wartości element

`object` używa atrybutu `data`, aby wskazać zasób, oraz atrybutu `type`, aby zapewnić jego typ `MIME`. Wszelkie treści zawarte w tagach elementu `object` będą używane jako zastępce dla przeglądarki, które nie obsługują osadzonego typu zasobu. Oto prosty element `object`, który umieszcza obraz `SVG` na stronie i zapewnia rezerwę w postaci pliku `PNG`.

```
<!--object_embed_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Object and embed example</title>
</head>
<body>
  <div id="main">
    <object data="images/hello.svg" type="image/svg+xml">
      
    </object>
  </div>
</body>
</html>
```



Dostępne są dodatkowe atrybuty elementu `object`, które różnią się w zależności od typu umieszczanego nośnika. Format multimedialnych może również wymagać, aby obiekt zawierał pewną liczbę elementów, które ustawiają parametry specyficzne dla tego typu multimedialnych.

Do niedawna przeglądarki nie miały wbudowanych możliwości obsługi wideo lub dźwięku, więc używały wtyczek, aby wypełnić lukę. Wraz z rozwojem sieci jako platformy otwartych standardów oraz łącz szerokopasmowych umożliwiających pobieranie większej ilości danych niż wcześniej, wydawało się, że nadszedł czas, aby obsługa multimedialnych stała się częścią gotowych do użycia możliwości przeglądarek. Wprowadzono nowe elementy `video` i `audio` oraz odpowiadające im interfejsy API. Dobrą wiadomością jest to, że elementy `video` i `audio` są dobrze obsługiwane w nowoczesnych przeglądarkach, w tym `Safari`, `Chrome`, `Opera` i `Firefox` na komputery stacjonarne oraz `iOS Safari 4+`, `Android 2.3+` i `Opera Mobile`. Chociaż wszyscy ustaliли się na znacznikach i JavaScript do osadzania odtwarzaczy multimedialnych, niestety nie uzgodnili, które formaty obsługują. Wybierzmy się w krótką podróż przez krainę formatów plików multimedialnych. Jeśli chcemy dodać wideo lub audio do swojej strony, ważne jest, aby to zrozumieć. Przygotowując zawartość audio lub wideo do udostępnienia w Internecie, należy podjąć dwie decyzje dotyczące formatu. Pierwszym z nich jest sposób kodowania mediów (algorytmy używane do konwersji źródła na jedynek i zer oraz sposób ich kompresji). Metoda używana do kodowania nazywana jest **kodekiem**, co jest skrótem od „kodowania/dekodowania” lub „skompresowania/dekompresji”. Istnieje bilion kodeków (to dane szacunkowe). Niektóre prawdopodobnie brzmią znajomo, jak `MP3`; inne mogą brzmieć jak nowe, takie jak `H.264`, `Vorbis`, `Theora`, `VP8` i `AAC`. Po drugie, musimy wybrać format kontenera dla mediów. Możemy myśleć o tym jako o pliku `ZIP`, który przechowuje skompresowany nośnik i jego metadane razem w pakiecie. Zwykle format kontenera jest zgodny z więcej niż jednym typem kodeka, a cała historia jest skomplikowana. Do najpopularniejszych kodeków oraz formatów plików wideo należą:

- `MPEG-4` + kodek wideo `H.264` + kodek audio `AAC`. Ta kombinacja jest ogólnie określana jako `MPEG-4` i przyjmuje sufiks pliku `.mp4` lub `.m4v`. `H.264` to wysokiej jakości i elastyczny kodek wideo, ale jest opatentowany i musi być licencjonowany za

opłatą. Wszystkie obecne przeglądarki obsługujące wideo HTML5 mogą odtwarzać pliki MPEG-4 za pomocą kodeka H.264.

Nowszy kodek H.265 (znany również jako HEVC, High Efficiency Video Coding) jest w fazie rozwoju i zmniejsza szybkość transmisji o połowę, ale nie jest dobrze obsługiwany w chwili pisania tego tekstu.

- WebM + kodk wideo VP8 + kodk audio Vorbis. WebM to format kontenera, który ma tę zaletę, że jest open source i jest wolny od opłat licencyjnych. Używa rozszerzenia pliku .webm. Został pierwotnie zaprojektowany do pracy z kodekami VP8 i Vorbis.
- WebM + kodk wideo VP9 + kodk audio Opus. Kodek wideo VP9 z projektu WebM oferuje taką samą jakość wideo jak VP8 i H.264 przy połowie bitrate. Ponieważ jest nowszy, nie jest tak dobrze obsługiwany, ale jest świetną opcją dla przeglądarek, które ją obsługują.
- Ogg + kodk wideo Theora + kodk audio Vorbis. Zwykle nazywa się to Ogg Theora, a plik powinien mieć sufiks .ogg. Wszystkie kodeki i kontener w tej opcji są open source i nie są obciążone patentami ani ograniczeniami licencyjnymi, ale niektórzy twierdzą, że jakość jest gorsza od innych opcji. Oprócz nowych przeglądarek jest obsługiwany w niektórych starszych wersjach Chrome, Firefox i Android, które nie obsługują WebM ani MP4, więc włączenie go zapewnia odtwarzanie dla większej liczby użytkowników.

Oczywiście problem polega na tym, że twórcy przeglądarek nie uzgodnili jednego formatu do obsługi. Niektórzy idą z otwartymi, darmowymi opcjami, takimi jak Ogg Theora lub WebM. Inni trzymają się H.264 pomimo wymagań dotyczących opłat licencyjnych. Oznacza to, że my, twórcy stron internetowych, musimy tworzyć wiele wersji filmów, aby zapewnić obsługę we wszystkich przeglądarkach. Poniżej znajduje się tabela wykazująca obsługę formatów wideo.

Format	Type	IE	MS Edge	Chrome	Firefox	Safari	Opera	Android	iOS Safari
MP4 (H.264)	video/mp4 mp4 m4v	9.0+	12+	4+	Yes*	3.2+	25+	4.4+	3.2+
WebM (VP8)	video/webm webm webmv	–	–	6+	4.0+	–	15+	2.3+	–
WebM (VP9)	video/webm webm webmv	–	14+	29+	28+	–	16+	4.4+	–
Ogg Theora	video/ogg ogv	–	–	3.0+	3.5+	–	13+	2.3+	–

Krajobraz wygląda podobnie w przypadku formatów audio: kilka do wyboru, ale żaden format nie jest obsługiwany przez wszystkie przeglądarki.

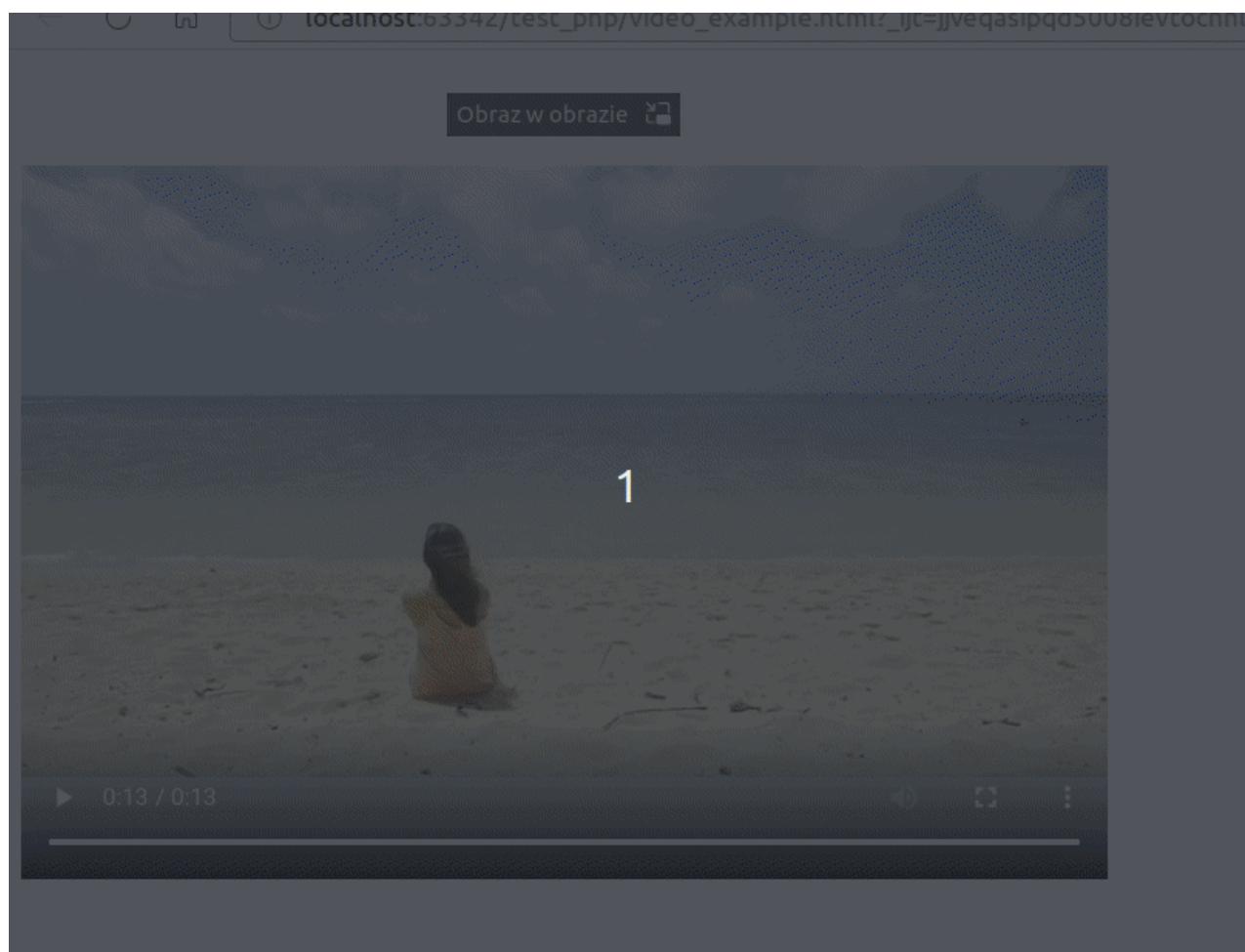
- MP3. Format MP3 (skróć od MPEG-1 Audio Layer 3) to kodek i kontener w jednym, z rozszerzeniem pliku .mp3. Stał się wszechobecny jako format pobierania muzyki.
- WAV. Format WAV (.wav) to także kodek i kontener w jednym. Ten format jest nieskompresowany, więc nadaje się tylko do bardzo krótkich klipów, takich jak efekty dźwiękowe.
- Ogg + kodk audio Vorbis. Jest to zwykle określane jako Ogg Vorbis i jest obsługiwane z rozszerzeniem pliku .ogg lub .oga.
- MPEG 4 + kodk audio AAC. Dźwięk MPEG4 (.m4a) jest mniej powszechny niż MP3.
- WebM + kodk audio Vorbis. Format WebM (.webm) może również zawierać tylko dźwięk.
- WebM + kodk audio Opus. Opus to nowszy, bardziej wydajny kodek audio, którego można używać z WebM.

Poniższa tabela zawiera wykaz wsparcia przeglądarek dla powyższych formatów audio.

Format	Type	IE	MS Edge	Chrome	Firefox	Opera	Safari	iOS Safari	Android
MP3	audio/mpeg mp3	9.0+	12+	3.0+	22+	15+	4+	4.1	2.3+
WAV	audio/wav or audio/wave	–	12+	8.0+	3.5+	11.5+	4+	3.2+	2.3+
Ogg Vorbis	audio/ogg ogg oga	–	–	4.0+	3.5+	11.5+	–	–	2.3+
MPEG-4/AAC	audio/mp4 m4a	11.0+	12+	12.0+	–	15+	4+	4.1+	3.0+
WebM/Vorbis	audio/webm webm	–	–	6.0+	4.0+	11.5+	–	–	2.3.3+
WebM/Opus	audio/webm webm	–	14+	33+	15+	20+	–	–	–

Chyba nadszedł czas, abyśmy doszli do znaczników dodawania wideo do strony internetowej (w końcu jest to rozdział HTML). Zaczniemy od przykładu, który zakłada, że projektujemy środowisko, w którym dokładnie wiemy, jakiej przeglądarki będzie używać użytkownik. W takim przypadku możemy podać tylko jeden format wideo, używając atrybutu `src` w tagu `video` (tak jak w przypadku obrazu).

```
<!--video_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Video example</title>
</head>
<body>
    <div id="main">
        <video src="videos/sample_640x360.mp4" width="640" height="480" poster="images/R.jpeg" controls
        autoplay>
            Your browser does not support HTML5 video. Get the <a
            href="videos/sample_640x360.mp4">Video</a>
        </video>
    </div>
</body>
</html>
```



Przeglądarki, które nie obsługują video, wyświetlają dowolne treści zawarte w elemencie `video`. W tym przykładzie udostępnia link do filmu, który odwiedzający może pobrać i odtworzyć w innym odtwarzaczu. W tym przykładzie jest też kilka atrybutów, którym warto się szczegółowo przyjrzeć. Atrybuty `width` oraz `height` określają rozmiar pola, które osadzony odtwarzacz multimedialny zajmuje na ekranie. Ogólnie rzecz biorąc, najlepiej jest ustawić wymiary dokładnie tak, aby odpowiadały wymiarom filmu w pikselach. Rozmiar filmu zmieni się, aby dopasować się do ustawionych tutaj wymiarów. Atrybut `poster` podaje lokalizację obrazu, który jest wyświetlany zamiast video przed jego odtworzeniem. Dodanie atrybutu `controls` skłania przeglądarkę do wyświetlenia wbudowanych kontrolek multimedialnych, zazwyczaj przycisku odtwarzania/wstrzymania, „wyszukiwacza”, który pozwala przejść do pozycji w filmie i kontroli głośności. Możliwe jest stworzenie własnego niestandardowego interfejsu odtwarzacza za pomocą CSS i JavaScript, jeśli chcemy uzyskać większą spójność między przeglądarkami. Atrybut `autoplay` powoduje automatyczne rozpoczęcie odtwarzania wideo po pobraniu wystarczającej ilości pliku multimedialnego, aby można było odtwarzać bez zatrzymywania. Ogólnie rzecz biorąc, należy

unikać korzystania z autoodtwarzania na rzecz umożliwienia użytkownikowi decydowania, kiedy film powinien się rozpocząć.

Autoplay nie działa w iOS Safari i niektórych innych przeglądarkach mobilnych w celu ochrony użytkowników przed niepotrzebnym pobieraniem danych. Ponadto element video może użyć atrybutu loop, aby ponownie odtworzyć wideo po jego zakończeniu (ad infinitum), muted w celu odtworzenia ścieżki wideo bez dźwięku i preload, aby zasugerować przeglądarce, czy należy pobrać dane wideo zaraz po załadowaniu strony (preload="auto") lub poczekać, aż użytkownik kliknie przycisk odtwarzania (preload="none"). Ustawienie preload="metadata" ładuje informacje o pliku multimedialnym, ale nie o samym nośniku. Urządzenie może zdecydować, jak najlepiej obsługiwać ustawienie automatyczne; na przykład przeglądarka w smartfonie może chronić wykorzystanie danych przez użytkownika, nie ładując wstępnie multimediów, nawet jeśli jest ustawiona na auto.

Jak widać, nie jest łatwo znaleźć jeden format wideo, który zadowoli wszystkie przeglądarki (choć MPEG4 / H.264 jest blisko). Ponadto dostępne są nowe wydajne formaty wideo, takie jak VP9 i H.265, które nie są obsługiwane w starszych przeglądarkach. Korzystając z elementu source, możemy pozwolić przeglądarkom korzystać z tego, co mogą. W source seria elementów źródłowych wewnętrz elementu video wskazuje każdy plik wideo. Przeglądarki przeglądają listę, aż znajdą taką, którą obsługują, i pobiorą tylko tę wersję. Poniższy przykład zawiera klip wideo w podporządkowanym formacie WebM/VP9 dla obsługiwanych przeglądarek, a także MP4 i Ogg Theora dla innych przeglądarek. Obejmuje to prawie wszystkie przeglądarki obsługujące wideo HTML5.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Video example</title>
</head>
<body>
    <div id="main">
        <video id="video" controls poster="img/poster.jpg">
            <source src="clip.webm" type="video/webm">
            <source src="clip.mp4" type="video/mp4">
            <source src="clip.ogg" type="video/ogg">
            <a href="clip.mp4">Download the MP4 of the clip.</a>
        </video>
    </div>
</body>
</html>
```

Jedną z potężnych zalet elementu video i interfejsu API odtwarzacza multimedialnego jest to, że system pozwala na wiele dostosowań. Możemy zmienić wygląd przycisków sterujących za pomocą CSS i manipulować funkcjonalnością za pomocą JavaScript. Możemy być również zainteresowani wypróbowaniem prefabrykowanego odtwarzacza wideo, który zapewnia dobry wygląd i zaawansowaną wydajność, taką jak obsługa strumieniowych formatów wideo.

Step 14

Jeśli znamy przykład znaczników video, wiemy już, jak dodać dźwięk do strony. Element audio wykorzystuje te same atrybuty, co element video, z wyjątkiem width, height i poster (ponieważ nie ma nic do wyświetlenia). Podobnie jak w przypadku elementu video, możemy udostępnić stos opcji formatu audio za pomocą elementu źródłowego, jak pokazano w poniższym przykładzie.

```

<!--audio_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Audio example</title>
</head>
<body>
    <div id="main">
        <audio id="example" controls preload="auto">
            <source src="audio/sample1.mp3" type="audio/mp3">
            <source src="audio/sample1.ogg" type="audio/ogg">
            <p>Download audio</p>
            <ul>
                <li><a href="audio/sample1.mp3">MP3</a></li>
                <li><a href="audio/sample1.ogg">OGG</a></li>
            </ul>
        </audio>
    </div>
</body>
</html>

```



Jeśli mamy tylko jeden plik audio, możemy zamiast tego użyć atrybutu `src`. Możemy osadzić dźwięk na stronie i ustawić go tak, aby odtwarzał się automatycznie, a następnie zapętlać i nie udostępniać żadnych elementów sterujących, które zatrzymałyby dźwięk dzięki następującemu kodu.

```
<audio src="audio/sample1.ogg" autoplay loop></audio>
```

Element `track` umożliwia dodawanie tekstu zsynchronizowanego z osią czasu ścieżki wideo lub audio. Niektóre zastosowania obejmują:

- Napisy w językach alternatywnych
- Napisy dla osób niedosłyszących
- Opisy tego, co dzieje się w filmie dla osób niedowidzących
- Tytuły rozdziałów umożliwiające nawigację po mediach
- Metadane, które nie są wyświetlane, ale mogą być używane przez skrypty

Oczywiście, dodanie ścieżek tekstowych sprawia, że media są bardziej dostępne, ale ma dodatkową zaletę poprawy SEO (optymalizacji wyszukiwarek). Może również pozwolić na głębokie linkowanie, łączące się z konkretnym miejscem na osi czasu mediów. Użycie elementu `track` należy umieścić wewnątrz elementu wideo lub audio, do którego chcemy dodać adnotację. Element `track` musi pojawić się po wszystkich elementach źródłowych, jeśli takie istnieją, i może zawierać następujące atrybuty. `src` wskazuje na plik tekstowy. `kind` określa typ wprowadzanej adnotacji tekstu (napisy, podpisy, opisy, rozdziały lub metadane). Jeśli rodzaj jest ustawiony na napisy, należy również określić język (atribut `srclang`) za pomocą standardowego dwuliterowego tagu językowego IANA. `label` zawiera nazwę ścieżki, której można użyć w interfejsie do wybrania określonej ścieżki. `default` oznacza określoną ścieżkę jako domyślną i może być używana tylko na jednej ścieżce w elemencie media. Poniższy kod udostępnia opcje napisów w języku angielskim i francuskim dla filmu:

```

<video width="640" height="320" controls>
  <source src="japanese_movie.mp4" type="video/mp4">
  <source src="japanese_movie.webm" type="video/webm">
  <track src="english_subtitles.vtt"
    kind="subtitles"
    srclang="en"
    label="English subtitles"
    default>
  <track src="french.vtt"
    kind="subtitles"
    srclang="fr"
    label="Sous-titres en français">
</video>

```

Zauważmy, że w poprzednim przykładzie ścieżka wskazuje na plik z sufiksem `.vtt`. Jest to plik tekstowy w formacie `WebVTT` (Web Video Text Tracks), który zawiera listę wskazówek. Plik z tym rozszerzeniem przykładowo wygląda następująco:

WEBVTT

```

00:00:01.345 --> 00:00:03.456
Welcome to Artifact [applause]

00:00:06.289 --> 00:00:09.066
There is a lot of new mobile technology to discuss.

00:00:06.289 --> 00:00:13.049
We're glad you could all join us at the Alamo Drafthouse.

```

Napisy są oddzielone pustymi odstępami między wierszami. Każdy napis ma czas rozpoczęcia i zakończenia w formacie `godziny:minuty:sekundy:milisekundy`, oddzielone „strzałką” (`-->`). Tekst (podtytuł, podpis, opis, rozdział lub metadane) znajduje się w wierszu poniżej. Opcjonalnie ID może być dostarczone dla każdej pamięci w wierszu powyżej sekwencji czasowej.

Przykład użycia znajduje się pod adresem: <https://www.iandevlin.com/html5test/webvtt/html5-video-webvtt-sample.html> (<https://www.iandevlin.com/html5test/webvtt/html5-video-webvtt-sample.html>).

Step 15

Kolejnym fajnym dodatkiem w `HTML5` jest element `canvas` i powiązany z nim interfejs `API Canvas`. Element `canvas` tworzy na stronie internetowej obszar do rysowania z zestawem funkcji `JavaScript` do tworzenia linii, kształtów, wypełnień, tekstu, animacji i tak dalej. Możemy go użyć do wyświetlenia ilustracji, ale to, co daje elementowi `canvas` tak duży potencjał (i tak zachwyca świat tworzenia stron internetowych), to to, że wszystko jest generowane za pomocą skryptów. Oznacza to, że jest dynamiczny i może rysować rzeczy w locie i reagować na dane wejściowe użytkownika. To sprawia, że jest to sprytna platforma do tworzenia animacji, gier, a nawet całych aplikacji – wszystko to przy użyciu natywnego zachowania przeglądarki i bez zastrzeżonych wtyczek, takich jak Flash. Warto zauważyć, że obszar rysowania `canvas` jest oparty na rastrowym, co oznacza, że składa się z siatki pikseli. To odróżnia go od innego standardu rysowania, `SVG`, który wykorzystuje kształty wektorowe i ścieżki zdefiniowane za pomocą punktów i matematyki. Dobłą wiadomością jest to, że każda obecna przeglądarka obsługuje element `canvas` w chwili pisania tego tekstu. Stało się tak dobrze, że oprogramowanie `Adobe Animate` (zamiennik Flash Pro) jest teraz eksportowane do formatu `canvas`. Opanowanie elementu `canvas` to więcej, niż możemy się tutaj podjąć, szczególnie bez żadnego doświadczenia z `JavaScriptem`, ale pokażemy, jak to jest rysować za pomocą `JavaScript`.

Dodajemy obszar `canvas` do strony z elementem `canvas` i określamy wymiary za pomocą atrybutów `width` i `height`. I to naprawdę wszystko, co dotyczy znaczników. W przypadku przeglądarek, które nie obsługują elementu `canvas`, możemy umieścić w tagach pewną zawartość zastępczą (wiadomość, obraz lub cokolwiek, co wydaje się właściwe).

```
<!--canvas_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Canvas example</title>
</head>
<body>
    <div id="main">
        <canvas width="600" height="400" id="my_first_canvas">
            Your browser does not support HTML5 canvas. Try using Chrome, Firefox,
            Safari or MS Edge.
        </canvas>
    </div>
</body>
</html>
```

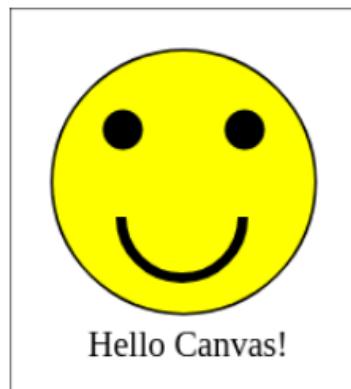
Znacznik po prostu czyści przestrzeń, na której nastąpi rysowanie. Możemy wpływać na samą przestrzeń rysowania za pomocą CSS (na przykład dodać obramowanie lub kolor tła), ale cała zawartość `canvas` jest generowana przez skrypty i nie można jej wybrać do stylizacji za pomocą CSS.

Interfejs API `Canvas` zawiera funkcje do tworzenia kształtów, takie jak `strokeRect()` do rysowania prostokątnego konturu i `beginPath()` do rozpoczętania rysowania linii. Niektóre funkcje przesuwają elementy, takie jak `rotate()` i `scale()`. Zawiera również atrybuty do stosowania stylów (na przykład `lineWidth`, `font`, `stroke-Style` i `fillStyle`). Utworzmy za pomocą interfejsu API `Canvas` prostą buźkę z napisem `Hello Canvas!`.

```

<!--canvas_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Canvas example</title>
</head>
<body>
    <div id="main">
        <canvas width="600" height="400" id="my_first_canvas">
            Your browser does not support HTML5 canvas. Try using Chrome, Firefox,
            Safari or MS Edge.
        </canvas>
    </div>
</body>
<script type="text/javascript">
    window.addEventListener('load', eventWindowLoaded, false);
    function eventWindowLoaded() {
        canvasApp();
    }
    function canvasApp(){
        var theCanvas = document.getElementById('my_first_canvas');
        var my_canvas = theCanvas.getContext('2d');
        my_canvas.strokeStyle = "black";
        my_canvas.lineWidth = 3;
        my_canvas.fillStyle = "yellow";
        my_canvas.stroke();
        my_canvas.fill();
        my_canvas.closePath();
        my_canvas.fillStyle = "black";
        my_canvas.beginPath();
        my_canvas.arc(65, 70, 10, (Math.PI/180)*0, (Math.PI/180)*360, false);
        my_canvas.stroke();
        my_canvas.fill();
        my_canvas.closePath();
        my_canvas.beginPath();
        my_canvas.arc(135, 70, 10, (Math.PI/180)*0, (Math.PI/180)*360, false);
        my_canvas.stroke();
        my_canvas.fill();
        my_canvas.closePath();
        my_canvas.lineWidth = 6;
        my_canvas.beginPath();
        my_canvas.arc(99, 120, 35, (Math.PI/180)*0, (Math.PI/180)*-180, false);
        my_canvas.stroke();
        my_canvas.closePath();
        my_canvas.fillStyle = "black";
        my_canvas.font = '20px _sans';
        my_canvas.fillText ("Hello Canvas!", 45, 200);
    }
</script>
</html>

```



To oczywiście załączek używania canvas w HTML5. Więcej przykładów można znaleźć w książce *The book HTML5 Canvas* (<https://www.oreilly.com/library/view/html5-canvas-2nd/9781449335847/>) (<https://www.oreilly.com/library/view/html5-canvas-2nd/9781449335847/>).

Przykłady znajdują się w repozytorium pod

adresem: https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_2

(https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_2)