Document Object Model

wprowadzenie, instalowanie skryptów, narzędzia deweloperskie, drzewo DOM, wyszukiwanie, modyfikacja, tworzenie i wstawianie elementów, obsługa zdarzeń

Plan zajęć

1. Wprowadzenie
2. Instalowanie skryptów
3. Narzędzia deweloperskie
4. Drzewo DOM
5. Wyszukiwanie elementów
6. Modyfikacja elementów
7. Tworzenie elementów
8. Wstawianie elementów
9. Obsługa zdarzeń

Wprowadzenie

Document Object Model to odzwierciedlenie struktury dokumentu HTML.

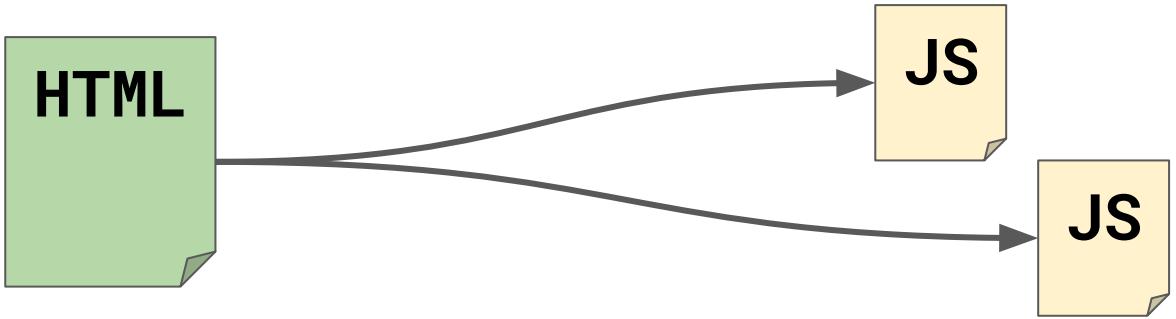
DOM to całkowicie niezależny od języka i platformy **standard**.

Wydany przez **World Wide Web Consortium** w roku 1998.

Instalowanie skryptów

Instalowanie skryptów JavaScript w dokumencie HTML sprowadza się do dodania znacznika `script` z atrybutem `src` określającym ścieżkę do pliku JavaScript.

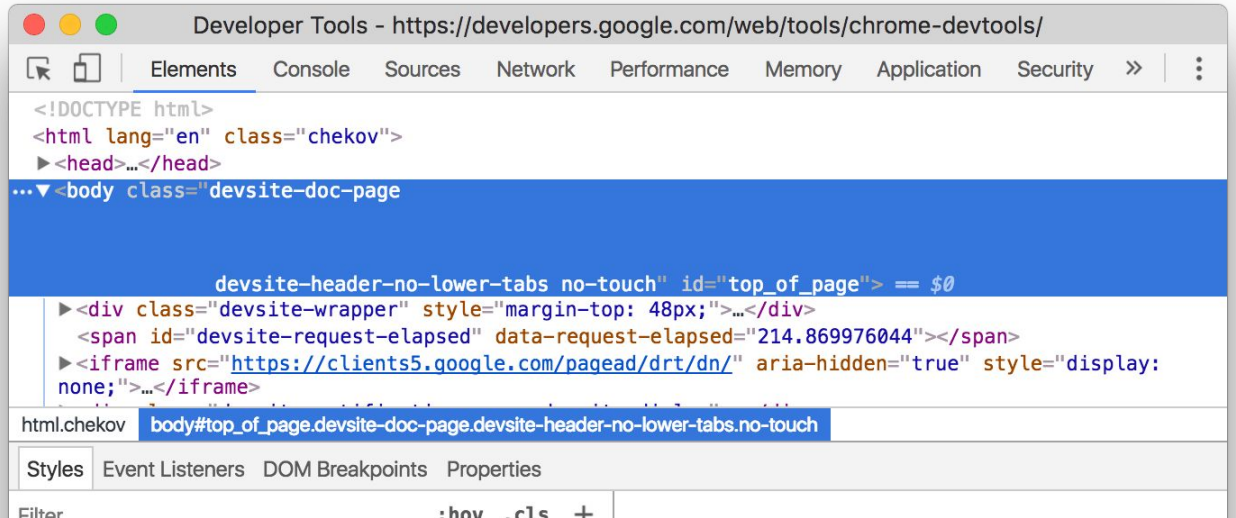
```
<script src="my-website/main.js"></script>
```



Narzędzia deweloperskie

W przeglądarce **Chrome**, narzędzia deweloperskie otwieramy wciskając **F12**.

Najpopularniejsze zakładki to **Elements**, **Console**, **Sources** oraz **Network**.



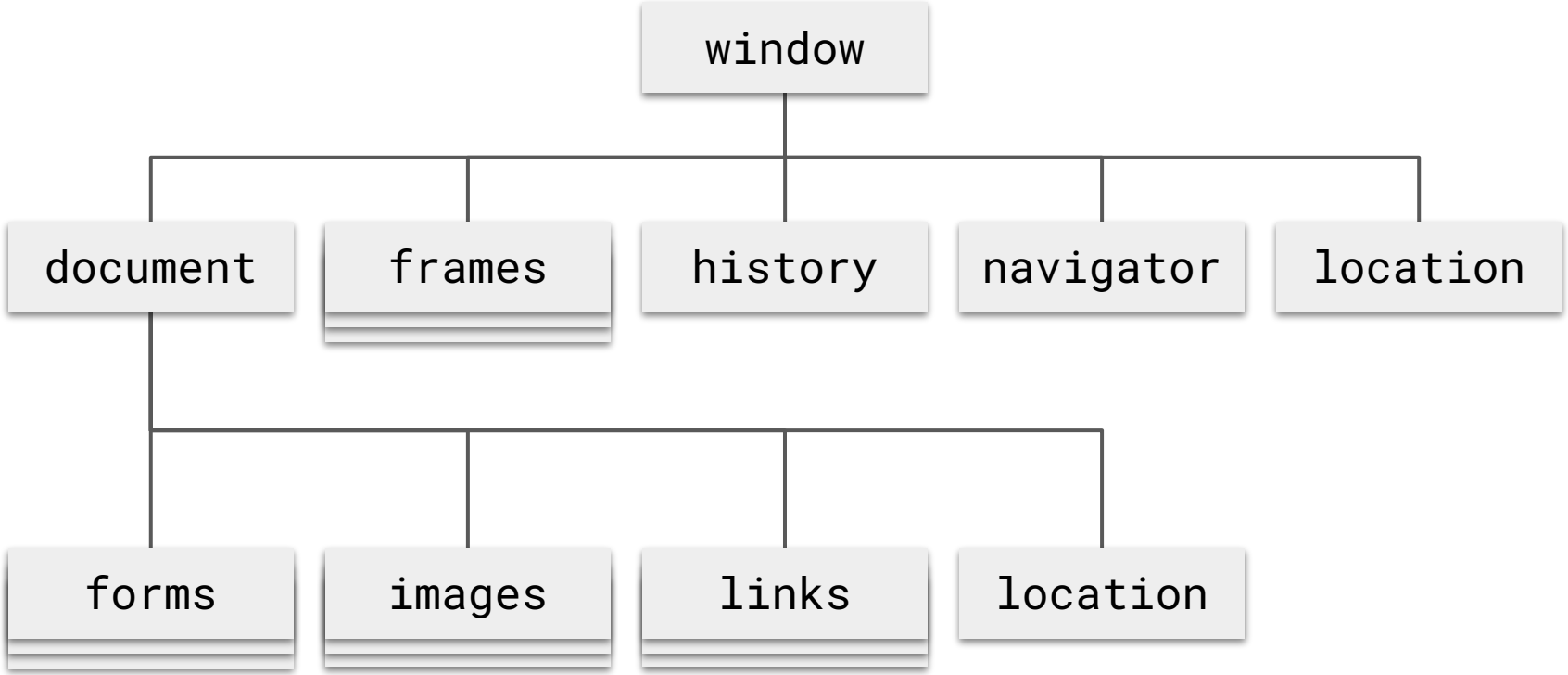
Drzewo DOM

Przeglądarka buduje Document Object Model na podstawie zawartości pliku HTML.

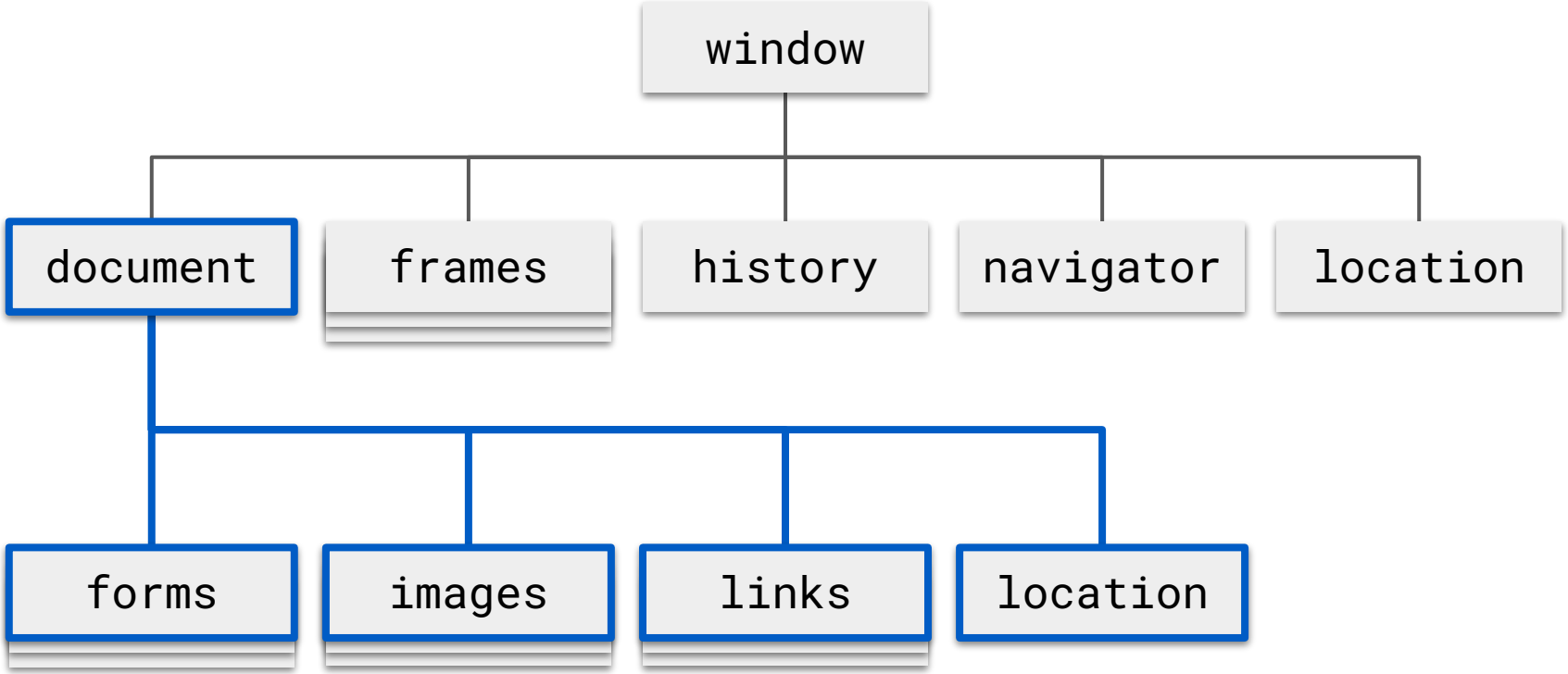
Każdy element wraz ze swoimi atrybutami i zawartością jest umieszczany w DOM.

JavaScript posiada metody niezbędne do pracy z drzewem DOM.

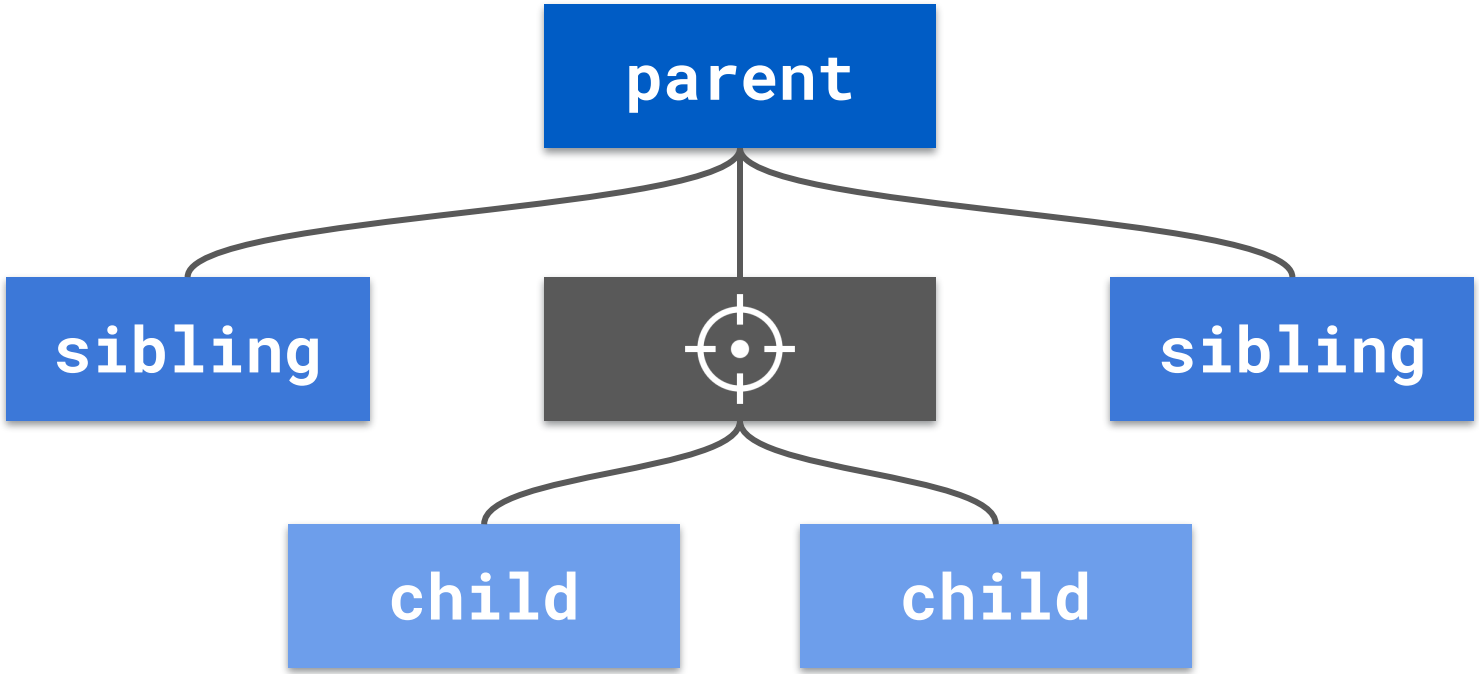
Drzewo DOM / Diagram



Drzewo DOM / Diagram



Drzewo DOM / Hierarchia elementów



Wyszukiwanie elementów

Umiejętne wyszukiwanie elementów HTML stanowi pierwszy krok do ich modyfikacji. Istnieje wiele metod pozwalających ‘uchwycić’ pożądany element, jednak najczęściej wykorzystuje się dwie z nich: `querySelector` i `querySelectorAll`.

Wyszukiwanie elementów / Metody

`getElementById`	zwraca element o wybranym id
`getElementsByTagName`	zwraca elementy o wybranym znaczniku
`getElementsByClassName`	zwraca elementy o wybranej klasie
`getElementsByName`	zwraca elementy o wybranym atrybucie *name*
`querySelector`	zwraca pierwszy element pasujący do selektora CSS
`querySelectorAll`	zwraca wszystkie elementy pasujące do selektora CSS

Wyszukiwanie elementów / Przykłady

```
<section class="news">
    <article> ... </article>
    <article> ... </article>
    <article> ... </article>
</section>

var newsSection = document.getElementsByClassName("news");

// Lista elementów zwrócona przez getElementsByTagName jest "żywa"
var liveList = newsSection.getElementsByTagName("article");
```

Wyszukiwanie elementów / Przykłady

```
<section class="news">
    <article> ... </article>
    <article> ... </article>
    <article> ... </article>
</section>

var newsSection = document.getElementsByClassName("news");

// Lista elementów zwrócona przez querySelectorAll jest "martwa"
var deadList = newsSection.querySelectorAll("article");
```

Wyszukiwanie elementów / Metody

`parentNode`	zwraca węzeł-rodzica
`parentElement`	zwraca element-rodzica
`previousSibling`	zwraca poprzedni element-rodzeństwo
`nextSibling`	zwraca następny element-rodzeństwo
`firstChild`	zwraca pierwszy węzeł-dziecko
`firstElementChild`	zwraca pierwszy element-dziecko
`lastChild`	zwraca ostatni węzeł-dziecko
`lastElementChild`	zwraca ostatni element-dziecko

Wyszukiwanie elementów

Modyfikacja elementów

Do powszechnych operacji na elementach można zaliczyć zmianę zawartości tekstowej, ingerencję w atrybuty, lub ich wartości, a także dodawanie i usuwanie klas oraz styli CSS.

Modyfikacja elementów / Tekst

`element.innerText`	pobiera lub ustawia widzialną zawartość tekstową
`element.textContent`	pobiera lub ustawia całą zawartość tekstową

Modyfikacja elementów / Atrybuty

`element.attributes`	zwraca wszystkie atrybuty
`element.hasAttributes`	sprawdza obecność atrybutów
`element.getAttribute`	zwraca wartość wybranego atrybutu
`element.hasAttribute`	sprawdza obecność wybranego atrybutu
`element.removeAttribute`	usuwa wybrany atrybut
`element.setAttribute`	ustawia wartość wybranego atrybutu
`element.dataset`	pobiera lub ustawia niestandardowy atrybut

Modyfikacja elementów / Klasy

`element.classList.add`	dodaje wybraną klasę do elementu
`element.classList.remove`	usuwa wybraną klasę z elementu
`element.classList.item`	zwraca klasę o wybranym indeksie
`element.classList.toggle`	dodaje lub usuwa wybraną klasę
`element.classList.contains`	sprawdza obecność wybranej klasy
`element.classList.replace`	zastępuje wybraną klasę inną klasą
`element.className`	zwraca ustawione klasy jako ciąg znaków

Modyfikacja elementów / Style

`element.style`	pobiera lub ustawia style CSS
`element.style.backgroundColor`	pobiera lub ustawia background-color
`element.style.fontFamily`	pobiera lub ustawia font-family
`element.style.margin`	pobiera lub ustawia margin
`element.style.padding`	pobiera lub ustawia padding
`element.style.visibility`	pobiera lub ustawia visibility
`element.style.zIndex`	pobiera lub ustawia z-index
`...`	...

Modyfikacja elementów

Tworzenie elementów

Tworzenie elementów, zawartości tekstowej, albo komentarzy to codzienność w pracy dewelopera front-end. JavaScript ma kilka dedykowanych metod, które to umożliwiają.

Tworzenie elementów / Metody

`document.createElement`	tworzy nowy element
`document.createTextNode`	tworzy nowy węzeł tekstowy
`document.createComment`	tworzy nowy komentarz

Tworzenie elementów / Przykłady

```
var myButton = document.createElement(“button”);

myButton.setAttribute(“disabled”, false);
myButton.classList.add(“btn-primary”);
myButton.innerText = “Hello there!”;
```

Tworzenie elementów / Przykłady

```
var myText = document.createTextNode(“How are you?”);

// Doczepia “I’m just fine.” do węzła myText
myText.appendData(“I’m just fine.”);
```

Tworzenie elementów / Przykłady

```
var myComment = document.createComment(“This is a short comment”);


var warning = “Do NOT delete this”;
var comment = document.createComment(warning);
```

Wstawianie elementów

Z pomocą JavaScript, populowanie drzewa DOM nowymi elementami jest bardzo łatwe. Istnieją dedykowane metody, które pozwalają umieścić wybrany element w dowolnym miejscu w hierarchii.

Wstawianie elementów / Metody

`element.before`	wstawia wybrany element jako pierwszy element rodzica
`element.after`	wstawia wybrany element jako ostatni element rodzica
`element.prepend`	wstawia wybrany element jako pierwszy element-dziecko
`element.append`	wstawia wybrany element jako ostatni element-dziecko

Wstawianie elementów / Przykłady

```
var section = document.querySelector(“section”);

section.before(“Header”)
section.prepend(“New article”)
section.append(“Old article”)
section.after(“Footer”)
```

```
Header
<section>
New article
    <article>...</article>
    Old article
</section>
Footer
```

Wstawianie elementów / Przykłady

```
var heading = document.createElement(“h2”);
var welcome = document.createTextNode(“Welcome home!”);

heading.append(text);

var section = document.createElement(“section”);

section.append(heading);
```

Wstawianie elementów / DocumentFragment

DocumentFragment umożliwia budowanie DOM bez naruszania bieżącego dokumentu.

```
// Pierwszy sposób tworzenia fragmentu
var fragment = new DocumentFragment();

// Drugi sposób tworzenia fragmentu
var fragment = document.createDocumentFragment();
```

Wstawianie elementów / DocumentFragment

```
var heading = document.createElement(“h2”);
heading.innerText = “My Document Fragment”;

var text = document.createTextNode(“This is fun.”);

fragment.append(heading);
fragment.append(text);

document.body.append(fragment);
```

Wstawianie elementów

Obsługa zdarzeń

Akcje użytkownika wywołują zdarzenia, które można obłusigwać przy pomocy JavaScript.

Istnieje przeszło kilkadziesiąt zdarzeń ‘organicznych’ emitowanych przez przeglądarkę.

Można też definiować i emitować zdarzenia ‘syntetyczne’ na własne potrzeby.

Obsługa zdarzeń / Rodzaje zdarzeń

Mysz	Klawiatura	Fomularz	Przeglądarka
`click`	`keydown`	`focus`	`resize`
`dblclick`	`keyup`	`blur`	`scroll`
`mousemove`	`keypress`	`change`	`load`
`mouseenter`		`submit`	`unload`
`mouseleave`		`reset`	

Obsługa zdarzeń / Przykłady

```
<button class=”help-btn” onclick=”onClick()”>Help</button>

<script>
    function onClick() {
        alert("Help!");
    }
</script>
```

Obsługa zdarzeń / Metody

`addEventListener`	dodaje obsługę wybranego zdarzenia do elementu
`removeEventListener`	usuwa obsługę wybranego zdarzenia z elementu

Obsługa zdarzeń / Przykłady

```
<button class="help-btn">Help</button>

<script>
    var helpButton = document.querySelector(".help-btn");

    function onClick() {
        alert("Help!");
    }

    helpButton.addEventListener("click", onClick);
</script>
```

Obsługa zdarzeń / Callback

```
// Funkcja ‘obsługująca’ zdarzenie to, tzw. callback
function onClick() {
    alert(“Help!”);
}
```

Obsługa zdarzeń / Callback

```
// Funkcja ‘obsługująca’ zdarzenie to, tzw. callback
function onClick() {
    alert(“Help!”);
}

// Do callbacka zawsze przekazywany jest obiekt Event
function onChange(event) {
    console.log(event.target);
}
```

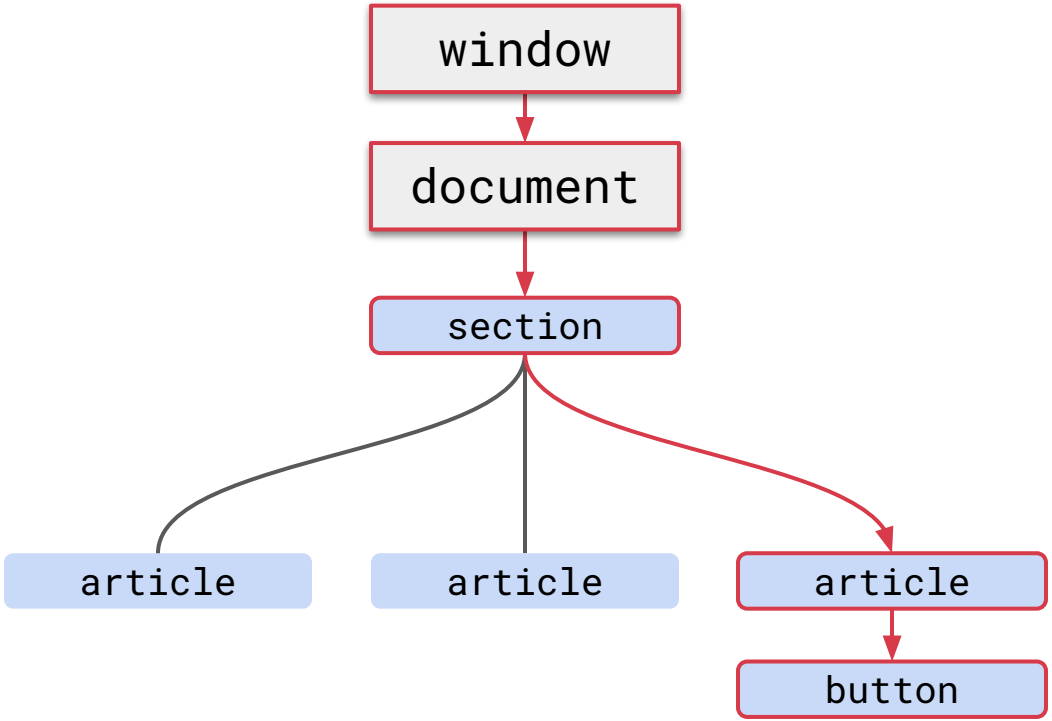
Obsługa zdarzeń / Przykłady

```
<form>
    <input type="text">
    <input type="submit">
</form>

function onSubmit(event) {
    event.preventDefault();
    alert("Submit prevented!");
}

document.querySelector("form").addEventListener("submit", onSubmit);
```

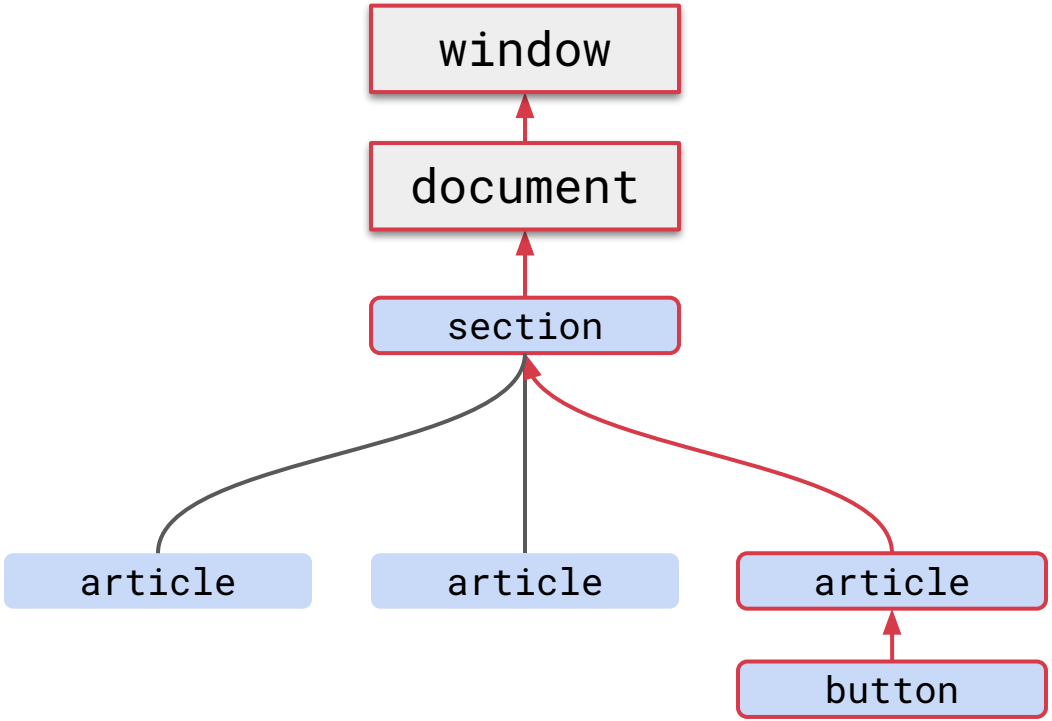
Obsługa zdarzeń / Event capturing



Obsługa zdarzeń / Event capturing

```
myButton.addEventListener(“click”, onSubmit, true);
```

Obsługa zdarzeń / Event bubbling



Obsługa zdarzeń / Event bubbling

```
myButton.addEventListener(“click”, onSubmit, false);
```

Obsługa zdarzeń / DOMContentLoaded

Jednym z najważniejszych zdarzeń jest `DOMContentLoaded`, które jest emitowane przez `document` w momencie załadowania jego zawartości i sparsowania do drzewa DOM.

Obsługa zdarzeń / DOMContentLoaded

```
function runProgram() {
  console.info("Teraz można pracować z DOM");
}

// readyState przyjmuje też wartość “interactive” lub “complete”
if (document.readyState === "loading") {
  document.addEventListener("DOMContentLoaded", runProgram);
} else {
  doSomething();
}
```

Obsługa zdarzeń / load

Zdarzenie `load` jest emitowane, gdy cały dokument—w tym załączone arkusze stylów, czcionki, obrazki oraz zagnieżdżone ramki—został załadowany.

Obsługa zdarzeń / load

```
function runProgram() {
  console.info("Cały dokument został załadowany.");
}

window.addEventListener("load", runProgram);
```

Obsługa zdarzeń

Dziękuję za uwagę.