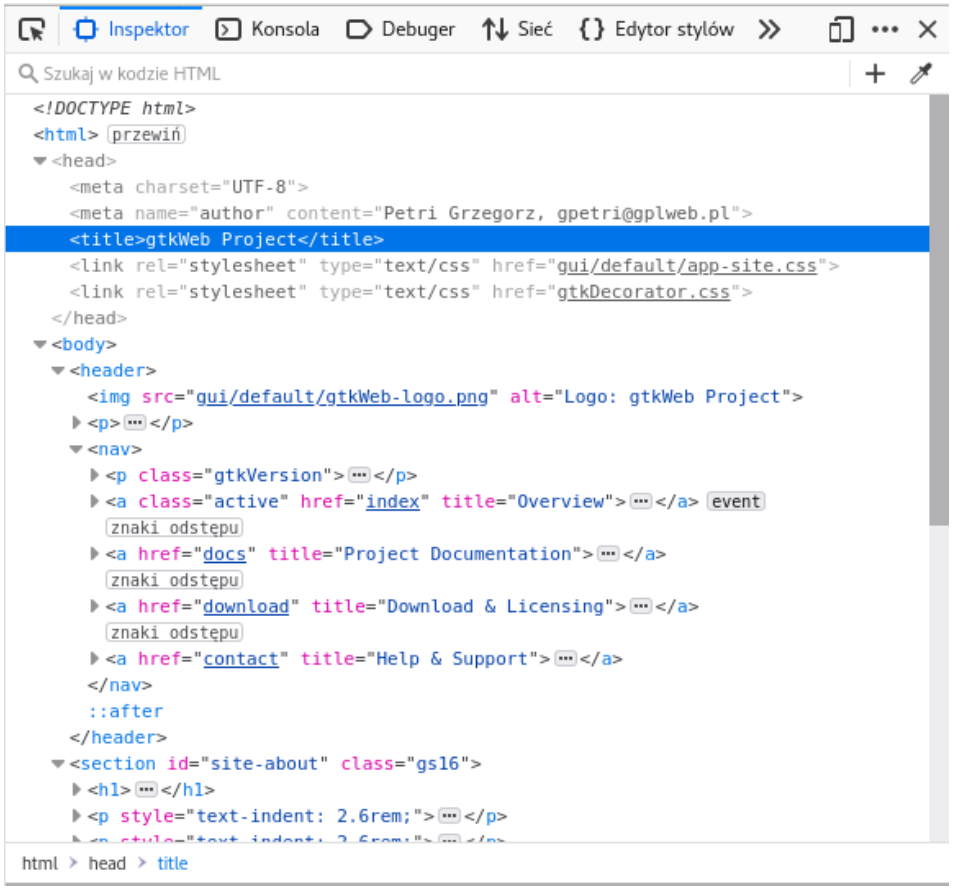
Programowanie Internetowe

- Standard ECMA Script
- Obiektowy model dokumentu
- Data, Czas, Wyrażenia regularne
- Przechowywanie danych
- Przetwarzanie asynchroniczne

Opracował: inż. Grzegorz Petri

Przegląd zagadnień

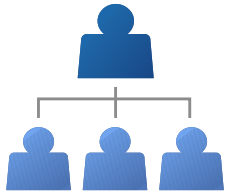
- Standard DOM
- Poziomy implementacji
- Węzły i ich typy
- Manipulacja elementami
  - Węzły
  - Style
  - Dodawanie
  - Modyfikacja
  - Usuwanie

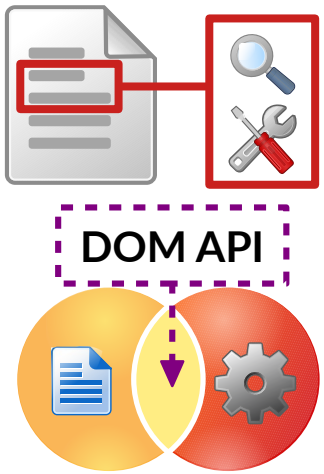


Document Object Model

Czyli obiektowy model dokumentu

```
let Book = {
     format: „A4”,
     pages:  54
}
```





- Jest obiektową reprezentacją danych, które zawierają strukturę oraz zawartość dokumentu
- Przedstawia dokument (*ciąg znakowy*) jako hierarchiczną strukturę drzewa złożoną z umiejscowionych w odpowiednich miejscach elementów /obiektów.
- to gotowy parser – ułatwia programistom dostęp do wybranych części dokumentu bez strachu o naruszenie spójności struktury reszty dokumentu.
- DOM API łączy dokumenty ze środowiskiem programistycznym, np. skryptami JavaScript, czy PHP poprzez udostępnianie interfejsów do **odczytu** oraz **manipulacji** elementami DOM.

DOM Levels

Poziomy implementacji standardu

Określenie poziomu* implementacji funkcjonalności modelu:

- **DOM Level 1** *pełny model obsługi elementów dokumentów HTML i XML*
- **DOM Level 2** *(2000) model obsługi zdarzeń, przestrzenie nazw XML i CSS*
- **DOM Level 3** *(2004) zdarzenia klawiatury, XPath, serializacja dokumentów jako XML*
- **DOM Level 4** *(2015) tzw. living standard, czyli migawka aktualnych prac nad standardem*

Nieoficjalny poziom – nie jest standardem W3C:

- **DOM Level 0** – *pochodzi od przeglądarki Netscape Navigator 3.0 i jako taki został zaimplementowany we wszystkich przeglądarkach internetowych.*

**Poziom (Level) można traktować jako wersję standardu, który może zostać zaimplementowany w danym rozwiązaniu.*

DOM Levels

Poziomy implementacji standardu

Określenie poziomu implementacji funkcjonalności modelu:

- Typy Dokumentów posiadające wspólne interfejsy DOM Level 2:
  - **HTML** – dokumenty oparte na języku znaczników renderowane w oknie przeglądarki – posiadają dodatkowe właściwości i metody oraz obsługę zdarzeń
  - **XML** – dokumenty również oparte na języku znaczników, które nie są bezpośrednio renderowane w oknie przeglądarki – są najczęściej formatem pośrednim lub wymiany danych.
- Szczegółowe specyfikacje DOM Level 3:
  - Core
  - Load and Save
  - Xpath
  - Views & Formatting
  - Requirements
  - Validation

Terminologia DOM

Elementy i węzły

Node / węzeł / element

- to obiekt bazowy **`Node`** dla pozostałych węzłów występujących w dokumencie
- to wspólny **`interfejs`** dla różnych typów obiektów dziedziczących te same właściwości oraz metody,
- to węzły różnego typu traktowane oraz testowalne w taki sam sposób

W podstawowych zastosowaniach wyszczególniamy 3 typy węzłów:

- Węzeł elementu (*element node*)
- Węzeł tekstowy (*text node*)
- ***Węzeł atrybutu (attribute node) – przestarzały w DOM Level 4***

Oraz 9 innych typów węzła z 12 opisanych i ponumerowanych w specyfikacji NodeType

Sprawdzenie typu węzła

Node.nodeType

- Typ węzła można sprawdzić wywołując właściwość **`nodeType`** elementu
- Jest to właściwość READ-ONLY (*tylko do odczytu*) typu **`Integer`**
- Wywołanie właściwości zwraca numer jednego z **12** typów węzłów, takich jak: *element*, *tekst*, czy *komentarz* zgodnie z tabelą następnego slajdu
- Sprawdzić typ można porównując zwracany *numer* lub *ENUM* obiektu Node

```
var node = document.documentElement.firstChild;
if (node.nodeType !== Node.COMMENT_NODE) {
  console.warn("Powinieneś komentować swój kod!");
}
```

Typy węzłów

Element reprezentujący przykład
Przestarzały Typ węzła usunięty w DOM4

Typ	*Nazwana stała*	*Przykład*
1	ELEMENT_NODE	HTML: <DIV class=”art”><P>Akapit</P></DIV>
2	*ATTRIBUTE_NODE*	HTML: <DIV class=”art”><P>Akapit</P></DIV>
3	TEXT_NODE	HTML: <DIV class=”art”><P>Akapit</P></DIV>
4	CDATA_SECTION_NODE	HTML: <DIV class=”art”> <!CDATA[[<P>Akapit</P>]]> </DIV>
5	*ENTITY_REFERENCE_NODE*	XML: &entity;
6	*ENTITY_NODE*	XML: <!ENTITY … >
7	PROCESSING_INSTRUCTION_NODE	XML: <?xml-stylesheet … ?>
8	COMMENT_NODE	HTML: <DIV class=”art”><!-- artykuł --><P>Akapit
9	DOCUMENT_NODE	Węzeł dokumentu
10	DOCUMENT_TYPE_NODE	<!DOCTYPE html>
11	DOCUMENT_FRAGMENT_NODE	Węzeł fragmentu dokumentu
12	*NOTATION_NODE*	XML: <!NOTATION … >

Etapy dodawania nowego węzła

Dodawanie, modyfikacja, usuwanie

- ✔ Utworzenie węzła w pamięci
- ✔ Przypisanie mu właściwości (*metodą setAttribute() lub poprzez właściwość*)
- ✔ Dostęp do węzła, który stanie się rodzicem (*gdzie umieścimy nowy węzeł*)
- ✔ Dodanie nowego węzła do rodzica (*jedną z metod w wybranym miejscu*)

```
let node = document.createElement( tag );
    node.setAttribute('class','nazwaKlasy');
    node.id = 'identyfikator';
let parent = document.getElementById( 'ajdi' );
    parent.appendChild(node);
```

Dostęp do węzłów

Różne metody dostępowe

Metoda	*Przyjmuje*	*Zwraca*
`getElementById(value)`	Ciąg znakowy – **nazwa elementu***	Pojedynczy element
`getElementsByTagName(value)`		Kolekcję elementów []
`getElementsByClassName(value)`		Kolekcję elementów []
`querySelector(selektorCSS)`	Ciąg znakowy – format **selektora CSS****	Pojedynczy element
`querySelectorAll(selektorCSS)`		Kolekcję elementów []

W obu przypadkach – dostęp poprzez element oraz dostęp poprzez selektor – jest realizowany przez ciąg znakowy, czyli *String*, to istnieje różnica w sposobie zapisu - formacie.

Nazwa **elementu np.: tag **P** nazwa klasy: **klasa** oraz identyfikator: **id***

Nazwa **selektora** np.: tag **P** nazwa klasy: **.klasa** identyfikatora: **#id** czy atrybutu **[atrybut=wartość]

Dostęp do węzłów

Właściwości dostępowe

Właściwość	*Znaczenie*	*Zwraca*
`parentNode`	Węzeł rodzica (nadrzędny)	Pojedynczy element
`childNodes`	Kolekcja węzłów dzieci danego elementu	Kolekcja elementów
`firstChild`	Pierwsze dziecko wskazanego elementu	Pojedynczy element
`lastChild`	Ostatnie dziecko wskazanego elementu	Pojedynczy element
`previousSibling`	Przedni węzeł* w elemencie (rodzeństwo)	Pojedynczy element
`nextSibling`	Następny węzeł* w elemencie (rodzeństwo)	Pojedynczy element

Właściwości są tylko do odczytu (READ-ONLY), ponieważ przechowują wskaźnik na dany element.

**Węzłem rodzeństwa dla danego elementu może być węzeł z elementem (znacznik) lub węzeł tekstowy (komentarz lub białe znaki – np. nowej linii)*

Typy i zawartość węzłów

Dodawanie, modyfikacja, usuwanie

Właściwości informacji o węźle:

- **`nodeName`** – nazwa węzła,
  zwraca przypisaną nazwaną stałą węzła
- **`nodeType`** – kod węzła,
  zwraca numer reprezentujący typ węzła.
- **`nodeValue`** – zwracane wartości:
  - dla dokumentu nodeValue zwraca `null`,
  - dla węzłów tekstowych, komentarzy i CDATA nodeValue zwraca zawartość węzła.
  - dla węzłów atrybutów, zwracana jest wartość atrybutu.

Właściwości pozawalające na manipulację istniejącym węzłem:

- **`tagName`** – zwraca nazwę tagu (*w HTML nazwę kanoniczną, w XML z zachowaniem wielkości liter*)
- **`innerHTML`** (*odczyt / zapis*) pozwala zapisać nowy węzeł w węźle – element lub tekst
- **`textContent`** (*odczyt / zapis*) – odczytuje lub zapisuje treść węzła*

**Jeśli węzeł to sekcja CDATA, komentarz, instrukcja przetwarzania lub węzeł tekstowy, textContent zwraca tekst wewnątrz tego węzła*

Manipulacja węzłami

Dodawanie, modyfikacja, usuwanie

Metody dodawania nowego węzła:

- AppendChild()
- CloneNode()
- InsertBefore()
- replaceChild()

Pozostałe metody manipulacji istniejącym węzłem:

- RemoveChild()

Atrybuty (właściwości) węzła

Dodawanie, modyfikacja, usuwanie

Metody manipulacji atrybutami:

- `setAttribute()`
- `getAttribute()`
- `hasAttribute()`
- `removeAttribute()`

Metody obsługujące atrybuty w *przestrzeni nazw* mają na końcu NS `<metoda>NS()`:

- `setAttributeNS()`
- `getAttributeNS()`
- `hasAttributeNS()`

Dostęp do atrybutów poprzez Właściwość elementu:

- `attributes` – zwraca kolekcję elementów

```
dialog(
  ‘Pytania?’
);
```