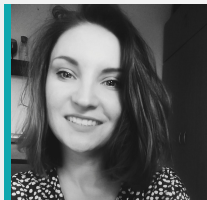




JS DOM

HTML / CSS

Prowadząca



Anna Rodziewicz
Senior Front-end
Developer

Plan gry

- + co to jest **DOM**
- + nawigacja po drzewie
- + manipulacje 🤔
- + Events & EventListeners

Osadzanie JS'a

Bezpośrednio w HTML'u pomiędzy tagami **script**:

```
<script>  
    alert('Hello world!');  
</script>
```



Osadzanie JS'a

Osadzenie **pliku** JS:

```
<script src="js/script.js"></script>
```





DOM - *Document Object Model*

Reprezentacja logiczna **struktury dokumentu**.

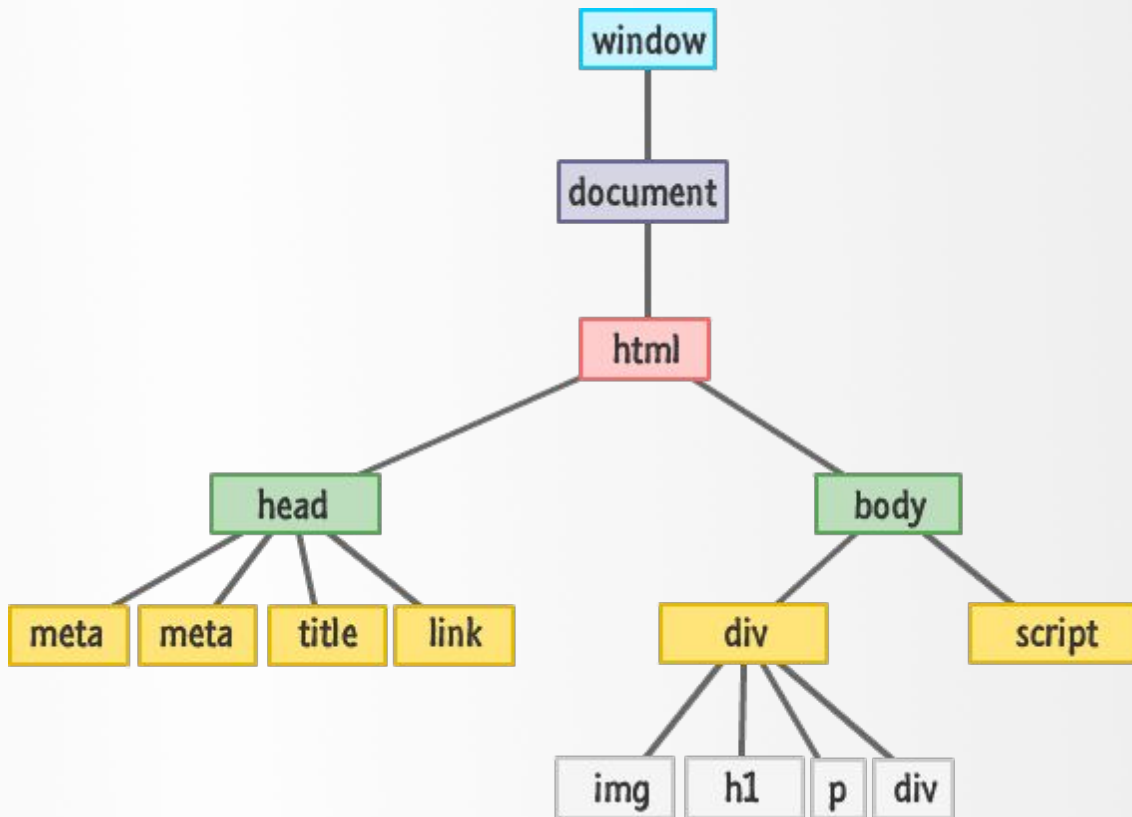
Zapewnia jej połączenie ze skryptem lub językiem programowania.

API jest niezależne od języka programowania.

DOM - *Document Object Model*

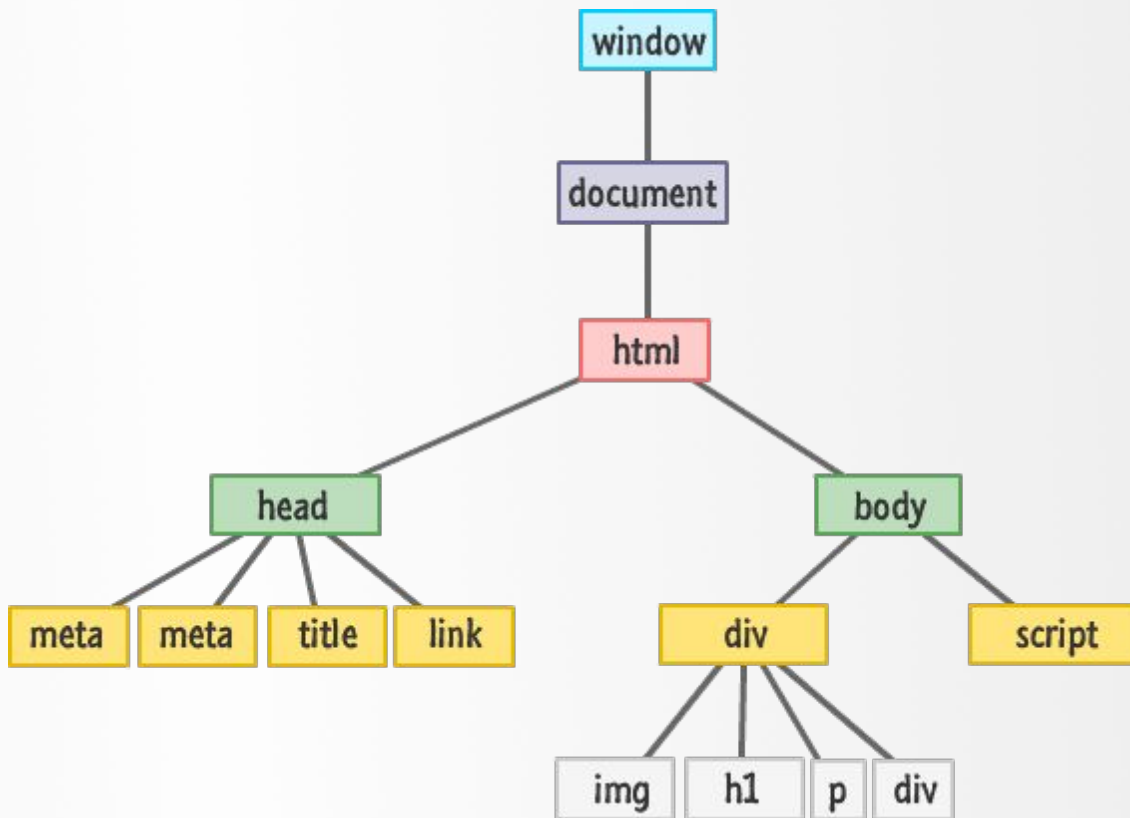
Każda gałąź (**branch**)
kończy się węzłem
(**node**), każdy węzeł
zawiera obiekt (**object**).

Każda gałąź może mieć
dołączone wydarzenie
(**event**).



DOM - *Document Object Model*

Window - reprezentuje
okno, zawierające
dokument DOM.



DOM - *Document Object Model*

// window.alert() - wyświetlenie okna dialogowego

// window.open() - otwarcie nowego okna

// window.close() - zamknięcie okna

// window.prompt() - wyświetlenie okna dialogowe z polem tekstowym

DOM - *Document Object Model*

// `window.alert(message);` - wiadomość w oknie dialogowym

// `window.prompt(message);` - wiadomość w oknie dialogowym z możliwością wpisania wiadomości

// `window.confirm(message);` - wiadomość w oknie dialogowym z możliwością potwierdzenia

// `window.open(url);` - otwiera okno z podanym adresem

// Zadanie

Zadeklaruj funkcję, w której:

// zapytasz użytkownika, **czy ma ochotę** odpowiedzieć na więcej pytań.

// jeśli się zgodzi - zapytaj o **imię**.

// następnie wyświetl okno dialogowe z przywitaniem.



DOM - *Document Object Model*

// window.location; - właściwość, służąca do obsługi adresu **url**
okna

// window.location = url; - zmiana adresu **url**

// Zadanie

Zadeklaruj funkcję, w której:

// zapytasz użytkownika o **url**,

// a następnie otworzysz **url** w nowym oknie.



Wyszukiwanie elementów DOM

// document.getElementById(); - pobieranie elementu po **Id**
>> zwraca referencję do elementu

// document.getElementsByClassName(); - pobieranie
elementów po **nazwie klasy**

>> zwraca HTML Collection (*array-like*). Możemy odwoływać się
po **indeksie** elementu, ale nie możemy wywoływać wszystkich
dostępnych metod dla tablic.

// Zadanie

// dodaj **5** paragrafów w
index.html

Za pomocą JS'a ustaw zielone tło
dla pierwszego oraz czerwone
dla ostatniego elementu.



Wyszukiwanie elementów DOM

// document.getElementsByTagName() ; - pobieranie
elementów po **nazwie tagu**

>> zwraca HTML Collection

// Zadanie

Za pomocą JS'a odwołaj się do
elementu wewnątrz
#container



Wyszukiwanie elementów DOM

// document.querySelector(); - pobieranie elementu po
selektorze lub grupie selektorów

>> zwraca referencję do pierwszego pasującego elementu

// document.querySelectorAll(); - pobieranie elementów po
selektorze lub grupie selektorów

>> zwraca NodeList - podobne do HTML Collection, ale posiada metody dostępne dla Array

// Zadanie

Dodaj 3 elementu do index.html, tak aby co najmniej jeden z nich posiadał:

// id,

// klasę.

Odnajdź je za pomocą JS'a, przypisz do zmiennej i wyświetl za pomocą console.log



Wyszukiwanie elementów DOM

// document.querySelector(); - pobieranie elementu po
selektorze lub grupie selektorów

>> zwraca referencję do pierwszego pasującego elementu

// document.querySelectorAll(); - pobieranie elementów po
selektorze lub grupie selektorów

>> zwraca NodeList - podobne do HTML Collection, ale posiada metody dostępne dla Array

Wyszukiwanie elementów DOM

```
// element.parentNode;
```

właściwość, przechowuje rodzica elementu

```
// element.firstChild;
```

właściwość, przechowuje pierwsze dziecko lub **null**

// Zadanie

Za pomocą JS'a:

// odwołaj się do elementu wewnątrz **#container**, dodaj szarą ramkę.

// do parenta elementu **#container** dodaj czerwoną ramkę.



Wyszukiwanie elementów DOM

```
// element.nextSibling;
```

właściwość, przechowuje najbliższy następujący **po** elemencie element lub **null**

```
// element.previousSibling;
```

właściwość, przechowuje najbliższy następujący **przed** elementem element lub **null**

// Zadanie

Do jednego z utworzonych paragrafów dodaj **id**.

Za pomocą JS'a:

// przypisz go do zmiennej

// odwołaj się do elementu przed

// odwołaj się do elementu po



Usuwanie elementów DOM

// `element.remove()`; - metoda, usuwa element z drzewa

Dodawanie elementów DOM

```
// document.createElement();
```

tworzy element o określonym tagu

```
const el = document.createElement('header');
```

Dodawanie elementów DOM

```
// element.appendChild();
```

metoda, dodaje element na końcu listy dzieci elementów

```
// element.insertBefore();
```

metoda, dodaje element do listy dzieci elementów

```
parentEl.insertBefore(newEl, referenceEl);
```

// Zadanie

Za pomocą JS'a dwa elementy z tagiem **p** w **#container**. Jeden powinien pojawić się na pierwszym miejscu na liście dzieci, drugi jako ostatni.



Dodawanie elementów DOM

```
// document.createTextNode();
```

metoda, tworzy węzeł tekstowy

```
let text = "Hello!"
```

```
let textNode = document.createTextNode(text);
```

```
element.appendChild(textNode);
```


// Zadanie

Za pomocą JS'a dodaj tekst do nagłówka **h1** przypisując mu zmienną, następnie: przypisz nową wartość do zmiennej, podmień go ponownie.



Zarządzanie drzewem DOM

```
// element.hasChildNodes();
```

zwraca `true` lub `false`, jeśli element posiada dzieci

```
// element.innerHTML;
```

właściwość, przechowująca zawartość elementu

```
// element.innerText;
```

właściwość, przechowująca tekstową zawartość elementu

// Zadanie

Użyj tylko JS'a.

// stwórz **div**, w którym umieścisz **paragraf**.

// następnie dodaj swoje imię i nazwisko do paragrafu.



Zarządzanie klasami

```
// element.classList;
```

właściwość, zwraca kolekcję nazw klas danego elementu

dwie podstawowe **metody**:

```
// add ("class");
```

- dodaj *class* do kolekcji nazw klas, jeśli nie istnieje

```
// remove ("class");
```

- usuń *class* z kolekcji



Zarządzanie klasami

```
element.classList.add("test")
```

```
element.classList.add("remove")
```



Zarządzanie klasami

```
// element.className;
```

właściwość, zwraca wartość atrybutu class

Zarządzanie atrybutami

```
// element.removeAttribute() ;
```

```
// element.setAttribute() ;
```

metoda, służąca do ustawiania wartości atrybutu.

// Jeśli atrybut **nie istnieje** - zostaje dodany do elementu,

// Jeśli atrybut **istnieje** - jego wartość jest aktualizowana.

```
// element.setAttribute(attributeName, attributeValue) ;
```

Zarządzanie atrybutami

```
var b = document.querySelector("button");  
b.setAttribute("name", "helloButton");  
b.setAttribute("disabled", "");
```

Zarządzanie stylami

```
div.style.color = 'red';
```

CSS	JavaScript
background	background
background-attachment	backgroundAttachment
background-color	backgroundColor
background-image	backgroundImage
background-position	backgroundPosition
background-repeat	backgroundRepeat
border	border
border-bottom	borderBottom
border-bottom-color	borderBottomColor
border-bottom-style	borderBottomStyle
border-bottom-width	borderBottomWidth

// Zadanie

// zapytaj użytkownika o ulubiony kolor,

// ustaw ten kolor jako tło **body**.



Zarządzanie formularzem

`element.value` - **właściwość**, która pozwala na ustawianie i pobieranie wartości inputu

```
<input name = "text" />
```

```
input.value = 'Hello input';
```

```
const val = input.value; >> 'Hello input'
```

Zarządzanie formularzem

`element.submit()` - **metoda**, która pozwala na *zatwierdzenie* formularza.

```
var form = document.querySelector('form');  
form.submit();
```


// Zadanie

W index.html dodaj formularz kontaktowy.

Zapytaj użytkownika o **imię**, a następnie o **e-mail**. Uzupełnij danymi formularz.

Jeśli imię zostało podane oraz jeśli podany e-mail będzie prawidłowy*
- automatycznie zatwierdź formularz.



Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Events

```
<button onclick="alert('Click!')">Click!</button>
```



// Zadanie

W index.html dodaj przycisk, który po naciśnięciu:

- // wyświetli alert

- // zmieni kolor elementu body

- // po upływie 5 sekund zamknie okno*.



Events

```
element.addEventListener(type, listener[, options]);
```

type - typ nasłuchiwanego wydarzenia (podany jako string)

listener - obiekt, wywoływany po wywołaniu wydarzenia
(najczęściej funkcja)

```
element.removeEventListener();
```



Events

// funkcja

```
element.addEventListener("click", myFunction);  
function myFunction() {  
    alert ("Hello World!");  
}
```

// funkcja anonimowa

```
element.addEventListener("click", function() {  
    alert("Hello World!");  
});
```

Events

Event	Description	Belongs To
<u>abort</u>	The event occurs when the loading of a media is aborted	<u>UIEvent</u> , <u>Event</u>
<u>afterprint</u>	The event occurs when a page has started printing, or if the print dialogue box has been closed	<u>Event</u>
<u>animationend</u>	The event occurs when a CSS animation has completed	<u>AnimationEvent</u>
<u>animationiteration</u>	The event occurs when a CSS animation is repeated	<u>AnimationEvent</u>
<u>animationstart</u>	The event occurs when a CSS animation has started	<u>AnimationEvent</u>
<u>beforeprint</u>	The event occurs when a page is about to be printed	<u>Event</u>
<u>beforeunload</u>	The event occurs before the document is about to be unloaded	<u>UIEvent</u> , <u>Event</u>
<u>blur</u>	The event occurs when an element loses focus	<u>FocusEvent</u>
<u>canplay</u>	The event occurs when the browser can start playing the media (when it has buffered enough to begin)	<u>Event</u>
<u>canplaythrough</u>	The event occurs when the browser can play through the media without stopping for buffering	<u>Event</u>
<u>change</u>	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)	<u>Event</u>

// Zadanie

W index.html dodaj element z **czernym** tłem, który będzie zmieniał kolor na **niebieski** po najechaniu na niego kursorem.

Kolor czarny powinien powracać, gdy kursor będzie poza elementem.

Użyj tylko JS'a



// Zadanie

W index.html dodaj i ostyluj element - **parent**, który po naciśnięciu:

// spowoduje utworzenie nowego węzła tekstowego (np *emoji*)

// dodanie nowego elementu z węzłem tekstowym - **child**



// Zadanie

Zmodyfikuj poprzednią funkcję
tak, aby:

// z elementu **parent** **wyskakiwały**
elementy **child**,

// elementy **child** powinny spadać
w randomowe miejsca *



// Zadanie

Zmodyfikuj poprzednią funkcję
tak, aby:

// **parent** był wycentrowany i
niezależny od scrollowania,

// elementy **child** **wyskakiwały**
podczas scrollowania,





Kontakt

Dziękuję!

Ania Rodziewicz

aerodziejewicz@gmail.com