



Di Mon

JavaScript: Obiekty

dimon.work/kurs.html

1. Obiekty / Asocjacyjny zbiór

Obiekty ?!



Obiekty w JavaScript

```
1
2
3   let person = {
4       name: 'Jhon',
5       lastName: 'Smith',
6       age: 28,
7       city: 'London',
8       canDrive: true
9   }
10
11
12   console.log( person.name, person.lastName );
13
14   person.country = 'UK';
15   person['phone'] = '+1234567890';
16
17
```

Obiekty w JavaScript to struktura danych (tzw. tablica asocjacyjna), w której kluczami (**indeksami**) do komórek z danymi są ciągi znaków. **Obiekty w JavaScript** są używane wszędzie, **ciągi znaków, zbiory**, a nawet każdy **tag** - jest **obiektem**.

Obiekt - struktura referencyjna

Zmienne nie przechowują samych **obiektów**, ale **linki** do obszarów pamięci, w których znajdują się obiekty, więc podczas „kopiowania” zmiennej przypisywane jest **odniesienie do obiektu**. Obie zmienne umożliwiają pracę z tym samym **obiektem**.

2. Document Object Model

DOM

Lorem ipsum dolor sit amet consectetur adipisicing elit.
Quam hic quae quisquam corporis quibusdam rerum libero.

Click Me!

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Officia delectus doloribus assumenda incidunt, quos, quaerat illo possimus, architecto enim odit nam! Distinctio consectetur non, rerum possimus officia, consequuntur esse iste sequi ab quisquam magnam quae voluptatibus illo ex amet dolores hic dolor nisi neque quia? Dolores officiis aperiam error nihil ad dolorum eligendi! Pariatur fuga nobis tempore eum obcaecati laborum dolore asperiores nemo cupiditate dolor quos fugiat excepturi, dolorem ipsum voluptatum ipsam eaque nam? Delectus dicta esse ipsam ab, aut voluptas sint labore aliquid rem, voluptatem beatae saepe repudiandae reprehenderit eius eligendi debitis minima quia quod veritatis. Aut architecto veniam laborum unde ullam similique voluptate provident, laboriosam corporis praesentium labore. Ex ab, asperiores voluptatum inventore, velit quam, omnis beatae eius pariatur quae quas quos blanditiis dicta illum. Dolore repellendus soluta animi doloremque tempore consequuntur, exercitationem debitis a architecto ut aliquam, fugit quisquam laudantium dicta voluptates nam officia, incidunt velit nesciunt quia maxime earum alias cum molestias? Architecto, consequatur optio. Consectetur quis iusto aut adipisci odio, iure nesciunt corporis illum quibusdam culpa error, cupiditate, tenetur sit doloribus esse amet laborum sed quia dolorum? Et atque accusantium soluta quibusdam quisquam rerum necessitatibus commodi eos! Saepe, nulla aspernatur pariatur dolorum dicta velit non.

Dolor eius, corporis repudiandae similique facere veritatis. Vero illo quisquam facilis vel molestiae voluptate natus consequatur deserunt omnis error ex, amet quo esse, nisi alias laborum eos temporibus nihil doloremque consectetur tempore ut. Pariatur nam, cupiditate natus ex non quam sapiente, similique quod dolor veritatis velit, iste mollitia. Perferendis, aliquam. Esse, corporis eum nisi, et eveniet sequi possimus deserunt porro eius, maiores natus debitis. Voluptate, deserunt reiciendis, harum iure illo ratione accusamus, dignissimos molestiae placeat dolores vitae autem voluptates laborum id expedita pariatur similique

Pliki do pobrania:

dimon.work/kurs/assets/les23.zip

DOM – Document Object Model

(obiektowy model dokumentu)

Standard, który definiuje, z jakich **obiektów** przeglądarka buduje **drzewo dokumentu** i jakie **właściwości** mają te **obiekty**. Zgodnie ze standardem **DOM**, każdy **tag dokumentu HTML** jest reprezentowany przez **obiekt w JavaScript**.

Aby zarządzać tagiem, należy go najpierw znaleźć...

`document.querySelectorAll("css_selector")`

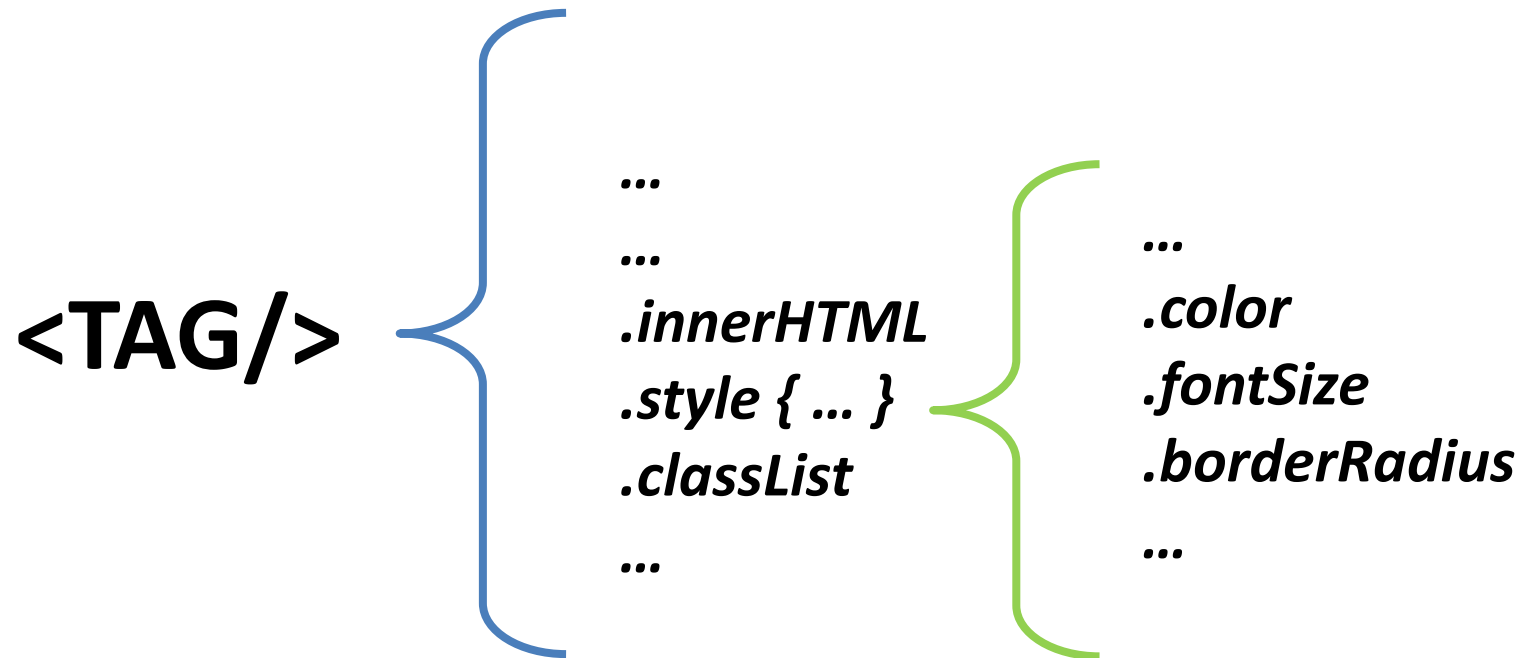
zwraca **pseudo zbiór** obiektów (tagów), które pasują do **css selektora** przekazanego jako **parametr** funkcji;

`document.querySelector("css_selector")`

zwraca **pierwszy obiekt** (tag) znaleziony w dokumencie, który **odpowiada css selektorowi** przekazanemu jako **parametr** funkcji;

id – elementy, które mają atrybut **id**, mogą być używane bez wyszukiwania, takie elementy są dostępne jako **zmienne globalne** (z nazwą pasującą do **id**).

Z czego wykonany jest tag?



Każdy **tag** w JavaScript jest reprezentowany przez **obiekt**, który przechowuje całą **zawartość**, wszystkie **style** i wszystkie **atrybuty tagu**. Oczywiście można je **zmieniać**.

Zmiana zawartości elementu i/lub jego właściwości

Tagi (**elementy dokumentu HTML**) mają szereg właściwości, które określają ich zawartość i wygląd:

...

.innerHTML – właściwość definiująca (lub ustawiająca) zawartość znacznika, tj. wszystko pomiędzy znacznikiem otwierającym i zamykającym;

.style – właściwość definiująca obiekt ze wszystkimi właściwościami stylu obsługiwanymi przez przeglądarkę;

.classList – definiująca listę klas znaczników (jako zbiór, metody **.classList.add()** i **.classList.remove()** umożliwiają dodawanie i usuwanie klas znaczników). Metoda **.classList.contains()** pozwala sprawdzić, czy dana klasa znajduje się na liście.

...

3. JavaScript i interaktywność

Wydarzenia i interaktywność

Funkcje są ściśle związane z obsługą zdarzeń, możemy powiedzieć przeglądarce, **która funkcja ma zostać wywołana, gdy wystąpi zdarzenie**. Zdarzenia to nie tylko kliknięcia myszą, ale także na przykład zmiany danych w elemencie wejściowym.

Zdarzenie **oninput** elementu input jest odpowiedzialne za momenty, w których **dane są wprowadzane** do elementu input, tj. podczas procesu wprowadzania, na przykład, gdy użytkownik wprowadza nowe znaki. **Śledzenie tego zdarzenia** pozwala **dynamicznie reagować** na działania użytkownika.

4. Trochę praktyki

Elementy, zdarzenia i interaktywność w JavaScript

Размер шрифта px. Фоновый цвет

Lorem ipsum dolor sit amet consectetur, adipisicing elit.
Atque, distinctio!

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Nobis accusantium quidem consectetur est tenetur reiciendis ab, voluptates repudiandae temporibus dolor totam fugit odit praesentium! Facere non, repudiandae ratione assumenda dicta quas repellat error itaque quidem ea adipisci veniam accusantium. Assumenda dolorum iure commodi fugiat sapiente praesentium harum? Beatae nam, eligendi minima mollitia, adipisci vitae totam eius cupiditate similique incidunt labore necessitatibus id! Qui, dicta aperiam. Explicabo veritatis nobis consectetur repellat eligendi eius necessitatibus quia corporis provident rem? Accusantium, quam exercitationem facere omnis ex excepturi nemo nulla temporibus vero enim obcaecati. Officia exercitationem dicta odio, necessitatibus illum cumque quas molestiae totam.



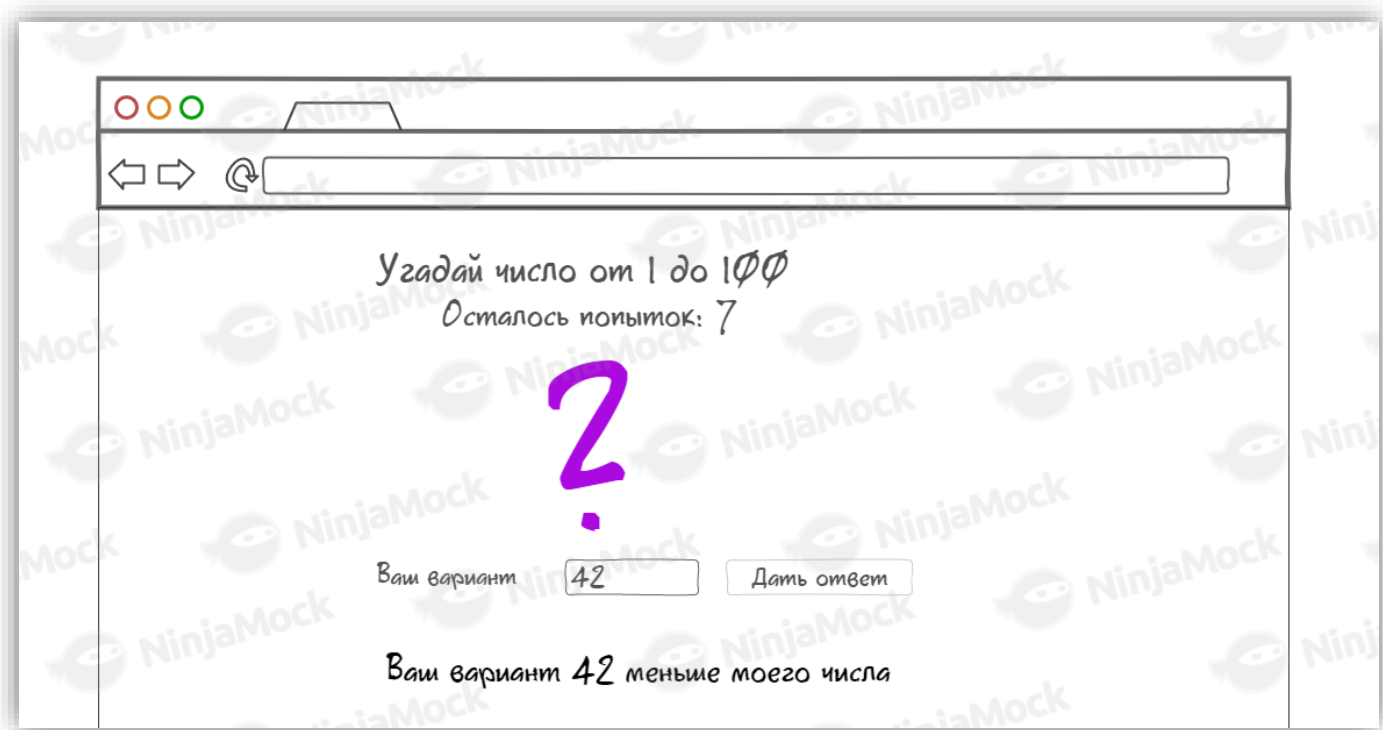
Rem dignissimos voluptas voluptatibus atque tempore explicabo nihil ad deserunt neque aut possimus, sequi dolorem aspernatur nostrum ut! Placeat sed aspernatur libero, officiis optio repellat aperiam at, deserunt repellendus, cupiditate voluptatem corrupti quisquam quo. Distinctio, dignissimos esse? Recusandae earum voluptatibus, consectetur laboriosam aliquid possimus assumenda delectus ullam tempora obcaecati nam doloribus maxime tempore fugiat facere excepturi rerum dolor? Ipsa nesciunt labore sint deleniti error maiores voluptas, expedita ipsam ex sunt impedit, fuga enim, at vero quo incidunt blanditiis id recusandae

Pliki do pobrania

dimon.work/kurs/assets/les23.zip

Gra „Zgadnij liczbę” z formularzem

Skrypt **losuje liczbę** (od 1 do 100 włącznie) i daje graczowi **10 prób** odgadnięcia jej, jeśli użytkownik nie zgadnie - skrypt informuje, że przegrał, a gra mówi mu, jaka była prawidłowa odpowiedź. Jeśli gracz zgadł, gra informuje, że wygrał. Do generowania liczb należy użyć funkcji **Math.random()**. Gra powinna **podpowiedzieć**, czy wprowadzona przez użytkownika liczba jest **większa** lub **mniejsza** od odgadniętej.



Podany przykładowy interfejs możesz dowolnie modyfikować.