JavaScript podstawy – DOM

v 1.3

Coders Lab
SZKOŁA PROGRAMOWANIA

Eventy w DOM-ie

Eventy i funkcja callback

Eventy

- Są to wydarzenia odbywające się na naszej stronie WWW. Dzięki językowi JavaScript jesteśmy w stanie przejąć kontrolę nad eventem i odpowiednio reagować.
- Eventy dzielimy wedle rodzaju interakcji np. użycie myszki czy klawiatury, edycja formularza lub okna przeglądarki itp.
- W obiekcie **event** są zawarte informacje dotyczące danej akcji.

Callback

- Jest to to specjalna funkcja, którą podajemy do wywołania. Nie jest uruchamiana od razu, lecz po wystąpieniu jakiegoś zdarzenia.
- Każdy event w JavaScript jest tworzony za pomocą funkcji callback.

Coders Lab
SZKOŁA PROGRAMOWANIA

Dodawanie eventów do elementów

Eventy dodajemy przez użycie metody **addEventListener(eventName, callback)** na obiekcie elementu.

Zazwyczaj robimy to poprzez użycie anonimowych wyrażeń funkcyjnych (czyli poprzez definicje funkcji w miejscu w którym ją podajemy).

Dzięki temu mamy pewność że nasza funkcja zostanie użyta tylko i wyłącznie w danym miejscu.

Kod HTML

```
<button id="counter">Click me!</button>
```

Kod JavaScript

```
var button = document.querySelector("button");
var clickCount = 0;
button.addEventListener("click", function (event) {
  clickCount += 1;
  console.log("Click number", clickCount);
});
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Dodawanie eventów do elementów

Eventy dodajemy przez użycie metody **addEventListener(eventName, callback)** na obiekcie elementu.

Możemy jednak czasami przekazać normalnie stworzoną funkcję jako callback do eventu.

Kod HTML

```
<button id="counter">Click me!</button>
```

Kod JavaScript

```
var clickCount = 0;
function clickCounter (event) {
  clickCount += 1;
  console.log('Click number', clickCount);
}
var button = document.querySelector("button");
button.addEventListener(clickCounter);
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Dodawanie eventów do elementów

Kod HTML

```
<button id="counter">Click me!</button>
```

Kod JavaScript

```
var button = document.querySelector("button");
var clickCount = 0;
var randomWords = ['Some', 'Random', 'Words'];

function clickCounter (event) {
  clickCount += 1;
  console.log('Click number', clickCount);
}
```

Kod JavaScript – ciąg dalszy

```
function randomWord (event) {
  var myWord = randomWords[Math.floor(Math.random()
                          * randomWords.length)];
  console.log(myWord);
}

button.addEventListener('click', clickCounter);
button.addEventListener('click', randomWord);
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Usuwanie eventów z elementów

- Możemy też usunąć event z elementu. robimy to za pomocą metody: **removeEventListener(event, callback)**.

- Nie da się usunąć eventów, które zostały dodane za pomocą funkcji anonimowych!

Kod HTML

```
<button id="counter">Click me!</button>
```

Kod JavaScript

```
var button = document.querySelector('button');
var clickCount = 0;

function clickCounter (event) {
  console.log('Click number', clickCount);

  clickCount += 1;
  if(clickCount >= 10) {
    event.target.removeEventListener('click',
                                     clickCounter);
  }
}
button.addEventListener('click', clickCounter);
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Lista eventów

mouse:
mousedown, mouseup, click, dblclick,
mousemove, mouseover, mouseout

key:
keydown, keypress, keyup

touch:
touchstart, touchmove, touchend, touchcancel

control:
resize, scroll, focus, blur, change, submit

no arguments:
load, unload, DOMContentLoaded

Pełna lista eventów:
https://en.wikipedia.org/wiki/DOM_events

Coders Lab
SZKOŁA PROGRAMOWANIA

DOMContentLoaded

- DOMContentLoaded jest specjalnym eventem, uruchamiającym się w momencie załadowania całej strony.
- Nasz cały kod JavaScript operujący na DOM powinien znajdować się w tym evencie. Inaczej nie mamy gwarancji, że element którego szukamy, został już stworzony!

- Jeżeli wykonujesz operacje na DOM, upewnij się, że cały dokument został uprzednio załadowany!

```
document.addEventListener("DOMContentLoaded", function () {
        console.log("DOM fully loaded and parsed");
});
```


Coders Lab
SZKOŁA PROGRAMOWANIA

this w eventach

- W każdym evencie mamy możliwość odwołania się do zmiennej **this**.
- Jest to specjalna zmienna reprezentująca element, na którym został wywołany event.
- Jest ona szczególnie przydatna, jeżeli taki sam event nastawiamy na wiele elementów.
- W przykładzie w jednym miejscu zakładamy event na wszystkie guziki.
- **Event ten zmieni kolor tylko tego guzika, w który klikamy, nie wpływa na inne.**

Kod HTML

```
<button class="btn">Click me!</button>
<button class="btn">Click me!</button>
<button class="btn">Click me!</button>
```

Kod JavaScript

```
var buttons = document.querySelectorAll(".btn");

for(var i = 0; i < buttons.length; i++) {
    buttons[i].addEventListener("click", function(event) {
        this.style.backgroundColor = "red";
    });
}
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Obiekt event

Event jest opisywany przez specjalny obiekt. Dzięki niemu możemy dowiedzieć się wielu przydatnych rzeczy na temat zdarzenia. Oto jego przykładowe właściwości:

- **event.currentTarget** – zwraca element, na którym wywołany został event,
- **event.target** – zwraca element, który spowodował wywołanie eventu,
- **event.timeStamp** – zwraca czas, w którym został wywołany event,
- **event.type** – zwraca typ eventu (jako string).

Obiekt Event ma jeszcze kilka przydatnych metod:

- **event.preventDefault()** – anuluj oryginalną akcję,
- **event.stopPropagation()** – anuluj wszystkie eventy tego samego typu z elementów nadrzędnych,
- **event.stopImmediatePropagation()** – anuluj wszystkie eventy tego samego typu przypięte do tego elementu oraz wszystkich elementów nadrzędnych.

Coders Lab
SZKOŁA PROGRAMOWANIA

Propagacja eventów

Event bubbling

W DOM mamy do czynienia z tak zwaną propagacją eventów. Polega ona na przekazywaniu eventu w górę drzewa DOM. Nazywa się to **event bubbling**.

Event capturing

Stare przeglądarki czasami implementowały **event capturing**, czyli przekazywanie eventów w dół drzewa. Jest to jednak metoda przestarzała.

Kod HTML

```
<div id="foo">
  <button id="bar">Click me!</button>
</div>
```

Kod JavaScript

```
document.querySelector('#foo').addEventListener
                                    ('click', function () {
  console.log('Event zarejestrowany, element #foo');
});

document.querySelector('#bar').addEventListener
                                    ('click', function () {
  console.log('Event zarejestrowany, element #bar');
});
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Document Object Model (DOM)

Kod HTML

```html
<div id="foo">
  <button id="bar">Click me!</button>
</div>
```

Kod JavaScript

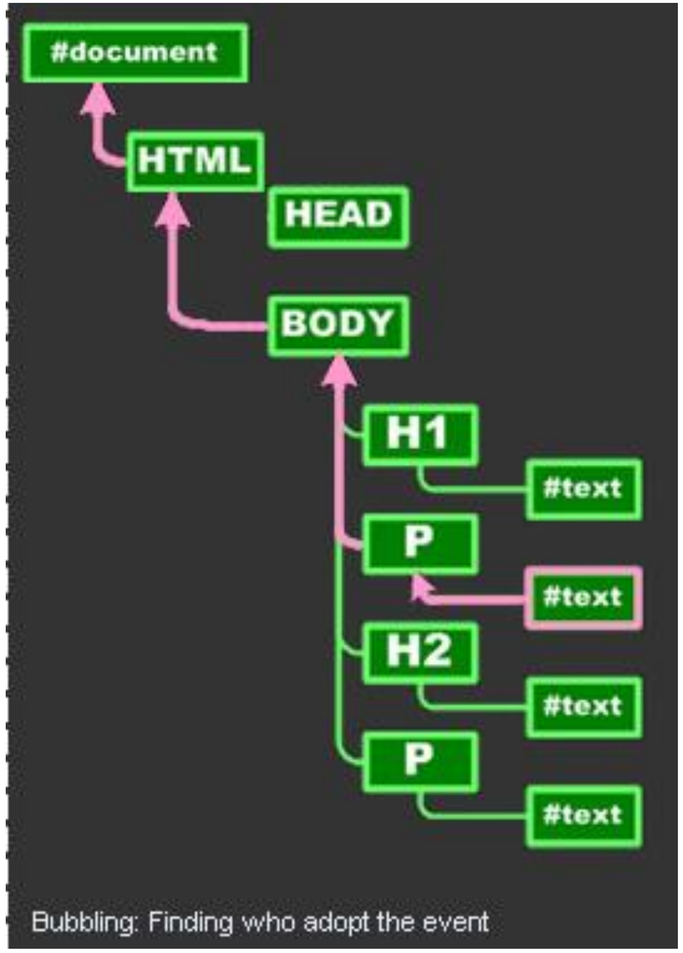
```javascript
document.querySelector('#foo').addEventListener
                                        ('click', function (e) {
  console.log('Target:', e.target.id);
  console.log('CurrentTarget:', e.currentTarget.id);
});

// "Target:" "bar"
// "CurrentTarget" "foo"
```

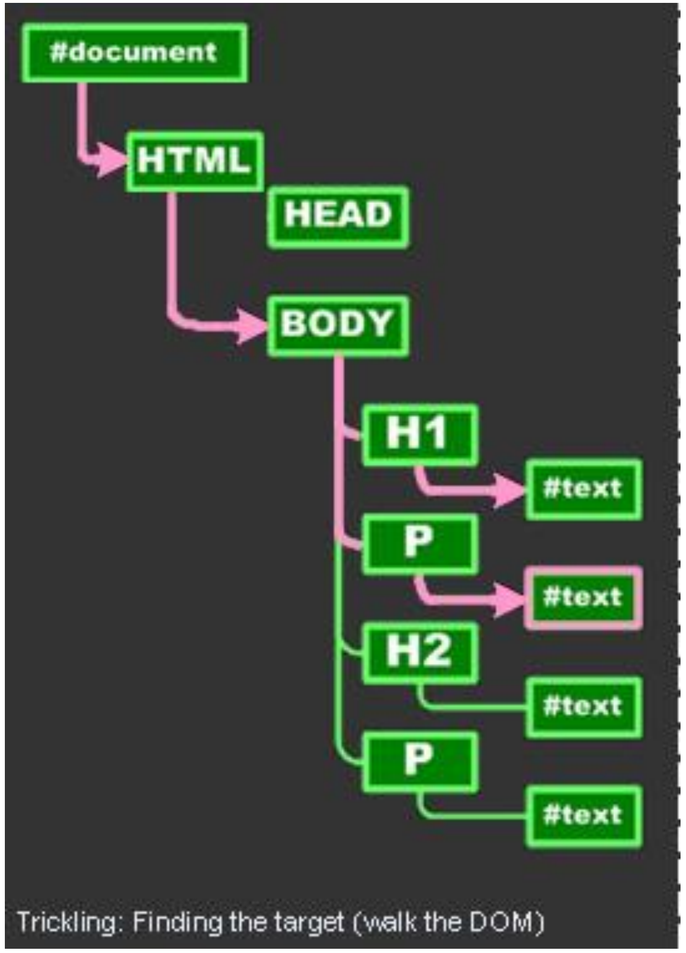

Coders Lab
SZKOŁA PROGRAMOWANIA

Bubbling vs capturing

Event bubbling



Event capturing



Coders Lab
SZKOŁA PROGRAMOWANIA

MouseEvent

Jest to specjalny typ eventu tworzony podczas zdarzeń związanych z myszką. Rozszerza on podstawowy event o następujące atrybuty:

- **event.button** – zwraca przycisk myszki, który został naciśnięty,
- **event.clientX** – zwraca koordynat X (horyzontalny) myszki, relatywnie do górnego, lewego rogu strony,

  event.clientY – zwraca koordynat Y (wertykalny) myszki relatywnie do górnego, lewego rogu strony,
- **event.screenX** – zwraca koordynat X (horyzontalny) myszki, relatywnie do górnego, lewego rogu okna,
- **event.screenY** – zwraca koordynat Y (wertykalny) myszki, relatywnie do górnego, lewego rogu okna.

Coders Lab
SZKOŁA PROGRAMOWANIA

KeyboardEvent

Jest to specjalny typ eventu tworzony podczas zdarzeń związanych z klawiaturą. Rozszerza on podstawowy event o następujące atrybuty:

- **event.altKey** – zwraca **true**, jeżeli **alt** był naciśnięty,
- **event.ctrlKey** – zwraca **true**, jeżeli **ctrl** był naciśnięty
- **event.shiftKey** – zwraca **true**, jeżeli **shift** był naciśnięty.
- **event.charCode** – zwraca znak opisujący klawisz, który wywołał event,
- **event.key** – zwraca wartość klawisza, który wywołał event,
- **event.keyCode** – zwraca wartość Unicode klawisza, który wywołał event,

Pełna lista typów eventów:

http://www.w3schools.com/jsref/dom_obj_event.asp

Coders Lab
SZKOŁA PROGRAMOWANIA

Inne eventy

Pokazane na kursie eventy nie są jedynymi wspieranymi przez DOM. Ich ilość i typy mogą się też zmieniać z biegiem czasu (stare mogą zostać wyrzucone z specyfikacji a na ich miejsce mogą wejść nowe eventy).

Pełna lista typów eventów:

http://www.w3schools.com/jsref/dom_obj_event.asp

Coders Lab
SZKOŁA PROGRAMOWANIA

Czas na zadania

Wykonajcie zadania z działu:

2. DOM

Katalog

4. Eventy

Coders Lab
SZKOŁA PROGRAMOWANIA

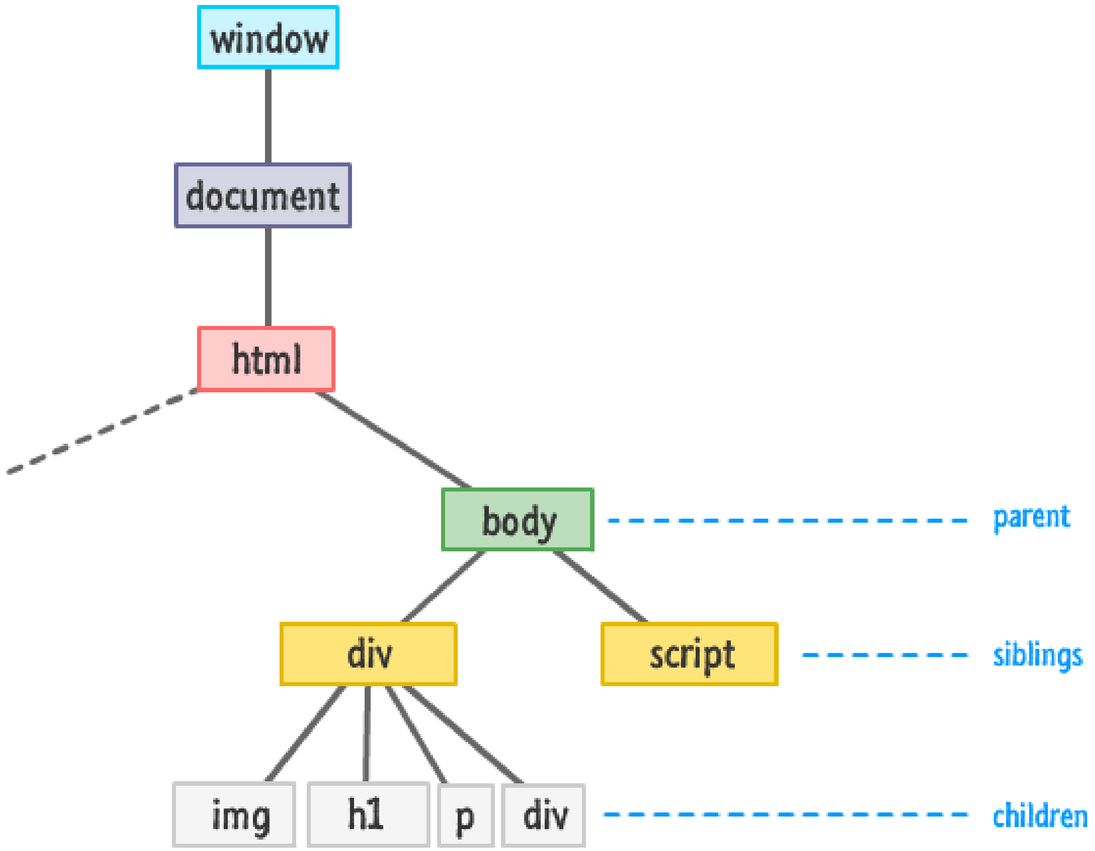
Poruszanie się po drzewie DOM

Poruszanie się po drzewie DOM

Dzięki odpowiednim metodom elementów możemy się swobodnie poruszać po całym dokumencie.

W drzewie rozróżniamy trzy ważne nazwy:

- rodzic (parent),
- brat (sibling),
- dziecko (child).



Coders Lab
SZKOŁA PROGRAMOWANIA

Poruszanie się w górę

- Poruszanie się w górę drzewa jest najłatwiejsze – istnieje tylko jedna ścieżka, którą możemy pójść, a wyznacza ją rodzic (parent) elementu.

- Żeby uzyskać element rodzica, należy użyć atrybutu:
  parentElement

```
<div id="foo">
    <span id="bar">Bar</span>
</div>


var barElement = document.querySelector('#bar');
var barParent = barElement.parentElement;
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Poruszanie się na boki

Mamy dwie możliwości poruszania się na boki w drzewie DOM:

- **el.nextElementSibling** – zwraca następny element mający tego samego rodzica,
- **el.previousElementSibling** – zwraca poprzedni element mający tego samego rodzica.

Funkcje te mogą zwrócić **NULL** w przypadku w którym nie ma poprzedniego / następnego elementu.

```
<div id="foo">
  <span id="bar">Bar</span>
  <span id="baz">Baz</span>
  <span id="buz">Buz</span>
</div>
```

```
var bazElement = document.querySelector('#baz');
var bar = bazElement.previousElementSibling;
var buz = bazElement.nextElementSibling;
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Poruszanie się w dół drzewa

Jeśli poruszamy się w dół drzewa, to mamy do wyboru wszystkie dzieci danego elementu.

Możemy użyć następujących atrybutów:

- **el.children** – zwraca tablicę wszystkich dzieci,
- **el.firstElementChild** – zwraca pierwsze dziecko,
- **el.lastElementChild** – zwraca ostatnie dziecko.

Kod HTML

```
<div id="foo">
  <span id="bar">Bar</span>
  <span id="baz">Baz</span>
  <span id="buz">Buz</span>
</div>
```

Kod JavaScript

```
var fooElement = document.querySelector('#foo');
var allChildren = fooElement.children;
var bar = fooElement.firstElementChild; // lub allChildren[0]
var buz = bazElement.lastElementChild;
                    // lub allChildren[allChildren.length - 1]
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Czas na zadania

Wykonajcie zadania z działu:

2. DOM

Katalog

5. Poruszanie po DOM

Coders Lab
SZKOŁA PROGRAMOWANIA

Tworzenie elementów

Tworzenie elementów

- Potrafimy już wyszukiwać gotowe elementy istniejące na stronie.
- W JavaScript możemy też tworzyć nowe elementy, które na bieżąco dołączamy do strony. Dzięki temu zwiększamy jej interaktywność.

- Elementy możemy tworzyć przez użycie metody **createElement()** na obiekcie **document**.
- Do metody tej przekazujemy napis oznaczający, jakiego typu element chcemy stworzyć.
- Nowo utworzony element najlepiej zapamiętać do zmiennej, żeby potem nim manipulować.

```
var newDiv = document.createElement("div");
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Klonowanie elementów

- Jeżeli mamy już element, na którym chcemy się wzorować (np. ma nastawione odpowiednie klasy, atrybuty), to łatwo możemy go sklonować dzięki metodzie **cloneNode(deep)**.
- Wartość **deep** przyjmująca **true** albo **false** oznacza, czy klonowanie ma być głębokie czy nie.
- Głębokie klonowanie kopiuje i zwraca element wraz z całym poddrzewem czyli wszystkimi potomkami.

```
var toClone = document.querySelector('#foo');
var newDiv = toClone.cloneNode(true);
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Element stworzony a element dodany do DOM

- Stworzenie elementu nie oznacza, że jest dodany do DOM. Możemy go przypisać do zmiennej, pracować na nim, ale nie będzie on dostępny dla użytkownika naszej strony.

- Element stanie się widoczny na stronie dopiero w chwili, w której zostanie on do niej podpięty.

Coders Lab
SZKOŁA PROGRAMOWANIA

Dodawanie elementów do DOM

W celu poprawnego dodania elementu do DOM możemy użyć następujących metod:

- **el.appendChild(nowyElement)** – dodaj element jako ostatnie dziecko danego węzła,
- **el.insertBefore(nowyElement, dziecko)** – dodaj element przed jednym z podanych dzieci,
- **el.replaceChild(nowyElement, dziecko)** – zamień podane dziecko.

```
<div id="foo"></div>

var fooElement =
document.querySelector('#foo');

var newBar = document.createElement("div");
fooElement.appendChild(newBar);

var newBuz = document.createElement("h1");
fooElement.insertBefore(newBuz, newBar);

var newBuz = document.createElement("p");
fooElement.replaceChild(newBuz, newBar);
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Usuwanie elementów z DOM

W celu usunięcia elementu już istniejącego na stronie musimy użyć metody na jego rodzicu:

el.**removeChild(element)**

```
var toDelete = document.querySelector('#bar');

toDelete.parentNode.removeChild(toDelete);
```


Coders Lab
SZKOŁA PROGRAMOWANIA

Czas na zadania

Wykonajcie zadania z działu:

2. DOM

Katalog

6. Tworzenie elementów

Coders Lab
SZKOŁA PROGRAMOWANIA

Inputy i formularze

Elementy typu form

Formularze mają kilka własnych atrybutów, możemy je przypisać tylko do nich. Są to:

- **form.action** – adres URL, do którego prowadzi formularz,
- **form.method** – metoda, którą wysyłany jest formularz,
- **form.elements** – kolekcja elementów należących do tego formularza (w kolejności wpisanej w kodzie HTML).

Formularze mają też specjalne eventy:

- **submit** – jest wywoływany przed wysłaniem formularza. Wysyłanie możemy zablokować przez **preventDefault()** albo zwrócenie **false** z tego eventu,
- **reset** – wywoływane po zresetowaniu formularza (rzadko używane).

Coders Lab
SZKOŁA PROGRAMOWANIA

Input.value

Elementy typu **input** mają kilka atrybutów specjalnych. Jeden z nich jest następujący:

- **input.value** – zwraca wartość, na jaką nastawiony jest **input**. Może służyć też do nastawienia wartości.

Kod HTML

```
<input id="name">
```

Kod JavaScript

```
var input = document.querySelector('#name');
```

```
input.value
```

Zwróci treść wpisaną przez użytkownika

ImięUżytkownika

```
input.value = "Adam"
```

Nastawi wartość Inputa na napis **Adam**

Adam

Coders Lab
SZKOŁA PROGRAMOWANIA

Elementy typu input

- **input.type** – inputy trzymają swój typ. Można go też łatwo zmienić na inny.
- **input.disabled** – zwraca wartość boolenowską, która oznacza, czy element jest włączony czy nie. Możemy ją zmieniać.
- **input.checked** (tylko checkboxy) – zwraca wartość boolenowską, która oznacza, czy element jest zaznaczony czy nie.
- **option.selected** (tylko elementy option) – zwraca wartość boolenowską, która oznacza, czy dany element jest wybrany czy nie.

Elementy typu **input** mają kilka specjalnych eventów, są to:

- **blur** – wywoływany, gdy użytkownik opuści pole,
- **focus** – wywoływany, gdy użytkownik zaznaczy pole,
- **change** – wywoływany, gdy zmieni się wartość pola,
- **keydown, keyup, keypress** – eventy związane z pisaniem na klawiaturze.

Coders Lab
SZKOŁA PROGRAMOWANIA

Czas na zadania

Wykonajcie zadania z działu:

2. DOM

Katalog

7. Inputy i formularze

Coders Lab
SZKOŁA PROGRAMOWANIA