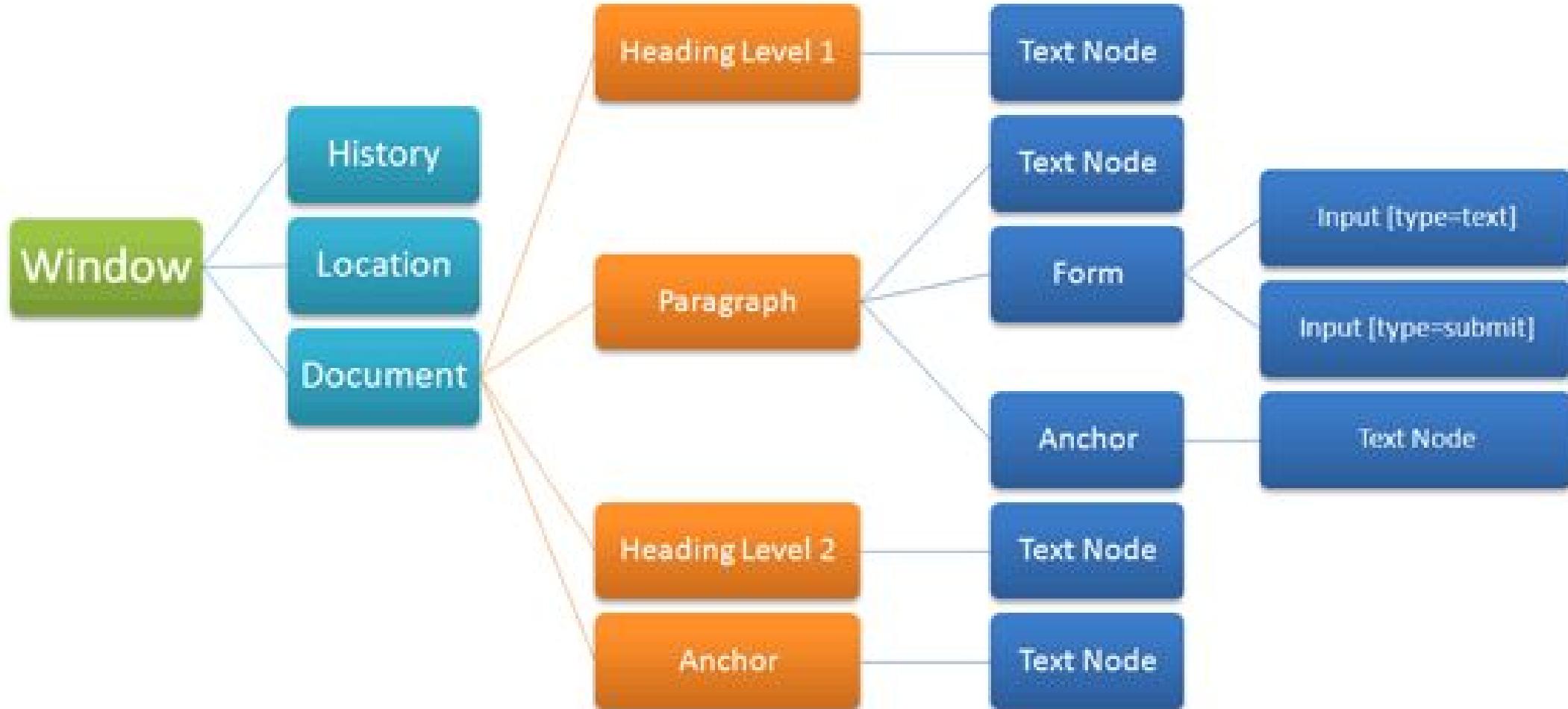




jQuery, DOM

Mateusz Marmołowski
www.ctadventure.pl
www.professor-why.pl
www.edu-sense.pl

DOM - Document Object Model



Poruszanie się po DOM

Poruszanie się po strukturze:

`window.document...`

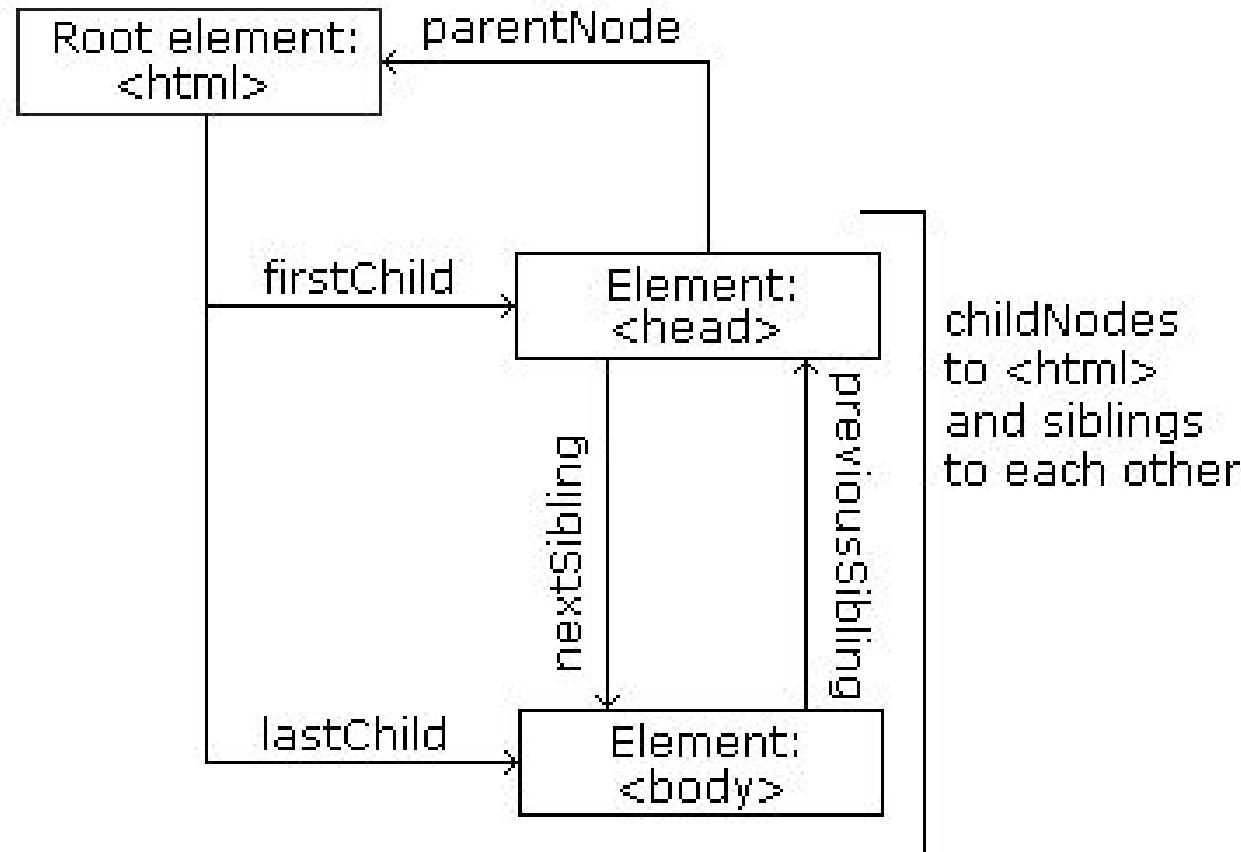
Użycie funkcji

`document.getElementById(...)`

`document.getElementsByName(...)`

`document.getElementsByClassName(...)`

Poruszanie się po DOM



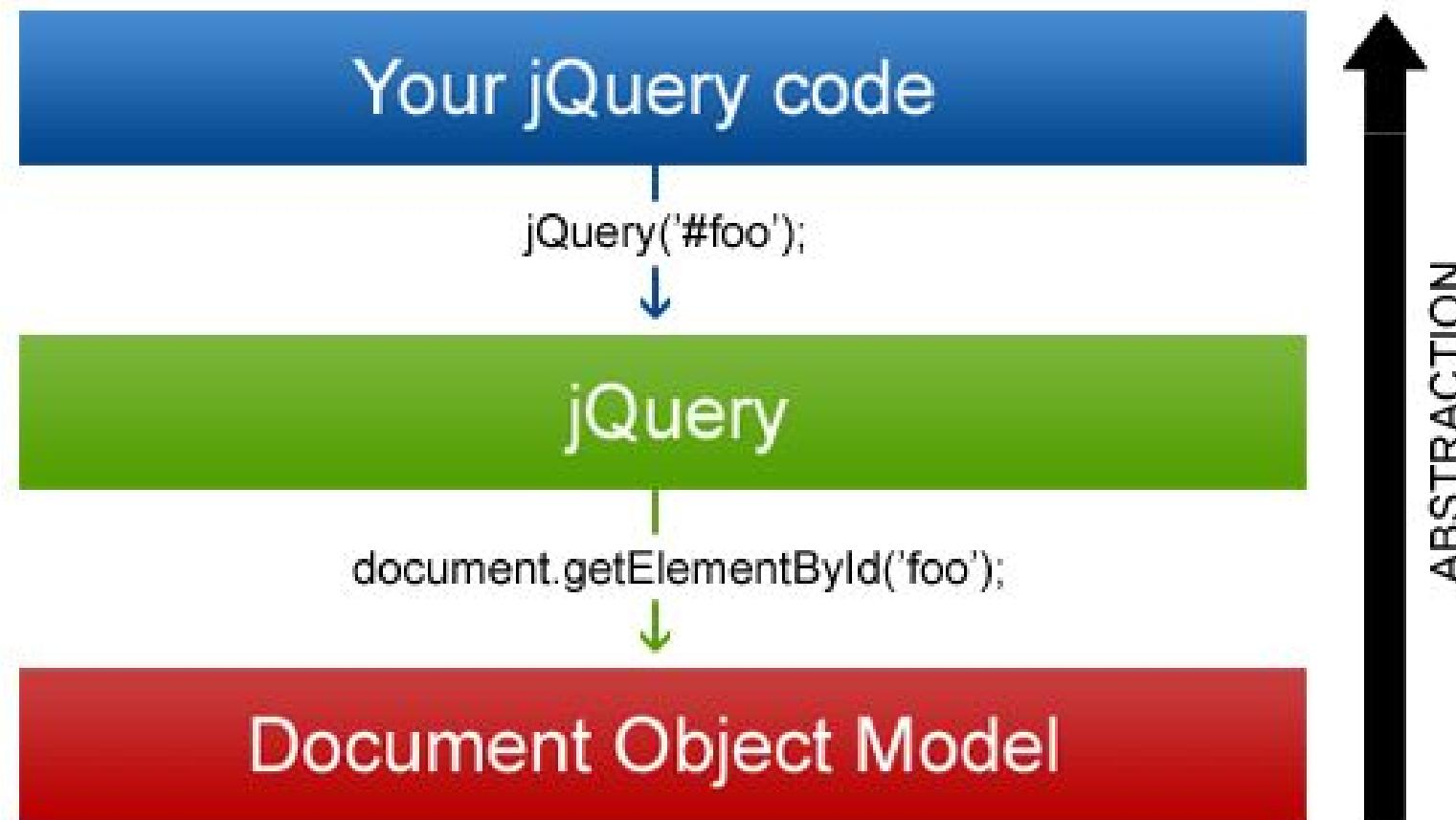
jQuery



- biblioteka JavaScript
- służy do wyszukiwania i modyfikowania elementów
- zapewnia działanie pod wieloma przeglądarkami
- znajduje się w pojedynczym pliku .js

<http://jquery.com/>

jQuery



Selektory jQuery

\$ (selector)

Funkcja jQuery:

\$

Wartość tekstowa definiująca selektor (kompatybilne z CSS):

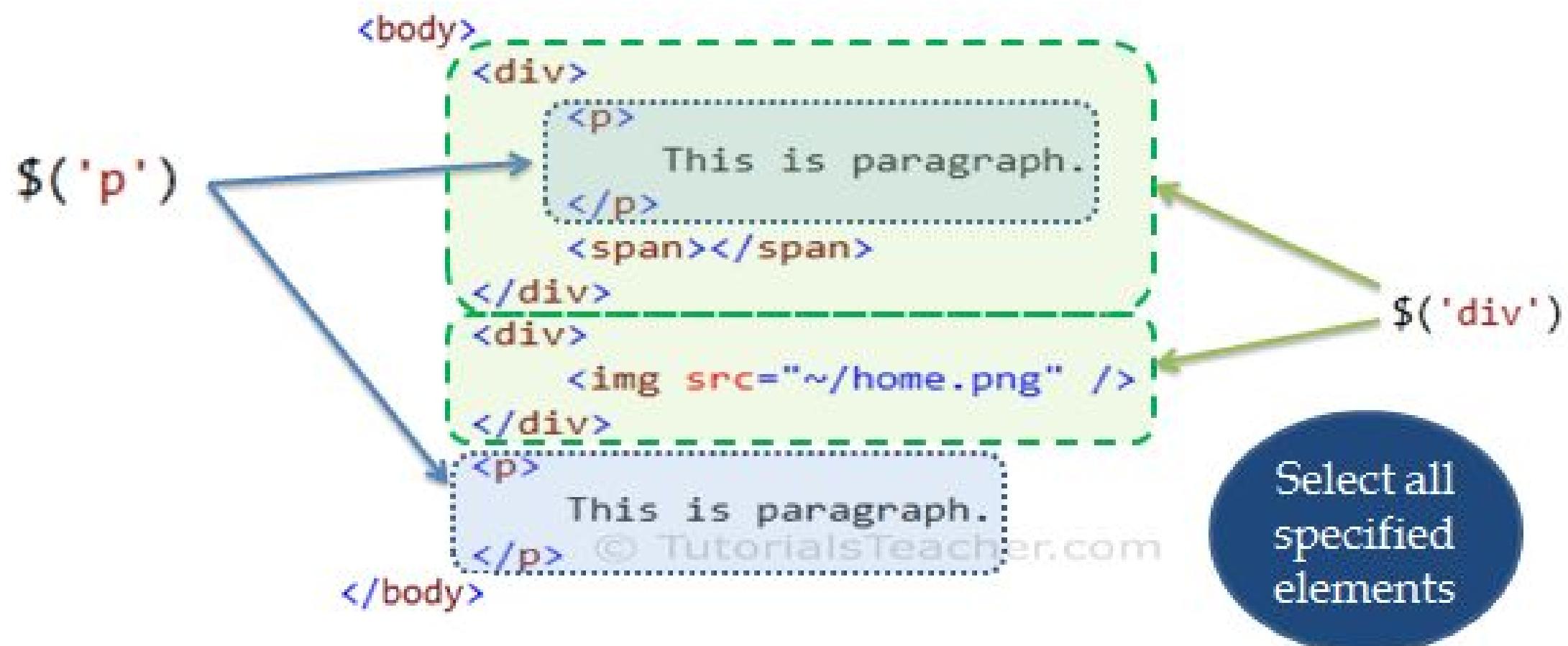
selector

Przykłady selektorów

```
$(`#identyfikator`)  
$('.klasa')  
$('p > a')  
$('p.klasa')  
$('div.news')  
$('h1 a')  
$('input[name=email]')  
$('input.terms:checked')
```

Więcej: <http://api.jquery.com/category/selectors/>

Zapytanie jQuery zwraca kolekcję elementów



Style Guide

Najpopularniejszy styleguide:

<https://github.com/airbnb/javascript>

Świadome użycie cudzysłoiów

```
var str = 'abc';  
<a href="#"></a>
```

Dwa warianty

```
var link = '<a href="#"></a>'  
var link = "<a href=\"#\"></a>"
```

Konstrukcja zapytania jQuery

`$(selector).action()`

Funkcja jQuery:

`$`

Wartość tekstowa definiująca selektor (kompatybilne z CSS):

`selector`

Akcja wykonywana na wszystkich znalezionych elementach:

`action()`

Przykłady zapytań

`$(selector).action()`

Pokazanie wszystkich elementów <p>

`$('p').show();`

Ukrycie wszystkich elementów <p>

`$('p').hide();`

Dobre praktyki

Standardowe zmienne w JavaScript:

```
var zmienna = 'wartosc';
```

Zmienne przechowujące element jQuery:

```
var $element = $('#identyfikator');
```

Oraz:

- nazwy w formacie camelCase
- angielskie nazwy (brak polskich znaków)
- sensowne nazewnictwo

Zadanie 1

1.hide-elements.html

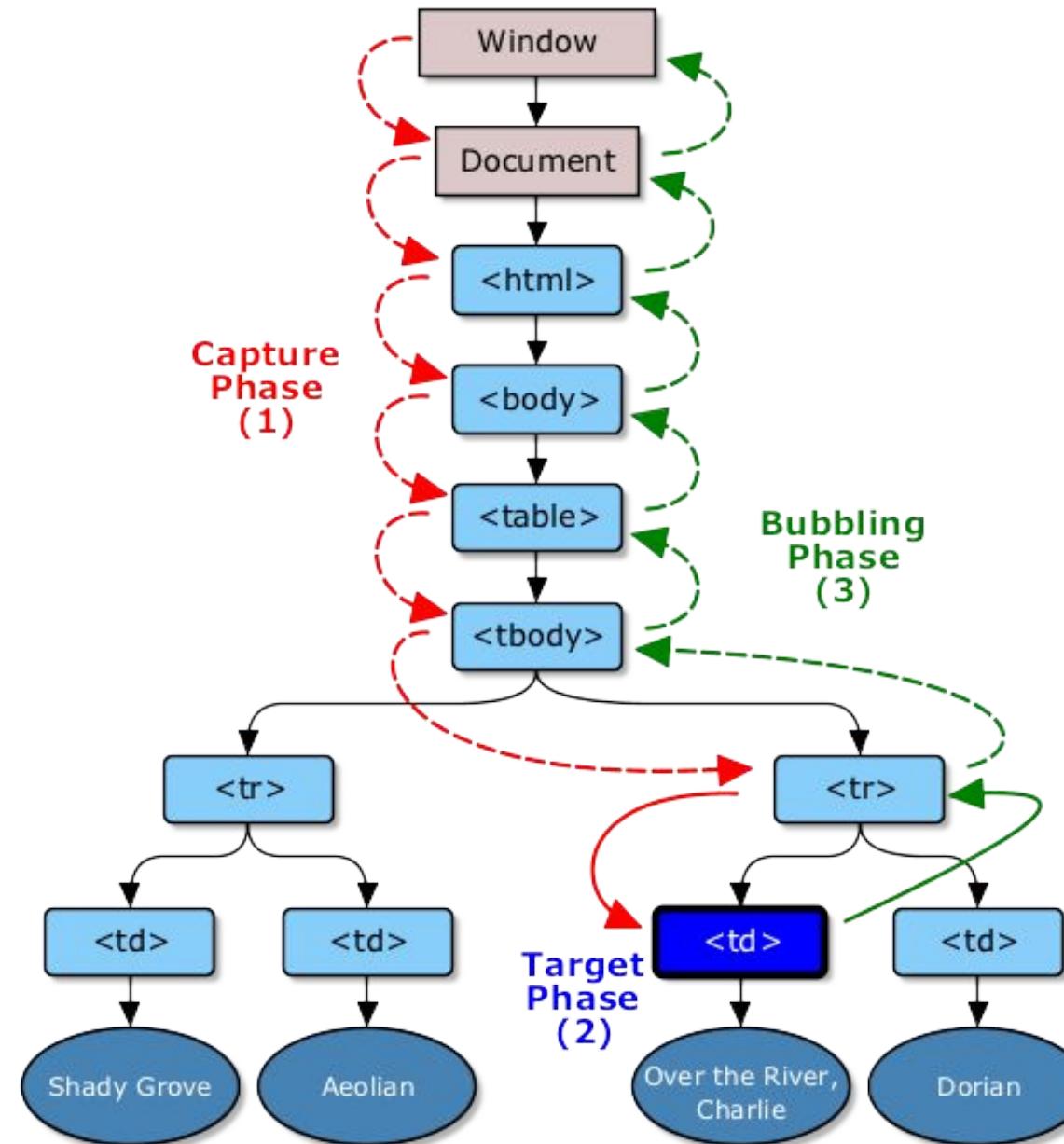
Zdarzenia JavaScript (events)

Zdarzenia przytrafiające się elementom HTML, np:

onclick
onchange
onmousedown
onfocus
itd...

Zdarzenia działają asynchronicznie.

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



Obsługa eventów z użyciem atrybutów HTML

Kliknięcie w kontener:

```
<div onclick="console.log('klik')">Kliknij</div>
```

Wstawienie kurSORA w pole formularza:

```
<input onfocus="funkcjaGlobalna()" />
```

Najechanie myszką na obrazek:

```
<img onmouseover="funkcjaGlobalna()" />
```

Obsługa eventów za pomocą jQuery

```
$(selector).action( ... , callback )
```

Obsługa zdarzenia dla istniejących elementów:

```
$( 'button' ).on( 'click' , function() {} );  
$( 'button' ).click(function() {} );
```

Bindowanie zdarzenia (działa dla dynamicznie dodawanych elem.):

```
$( 'body' ).on( 'click' , 'button' , function() {} );
```

<http://api.jquery.com/category/events/>

Funkcja callback - anonimowa

```
\$(selector).action( callback )
```

Dla przykładu:

```
\$('button').click(function() {  
    \$(this).hide();  
});
```

Funkcja callback - nazwana

```
\$(selector).action( callback )
```

Definicja funkcji nazwanej:

```
function nazwaFunkcji() { }
```

Użycie jako callback (handler):

```
\$('button').click(nazwaFunkcji);
```

Zadanie 2

2.events.html

Pokazywanie i ukrywanie elementów

```
$( '#id' ).hide()  
$( '#id' ).show()
```

```
$( '#id' ).fadeIn(500)  
$( '#id' ).fadeOut(500)  
$( '#id' ).fadeToggle(500)
```

```
$( '#id' ).slideUp(500)  
$( '#id' ).slideDown(500)  
$( '#id' ).slideToggle(500)
```

Dodawanie i usuwanie klas CSS

Sprawdza czy element posiada klasę o nazwie “*klikalny*”:

```
$( '#id' ).hasClass( 'klikalny' )
```

Dodaje klasę do elementów:

```
$( '#id' ).addClass( 'klikalny' )
```

Usuwa klasę:

```
$( '#id' ).removeClass( 'klikalny' )
```

Zamiennie dodaje i usuwa klasę:

```
$( '#id' ).toggleClass( 'klikalny' )
```

Zadanie 3

3.css-classes.html

Czekamy na załadowanie całego dokumentu

Zdarzenie ready na dokumencie:

```
$ (document).ready( function() {  
    console.log( "ready!" );  
} );
```

Skrócona wersja:

```
$ (function() {  
    console.log( "ready!" );  
} );
```

Kolejkowanie zapytań

Zamiast:

```
$( '#id' ).slideUp(2000)  
$( '#id' ).slideDown(2000);
```

Stosujemy:

```
$( '#id' ).slideUp(2000).slideDown(2000);
```

Zadanie 4

4.hazard.html

Tworzenie i usuwanie elementów

Nowy element gotowy do wstawienia do dokumentu:

```
$(`<a href="#">link</a>`)
```

Czyszczenie zawartości elementu:

```
$('#id').empty();
```

Usunięcie elementu z zawartością:

```
$('#id').remove();
```

Dołączanie elementów do drzewa DOM

Utworzenie nowego elementu:

```
var $element = $('link');
```

Dołączenie do struktury DOM, jako ostatni element selektora:

```
$('#id').append($element);
```

Umieszczenie elementu jako pierwszy wewnątrz selektora:

```
$('#id').prepend($element);
```

Uwaga! Pamiętaj, aby wstawiać nowe elementy do struktury DOM.

Modyfikacja atrybutów HTML

Odczytanie wartości atrybutu:

```
$( '#idLink' ).attr( 'href' )
```

Dodanie i zmiana wartości atrybutu:

```
$( '#idLink' ).attr( 'href' , 'http://...' );
```

Usunięcie atrybutu:

```
$( '#idLink' ).removeAttr( 'href' );
```

Manipulowanie treścią

```
<a href="#"><b>link</b></a>  
$( 'a' ).text()  
=> wynik to 'link'
```

Nadanie zawartości tekstowej:

```
$( 'a' ).text( 'nowy' )  
=> wynik to <a href="#">nowy</a>
```

```
$( 'a' ).text( '<i>nowy</i>' )  
=> wynik to <a href="#">&lt;i&gt;nowy&lt;/i&gt;</a>
```

Manipulowanie zawartością HTML

```
<a href="#"><b>link</b></a>  
$( 'a' ).html()  
=> wynik to '<b>link</b>'
```

Nadpisanie zawartości tekstowej:

```
$( 'a' ).html( 'nowy' )  
=> wynik to <a href="#">nowy</a>
```

```
$( 'a' ).html( '<i>nowy</i>' )  
=> wynik to <a href="#"><i>nowy</i></a>
```

Manipulowanie zawartością formularza

Odczyt wartości pola formularza:

```
$ ('<input id="email" value="test@gmail.com"/>');
$( 'input' ).val()
=> wynik to 'test@gmail.com'
```

Zmiana wartości pola formularza:

```
$( 'input' ).val('nowy@wp.pl')
=> wynik to <input id="email" value="nowy@wp.pl" />
```

Zadanie 5

5.form.html

Trawersowanie dokumentu

```
$(`#id`).children()  
$(`#id`).parent()  
$(`#id`).siblings()  
$(`#id`).next()  
$(`#id`).prev()
```

<https://api.jquery.com/category/traversing/>

Przeszukiwanie wyników

Przeszukiwanie potomków:

```
$( '#id' ).find( 'li' );
```

Filtrowanie wyników:

```
$( '#id' ).filter( ':even' );
```

Kontekstowy selektor

```
$( '#id' , '#section' )
```

<https://api.jquery.com/category/traversing/>

Przetwarzanie wyników

Iteracyjnie po wszystkich elementach:

```
$( "li" ).each(function(index) {  
    if (index%2==0) console.log($(this).text());  
});
```

Mapowanie elementów:

```
var evenItems = $( "li" ).map(function(index) {  
    if (index%2==0) return $(this).text();  
});  
console.log(evenItems);
```

Zadanie 6

6.menu.html

Definiowanie tablicy

Pusta tablica:

```
var myTable = [ ] ;
```

Liczbową:

```
var liczby = [ 10 , 11, 12 ] ;
```

Tekstową:

```
var cars = [ 'Saab' , 'Volvo' , 'BMW' ] ;
```

Odczyt wartości

```
var cars = ['Saab', 'Volvo', 'BMW'];
```

Pobranie pierwszego elementu:

```
cars[0]
```

Pobranie drugiego elementu:

```
cars[1]
```

Liczba elementów tablicy (właściwość):

```
cars.length
```

Zapis wartości

```
var cars = ['Saab', 'Volvo', 'BMW'];
```

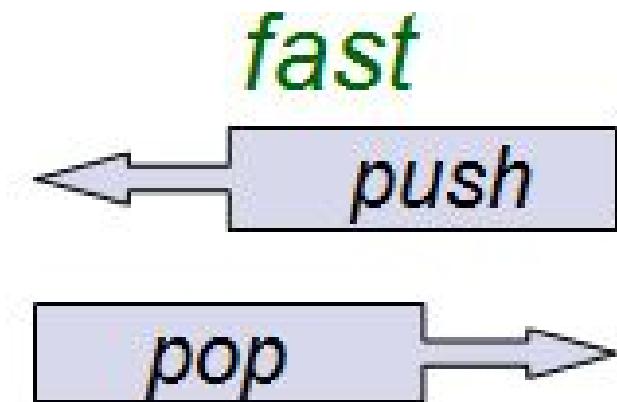
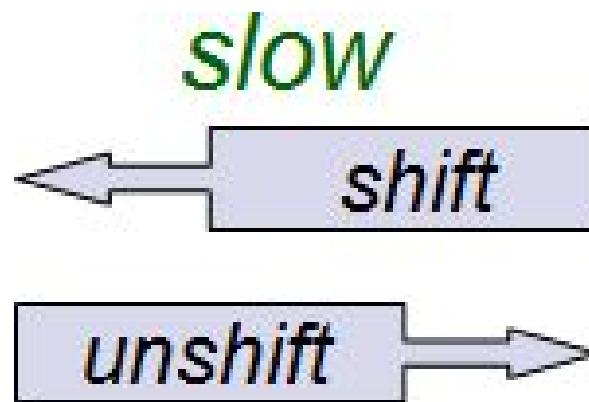
Napisanie pierwszego elementu:

```
cars[0] = 'Daewoo';
```

Dopisanie nowego, czwartego elementu:

```
cars[3] = 'Mercedes';
```

Dokładanie elementów



Sklejanie tablic

```
var a = [ "Batman", "Robin" ];  
var b = [ "Freeze", "Riddler" ];  
var c = a.concat(b)
```

Wynik:

```
a == [ "Batman", "Robin" ];  
b == [ "Freeze", "Riddler" ];  
c == [ "Batman", "Robin", "Freeze", "Riddler" ];
```

Iterowanie po elementach tablicy

```
var heroes = ['Batman', 'Robin', 'Gordon'];
```

Pętla for:

```
for (index=0; index<heroes.length; ++index)
    console.log(heroes[index], index);
```

Funkcja .forEach()

```
heroes.forEach(function(hero, index) {
    console.log(hero, index);
});
```

Mapowanie tablicy

Tablica wejściowa pozostaje bez zmian:

```
var heroes = ['Batman', 'Robin', 'Gordon'];
```

.map tworzy nową tablicę złożoną ze zwracanych wartości

```
var modifiedHeroes = heroes.map(function(hero) {  
    return 'Hero - ' + hero;  
});
```

Kolejkowanie funkcji (chaining)

```
var heroes = ['Batman', 'Robin', 'Gordon'];
```

.map tworzy nową tablicę złożoną ze zwracanych wartości
heroes

```
.map(function (hero) {  
    return 'Hero - ' + hero;  
})  
.forEach(function(hero) {  
    console.log(hero);  
});
```

Zadanie 7

7.array.html

Obiekty

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

Zapis i odczyt wartości

```
var car = {  
    name : 'Fiat' ,  
    model : 500 ,  
    color : 'white'  
} ;
```

Odczyt wartości obiektu:

```
car.name      // 'Fiat'  
car.model     // 500  
car['color']  // 'white'
```

Metody obiektu

```
var car = {  
    name : 'Fiat' ,  
    model : 500 ,  
    start : function() { console.log('bruum') } ,  
    fullModel : function() { return this.name +  
        ' ' + this.model; }  
};
```

```
car.start();      // undefined  
car.fullModel() // 'Fiat 500'
```

Zadanie 8

8.object.html

Animacje w jQuery

```
$ (selector).animate(  
    {  
        parametr_1_css: wartość_docelowa_1 ,  
        parametr_2_css: wartość_docelowa_2 ,  
        parametr_3_css: wartość_docelowa_3  
    },  
    czasTrwaniaWMilisekundach  
) ;
```

<http://api.jquery.com/animate/>

Zadanie 9

9.animate.html

Ćwiczenia

Katalog **10 . samples** zawiera ćwiczenia utrwalające jQuery.

Plugins

<https://plugins.jquery.com>

KOANS

Katalog `11.project` zawiera 15 zadań i w każdym z nich:

- modyfikujemy plik `js/app.js`
- testujemy postępy za pomocą `test.html`
- rezultat naszej pracy podglądamy za pomocą `index.html`



Dziękuję za uwagę

Mateusz Marmołowski
www.ctadventure.pl
www.professor-why.pl
www.edu-sense.pl