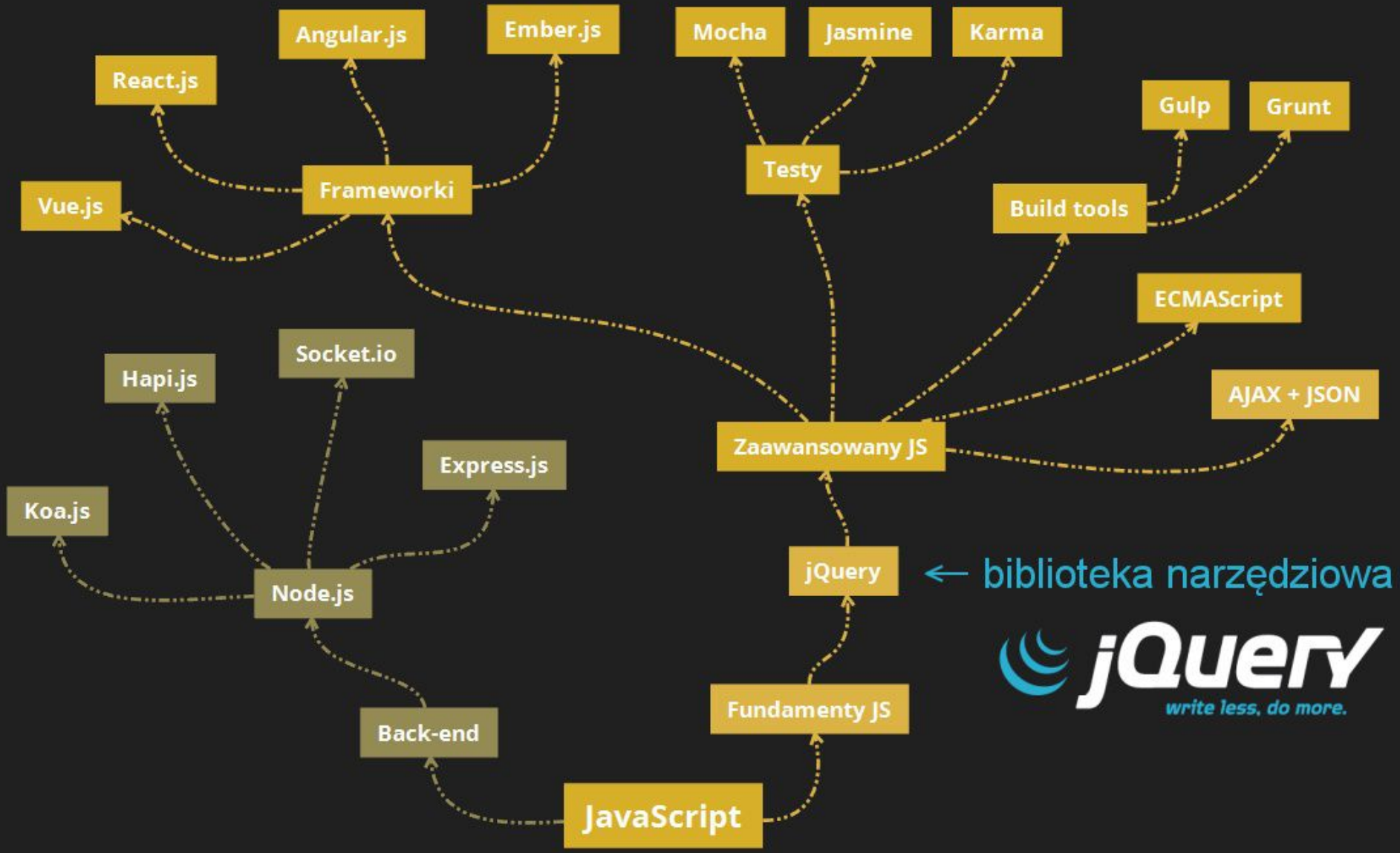
Angular.js
Ember.js
Mocha
Jasmine
Karma
React.js
Gulp
Grunt
Testy
Frameworki
Vue.js
Build tools
ECMAScript
Socket.io
Hapi.js
AJAX + JSON
Zaawansowany JS
Express.js
Koa.js
jQuery
← biblioteka narzędziowa
Node.js
Fundamenty JS
Back-end
JavaScript

Tworzenie uchwytów elementów w jQuery

Programista

Zazwyczaj tworzyłem je następująco:

js `document.getElementById('pojemnik')`

html `<div id="pojemnik"></div>`

Ło panie, to może stwórzmy je inaczej!

$( )

Codziennie przecież "wyjmujemy" elementy
ze strony, kiedy chcemy je ostylować w CSS!

`#pojemnik`	`.kafelek`	`h1`	`:hover`	`[target="_blank"]`
identyfikator	klasa	znacznik	pseudoklasa	atrybut o zadanej wartości

Tworzenie uchwytów elementów w jQuery

Programista

Zazwyczaj tworzyłem je następująco:

js `document.getElementById('pojemnik')`

html `<div id="pojemnik"></div>`

Używajmy zatem w jQuery selektorów CSS!

Zobaczmy kilka przykładów:

`$('#container')` `$('.kafelek')`

`$('input')` `$('[alt="Kocham bekon!"]')`

$( )

Co możemy wysłać do funkcji globalnej?

Programista

selektor CSS

```
$('#pojemnik')
```

obiekt, np. document

```
$(document)
```

kod HTML

```
$('<br>')
```

inna funkcja

```
$(function(){ })
```

$( )

Co możemy wysłać do funkcji globalnej?

selektory CSS

kilka przykładów:

Programista

```
$('#container')
```

identyfikator

```
$('.kafelek')
```

klasa

```
$('input')
```

znacznik

```
$('[alt="Kocham bekon!"]')
```

atrybut o zadanej wartości

$( )

jQuery

write less, do more.

query = zapytanie, kwerenda

Panie, daj mnie Pan teraz wszystkie elementy DOM o klasie "pomidor"

zapytanie

```
$('.pomidor').metoda(parametry);
```

uchwyt, który pozwala użyć metody z interfejsu biblioteki

Programista

$( )

OK, wiemy już co możemy do tej funkcji wysłać, jednak nie wiemy, co funkcja zwraca w zamian?

Otóż wartość, którą zwraca funkcja globalna nazywamy obiektem jQuery.

Co ważne: obiekt jQuery może reprezentować nie tylko jeden element, ale cały ich zbiór (kolekcję):

```
// Obiekt jQuery zawierający pojedynczy element
var obiekt1 = $('#container');
```

```
// Obiekt jQuery zawierający zbiór elementów
var obiekt2 = $('p.text');
```

Wiele osób nowych w jQuery mylnie zakłada, że zwrócony obiekt jQuery (zawierający zbiór elementów) to tablica – owszem, można odnieść takie wrażenie, jako że możemy się m.in. posługiwać nawiasami [ ] do wyciągania poszczególnych elementów kolekcji, jak również istnieje właściwość length takiego zbioru.

```
// Wszystkie divy na stronie "opakowane" w obiekt jQuery
var pojemniki = $('div');
// Atrybut length podaje ilość elementów w kolekcji
var ile_divow = pojemniki.length;
```

```
// Jeśli chcemy się dostać do pierwszego diva na stronie
// opakuj go w obiekt jQuery
var pierwszy_div = $('div').eq(0); pierwszy_div.html("Mamy dostęp!");
// albo zwróć bezpośrednio jako element DOM
var pierwszy_div = $('div')[0]; pierwszy_div.innerHTML = "Mamy dostęp!";
```

Jednak obiekt jQuery to dużo więcej niż tablica!

Jeśli do funkcji globalnej przekażemy obiekt, na przykład document (albo window, element) to po wywołaniu funkcja opakowuje dokument (okno, element) w obiekt jQuery i jako taki go do nas zwróci.

To opakowanie w obiekt jQuery pozwala użyć na nim metod jQuery, zamiast czystych metod DOM. Metoda jQuery to dostarczona w bibliotece gotowa funkcja, która może zostać wywołana na rzecz obiektu jQuery zwróconego przez funkcję globalną.

```
obiekt_jQuery.metoda_jQuery(parametry);
```

```
$('#pojemnik').html("Nowa zawartość diva");
```

Zyskujemy więc cały interfejs nowych, przetestowanych, odporniejszych na błędy metod, gotowych do wykorzystania!

A nasze manipulowanie żywym drzewem DOM zostaje odsunięte (wyabstrahowane, wyjęte) do abstrakcyjnej przestrzeni udostępnionych klas, obiektów i metod.

Dzięki temu interakcja z hierarchią DOM pozbawiona jest licznych tarć i problemów (na przykład łagodzona jest niekompatybilność pomiędzy różnymi przeglądarkami).

element.addEventListener( )

pozwala dodać ("przypiąć") do elementu dowolną ilość "nasłuchiwaczy" danego zdarzenia (w przeciwieństwie do sposobu z atrybutami - wówczas jesteśmy ograniczeni do jednej funkcji obsługującej zdarzenie)

w IE < 9 "nasłuchiwaczem" zdarzeń jest metoda attachEvent( ), stąd addEventListener( ) nie zadziała w tych przeglądarkach - no chyba, że ręcznie okodujemy wykrywanie tej przeglądarki w naszym kodzie, poparte wywoływaniem w przypadku IE < 9 attachEvent( ) zamiast addEventListener( )

lepiej oddzielamy kod JavaScript od HTML - następuje lepsza separacja pomiędzy mechaniką a wyglądem strony (brak atrybutu onclick w HTML)

kontrola nad aktywacją nasłuchiwania (przechwytywanie a propagowanie, zyskujemy dodatkową, opcjonalną, możliwą do użycia flagę useCapture)

element.addEventListener( )

Sposób wywołania metody:

```
element.addEventListener('click', function() {  }, false);
```

Trzeci, opcjonalny argument to flaga useCapture - określa, który sposób zajścia zdarzenia "usłyszy" nasz listener. Przestawienie na true spowoduje, iż obsługa zdarzenia zostanie uruchomiona najpierw dla obiektów nadrzędnych do naszego w hierarchii DOM. Ustawienie false najpierw uruchamia obsługę zdarzenia dla naszego obiektu, a dopiero potem dla obiektów nadrzędnych w drzewie DOM.

element.addEventListener( )

Alternatywne wywołanie metody:

```
var mojaFunkcja = function() {  }
element.addEventListener('click', mojaFunkcja, false);
```

Jednak w IE < 9 musimy zapisać:

```
element.attachEvent('onclick', mojaFunkcja);
```

Z użyciem jQuery byłoby tak:

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

Biblioteka łagodzi dla nas niekompatybilność w IE < 9, jako że sama sprawdzi przeglądarkę z którą ma do czynienia - w przypadku IE < 9 automatycznie zostanie użyta działająca metoda attachEvent( )

addClass(klasa)

dodaj ("przypnij") klasę do elementu, ale uwaga: dodajemy kolejną,
ale nie zastępujemy innych, już aktywnych ("przypiętych") klas

removeClass(klasa)

usuń podaną klasę z obiektu, czyli "odepnij" ją od elementu

toggleClass(klasa)

jeśli element posiada "przypiętą" klasę, to ją usuwamy z elementu,
zaś w przeciwnym wypadku (jeśli element nie posiada "przypiętej" klasy)
to wówczas dodaj klasę do tego obiektu (ponownie: "dodaj", a nie "zastąp")