JavaScript

Wstęp do podstaw elementów

dr inż. Paweł Boiński, Politechnika Poznańska

Wprowadzenie

- Historia
  - Maj 1995 -10 dni, Mocha, Netscape
  - Wrzesień 1995 - Mocha -> LiveScript
  - Grudzień 1995 - LiveScript - > JavaScript
  - Konkurencja: JScript (Microsoft), ActionScript (Adobe)
  - 1996-1997 - przekazany do ECMA (European Computer Manufacturers Association) celem standaryzacji
    - Standard ECMAScript (ES6 2015 r.)
    - JavaScript, ActionScript – implementacja ECMAScript
- Zastosowania
  - najczęściej uruchamiany po stronie klienta (w przeglądarce)
  - może być uruchamiany po stronie serwera (node.js)
  - można także pisać „zwykłe” aplikacje

Ogólne uwagi

- Wielkość liter **ma znaczenie**
- Automatyczny hoisting
- Specyficzny zasięg widoczności zmiennych
- Kontekst
  - `window` – globalny
  - `document` – dokumentu

Gdzie umieścić kod JavaScript?

- head

```
<html>
  <head>
    <script>
    //…
    </script>
  </head>
  <body>
  </body>
</html>
```

- body

```
<html>
  <head>
    <script language="javascript" type="text/javascript">
     //…
    </script>
  </head>
  <body>
  </body>
</html>
```

przestarzałe

wartość domyślna

- zewnętrzny plik

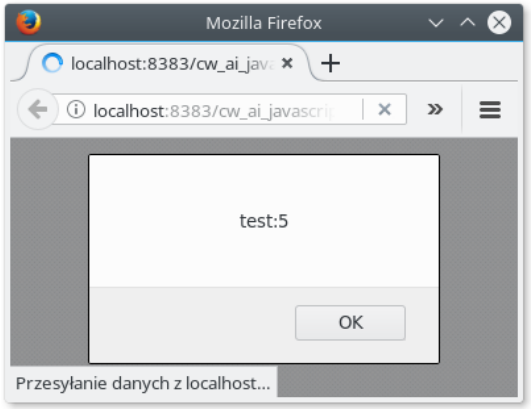
```
<html>
  <head>
    <script src="plik.js"></script>
  </head>
  <body>
  </body>
</html>
```

zalecane

Wyjście (1)

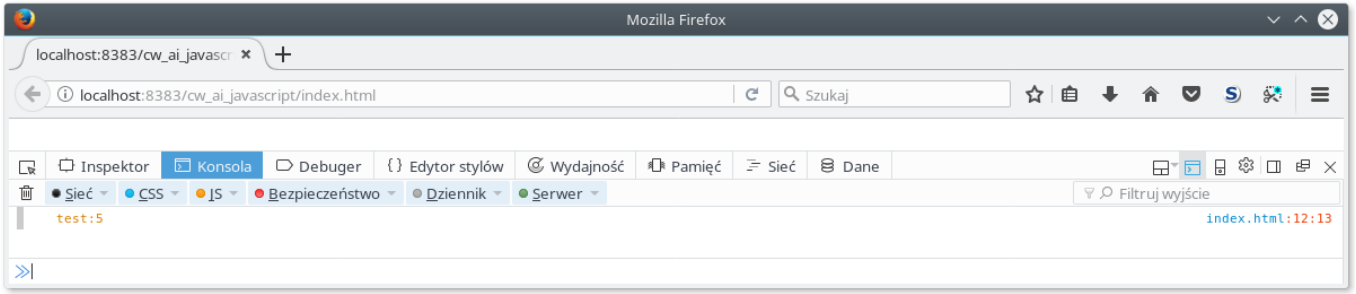
- `window.alert(tekst)`

```
var x = 5;
alert("test:" + x);
```



- `window.console.log(obiekt)/console.dir(obiekt)`

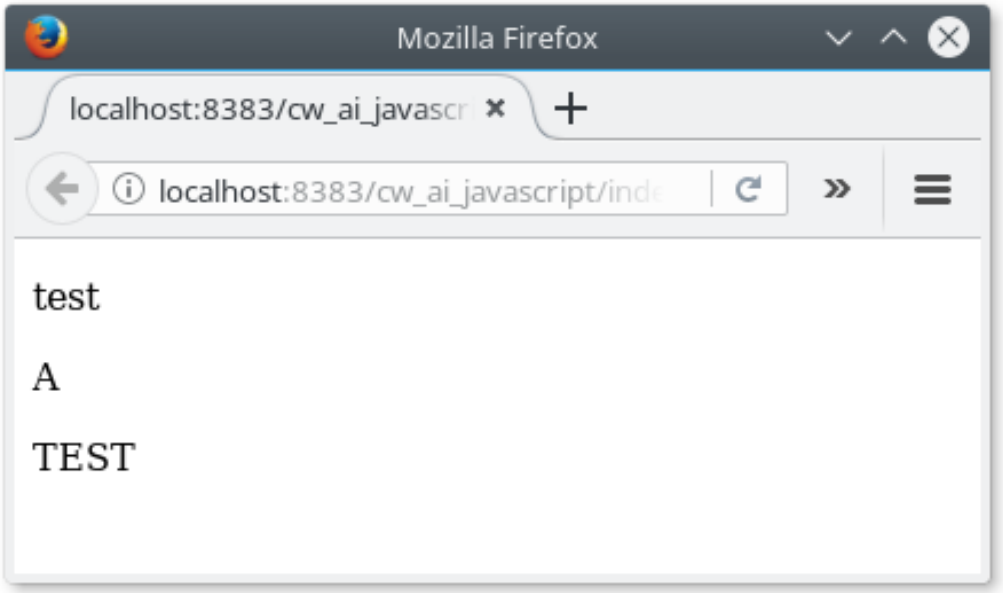
```
var x = 5;
console.log("test:" + x);
```



Wyjście (2)

- `window.document.write(tekst)`

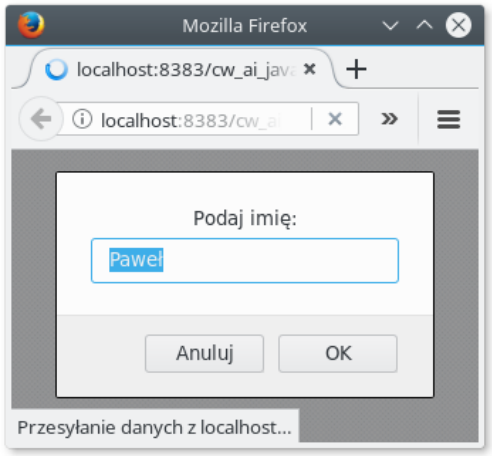
```
<html>
  <head></head>
  <body>
    <p>test</p>
    <script>
      document.write("A");
    </script>
     <p>TEST</p>
  </body>
</html>
```



Wejście

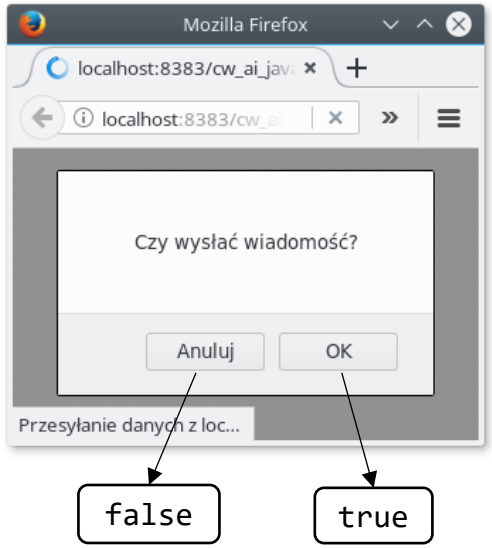
- `window.prompt(nagłówek, domyślnie))`
  - Pierwszy parametr to tzw. nagłówek
  - Drugi parametr (opcjonalny) to wartość tekstowa pola

```
var imie = prompt("Podaj imię:", "Paweł");
```



- `window.confirm(pytanie)`
  - Zwraca `true` jeżeli naciśnięto OK,
  - Zwraca `false` jeżeli naciśnięto Anuluj

```
var odp = confirm("Czy wysłać wiadomość?");
```



Komentarze, operatory

- Komentarz
  - Blokowy

```
/* To jest
Komentarz */
```

  - Liniowy

```
var a = 5; // to jest komentarz
```

- Operatory porównania
  - porównanie wartości == (np. `true` dla `1=="1"`)
  - porównanie wartości i typu === (np. `false` dla `1==="1"`)
- Operatory logiczne
  - && (AND)
  - || (OR)
  - ! (NOT)

Zmienne typowane dynamicznie

- Ta sama zmienna może raz przechowywać liczby, a innym razem łańcuchy znaków
- Skąd wiedzieć jaki jest aktualny typ zmiennej? **duck typing**

  Jeżeli coś wygląda jak kaczka, pływa jak kaczka i kwacze jak kaczka to prawdopodobnie jest kaczką

  Sprawdzamy na etapie wykonania, czy obiekt ma odpowiednie metody

Zmienne, funkcje

- Zmienne

```
var x = 10; // nie ma "wymuszonego" typu
```

- Funkcje – metody deklaracji

```
function isOK1(param1) {
  // instrukcje
  // return true; - opcjonalne
}
var isOK2 = function(param1, param2) {
  // instrukcje
}
```

- Wywołanie funkcji – różne parametry lub ich brak

```
isOK1(100);
isOK1(); // param1 w funkcji ma wartość niezdefiniowaną
isOK1("aa");
```

Typy danych

- Zmienna bez przypisania „ma wartość” undefined

```
var a;
console.log(a);
// undefined
```

- Liczby - przechowywane jako double precision float

```
var a = 5;
var A = 5.5; // uwaga: wielkość liter ma znaczenie
```

- boolean – tylko wartości true/false

Instrukcje sterujące

```
if (warunek) {
  // instrukcje
} else { // opcjonalne
  // instrukcje
}
```

```
switch (warunek) {
  case wartosc1:
    // instrukcje
    break;
  case wartosc1:
    // instrukcje
    break;
  //…
  default:
    // instukcje
    break; // opcjonalne
}
```

```
while (warunek) {
  // instrukcje
}
```

```
do {
  // instrukcje
} while (warunek);
```

```
for (instrukcje początkowe; warunek; instrukcje
                                     kroku) {
  // instrukcje
}
```

Dopuszczalne wiele instrukcji początkowych

```
for (własność in obiekt) {
  // instrukcje
}
```

Kolejność nie jest zdefiniowana

Tablice (1)

```
var samochody = ["AUDI", "BMW"];
samochody[3] = "VW";
for (s in samochody) {
   console.log(s);
} // 0 1 3
for (s of samochody) {
   console.log(s);
} // AUDI BMW undefined VW

// for (var i=0; i < samochody.length; i++) { … }
if (2 in samochody) // false
if (3 in samochody) // true
```

Tablice (2)

```
var x = new Array("HTML","CSS","JS");
// var x = ["HTML","CSS","JS"];

x[0] = "PHP";
console.log(x.length); // 3

console.log(x[5]) // undefined

x["abc"] = "xyz";
console.log(x["abc"]); // xyz
console.log(x.length); // 3

x[10] = 5;
console.log(x.length); // 11
```

```
for (s in x) {
    console.log(s);
}  // 0 1 2 abc
for (s of x) {
    console.log(s);
}  // PHP CSS JS
```

Nie ma tablic asocjacyjnych

W przykładzie "abc" to własność (property), którą ustawiliśmy dla obiektu

Tablice (3)

```
var x = new Array();
x[0] = 2008;
x[1] = "FIAT";
x[2] = {model: "126S"}
x[3] = function(stawka) {return stawka/100;};

console.log(x[0]);
console.log(x[1]);
console.log(x[2].model);
console.log(x[3](150));
```

Obiekty (1)

- Obiekty to kontenery nazwanych wartości (ang. properties)

```
var osoba = {imie: "Rysiu", nazwisko: "Kowalski", wzrost:159};
```

object literal

Property	Value
imie	Rysiu
nazwisko	Kowalski
wzrost	159

- Dostęp do wartości

```
osoba.imie // lub
osoba['imie']
```

- Built-in properties

```
osoba.imie.length
osoba['imie'].length
```

Obiekty (2)

- Tworzenie obiektów

```
var osoba = new Object();
osoba.imie = "Rysiu";
osoba.obliczWiek  = function(aktualnaData) {
  return …; };

// lub (wersja object literal)
var osoba = {
  imie: "Rysiu",
  obliczWiek:  function(aktualnaData) { return …; }
}

// lub (wersja z konstruktorem obiektowym)
function osoba(im) { // osoba to konstruktor
   this.imie = im;
   this.obliczWiek = function(aktualnaData) { return …; }
}
var rysiek = new osoba("Rysiu");
```

Obiekty (3)

- Dostęp do pól

```
console.log(osoba.imie);
console.log(osoba.obliczWiek(new Date()));
//lub
console.log(osoba["imie"]);
console.log(osoba["obliczWiek"](new Date()));
```

- Usuwanie pól

```
delete osoba.imie;
delete osoba["imie"];
```

- Wbudowane konstruktory, m.in.:

```
new Object();
new String();
new Number();
new Date();
```

- W praktyce lepiej unikać używania operatora `new` (i stosować tzw. prymitywy) – bardziej wydajne

Zasięg zmiennych

- Zmienne w JavaScript – także obiekty i funkcje
- Zasięg (scope) to zmienne, obiekty i funkcje, do których mamy dostęp
- Rodzaje zasięgu
  - lokalny – zmienne w funkcjach, dostęp tylko z tych funkcji, niszczone wraz z funkcją
    Uwaga: zmienne niezadeklarowane w funkcjach, stają się zmiennymi globalnymi!
  - globalny – zmienne poza funkcjami, dostęp ze wszystkich skryptów i funkcji, niszczone np. po zamknięciu strony
- Parametry funkcji – zachowują się jak zmienne lokalne w funkcjach

Zakres widoczności (1)

```
var x = 1;
f1();
function f1() {
    var x = 2;
    f2();
}
function f2() {
    console.log(x);
}

// wynik
1
```

```
var x = 1;
f1();
function f1() {
    console.log(x);
    f2();
}
function f2() {
    console.log(x);
}

//wynik
1
1
```

```
var x = 1;
f1();
function f1() {
    console.log(x);
    var x = 2;
    console.log(x);
    f2();
}
function f2() {
    console.log(x);
}

//wynik
undefined
2
1
```

Zakres widoczności (2)

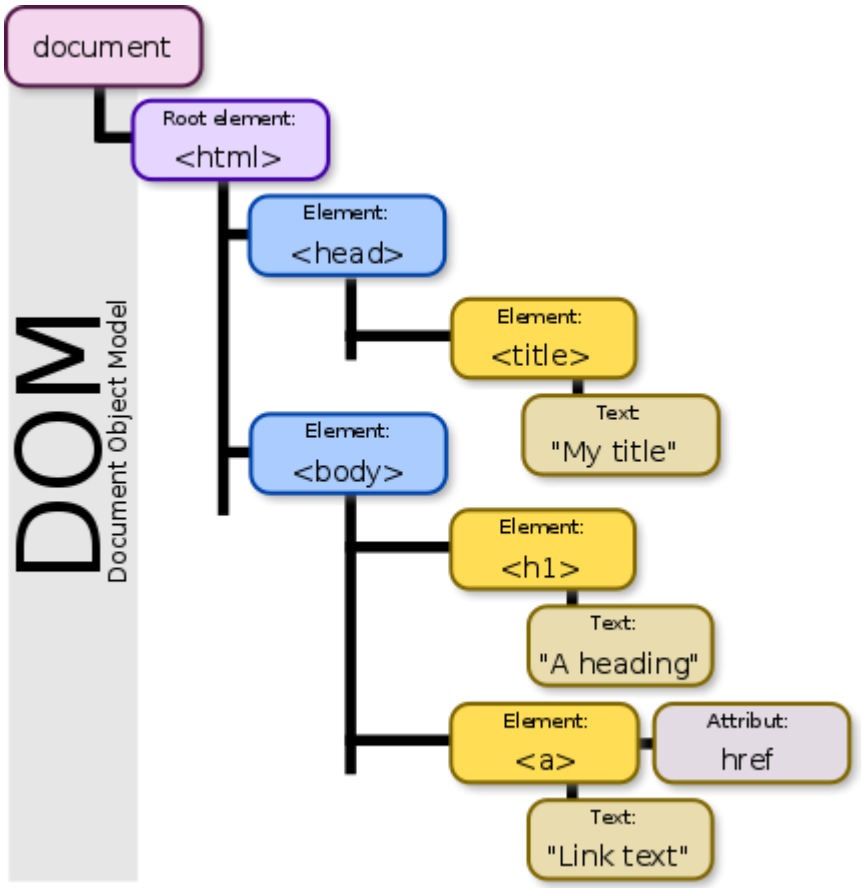
```
var x = 1;
f1();
function f1() {
    x = 2;
    f2();
}
function f2() {
    console.log(x);
}

// wynik
2
```

```
var x = 1;
f1();
function f1() {
    console.log(x);
    var x = 2;
    f2();
}
function f2() {
    console.log(x);
}

//wynik
undefined
1
```

```
var x = 1;
f1();
function f1() {
    console.log(
        parent.x);
    var x = 2;
    f2();
}
function f2() {
    console.log(x);
}

//wynik
1
1
```

DOM - Document Object Model

- Obiektowa reprezentacja strony
- Standard opracowany przez
  - W3C
  - WHATWG
- Możliwość dynamicznej reakcji na zmiany (Dynamic HTML)



Birger Eriksson, wikipedia.org

DOM & JavaScript

```
// wyszukiwanie po identyfikatorze (jeden element maksymalnie)
var myElem = document.getElementById("suma");

// wyszukiwanie po nazwie elementu
var x = document.getElementsByTagName("div");

// wyszukiwanie po nazwie klasy
var x = document.getElementsByClassName("student");

// wyszukiwanie po selektorze CSS (pierwsze dopasowanie)
var x = document.querySelector("div + p.student");

// wyszukiwanie po selektorze CSS (wszystkie pasujące)
var x = document.querySelectorAll("p.student");

// wykorzystanie znalezionego elementu
var footerElem = document.getElementById("footer");
var x = footerElem.getElementsByTagName("p");
```

Zwracanych może być wiele elementów

Poruszanie się po drzewie DOM

- Dla każdego węzła można użyć:
  - `parentNode`
  - `childNodes[`*`numer potomka`*`]`
  - `firstChild`
  - `lastChild`
  - `nextSibling`
  - `previousSibling`
- Uwaga na zawartość tekstową – to także nowy węzeł

```
<div id="nazwisko">Kowalski</div>
…
var myElem = document.getElementById("nazwisko");
var nazwisko = myElem.innerHTML; // lub
var nazwisko = myElem.firstChild.nodeValue;
```

Modyfikowanie drzewa DOM

```
<ol id="lista">Lista wiadomości</ol>
…
<script>
  var li = document.createElement("li");
  var textn = document.createTextNode("Nowa wiadomość");
  li.appendChild(textn);

  var div = document.getElementById("lista");
  div.appendChild(li);
</script>
```

- Inne metody:
  - `insertBefore(nowy_potomek, przed_ktorym)`
  - `removeChild(potomek)`
  - `replaceChild(nowy, poprzedni)`

Events

```
<element event='kod JavaScript'>
<element event="kod JavaScript">
```

```
<buton id="tb" onclick="displayTime()">Pokaż czas</button>
// lub
document.getElementById("tb").onclick = displayTime;
```

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

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

Event Listners

```
element.addEventListener("mouseover", funkcjaObslugi);
element.addEventListener("click", funkcjaObslugi);

function funkcjaObslugi {
// …
}

element.removeEventListener("click", funkcjaObslugi);
```

- Propagacja
  - bubbling – najpierw obsługa zagnieżdżonego, potem zewnętrznego – domyślne

```
addEventListener( zdarzenie, obsługa, false);
```

  - capturing – najpierw obsługa zewnętrznego, potem zagnieżdżonego

```
addEventListener( zdarzenie, obsługa, true);
```

Koniec