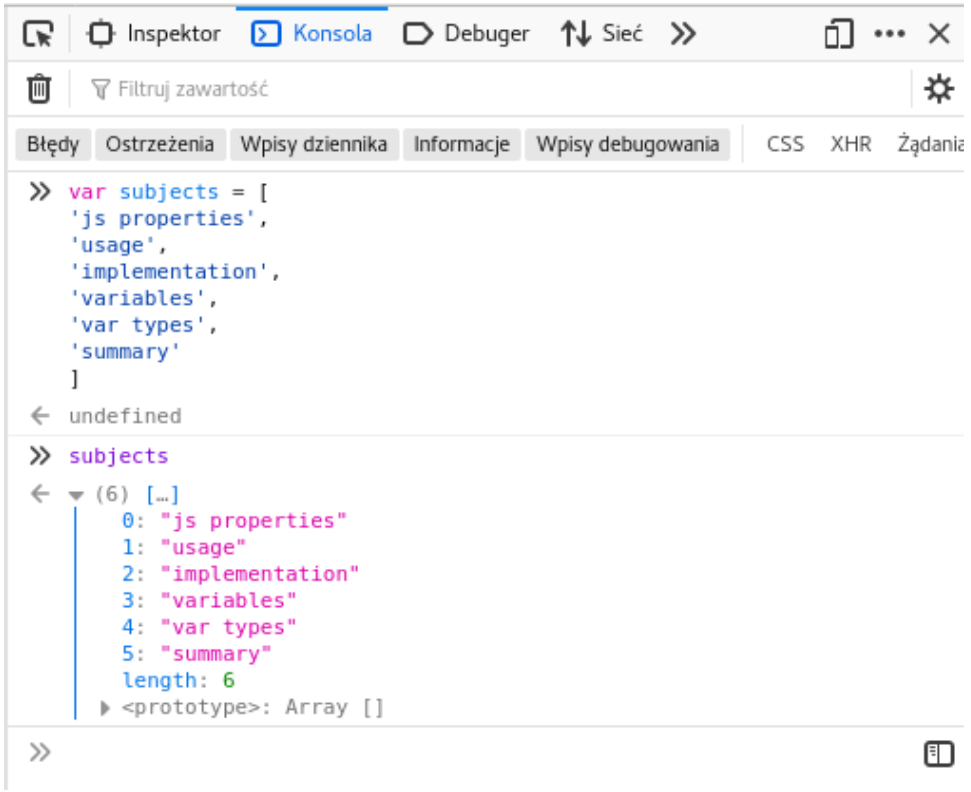
Programowanie Internetowe

- Standard ECMA Script
- Obiektowy model dokumentu
- Data, Czas, Wyrażenia regularne
- Przechowywanie danych
- Przetwarzanie asynchroniczne

Opracował: inż. Grzegorz Petri

Przegląd zagadnień

- Tablice
- Iterowanie przez tablice
- Tablice zagnieżdżone
- Typy Obiektów
- Struktura i budowa obiektu
- Iterowanie po obiekcie
- Tworzenie aplikacji



Tablica

struktura listo-podobna wysokiego poziomu

- **Zmienne** przechowują tylko **1 wartość**
- **Tablice** pozwalają przechowywać **więcej niż 1** wartość
- Tablice nie posiadają określonego typu przechowywanych wartości – pozwalają przechowywać **dowolny typ wartości**
- Podczas tworzenia tablic używany jest globalny obiekt Array

```
var tablica = [];// deklaracja pustej tablicy
var tablica = [5,10,15]; // deklaracja tablicy
typeof tablica;   // to obiekt…
Array.isArray(tablica);  // …czy to tablica?
```

Długość tablicy i jej elementy

- Sprawdzenie długości umożliwia właściwość length
- Dostęp do pozycji elementu umożliwia tzw. indeks numeryczny
- Indeks jak w każdej tablicy zaczyna się od 0 dla 1-go elementu
- JS jest tolerancyjny – nie generuje błędów wywołania nieistniejących elementów tablic

```
var test = new Array(5);
var imiona = new Array('Aga','Iga','Ola');
imiona.length; // długość tablicy to 3 elementy
imiona[1]; // dostęp do którego elementu?
imiona[3]; // indeks poza zakresem tablicy… czy dostaniemy się… do Uli?
```

Tablice puste i zagnieżdżone

Przykłady

```
var dziurawka = [];

dziurawka[3] = ‘wartość’;
dziurawka[5] = ‘wartość’;
dziurawka[7] = ‘wartość’;

dziurawka.length; // 8 elem.
console.log(dziurawka);

// wyjście konsoli
Array(8) [
   <3 empty slots>,
   "wartość",
   <1 empty slot>,
   "wartość",
   <1 empty slot>,
   "wartość"
]
```

```
var tablica = [5,10,15];

var obiekt = {s1:’X’,s2:’L’}

var nested = [];

nested[1] = obiekt;
nested[2] = [5,10,15];
nested[3] = ‘wartość’;
nested[4] = tablica;
nested[5] = ‘wartość’;
nested[6] = {s1:’X’,s2:’L’};
nested[7] = ‘wartość’;

nested.length;   // 8 elementów
nested[2].length; // 3 elementy
Object.keys(nested[6]).length;
// 2 elementy (klucze)
```

Manipulowanie zawartością tablic

Metody (wybrane) obiektu Array

`imiona.push('Ula');`	dodaj element na końcu tablicy
`imiona.pop();`	usuwa ostatni element tablicy, czyli ... ?
`imiona.shift();`	usuwa pierwszy element tablicy
`imiona.unshift('Ala');`	dodaje pierwszy element tablicy
`imiona.indexOf('Iga');`	sprawdźmy gdzie jest teraz Iga
`rem=imiona.splice(2,0,'Ewa');`	wstawia element na wskazaną pozycję (2)
`imiona.splice(2,1,'Ewelina');`	podmienia element(y) na pozycji (2)
`kolezanki = imiona.slice(0,2);`	wydziela 2 elementy od pozycji 0 tablicy
`znajome = imiona.slice(-2);`	wydziela 2 elementy od końca tablicy
`znajome.includes('Judi');`	TRUE jeżeli element jest w tablicy, FALSE gdy brak

Kopiowanie tablicy: https://developer.mozilla.org/pl/docs/Web/JavaScript/Referencje/Obiekty/Array/slice

Przypadki użycia: https://developer.mozilla.org/pl/docs/Web/JavaScript/Referencje/Obiekty/Array/splice

Wszystkie metody: https://developer.mozilla.org/pl/docs/Web/JavaScript/Referencje/Obiekty/Array

Pętle i tablice

Iterowanie przez tablice

Istnieje kilka sposobów na iterowanie poprzez elementy tablicy

- for ... i
```
for( var i=0; i<liczbaElementow; i++){ … }
```
- for ... in
```
for( let element in tablica){ … }
```
- for ... of
```
var iterator = tablica.values();
for( let element of iterator ){ … }
```
- for ... of
```
for(const [index, elem] of tablica.entries()){…}
```
- Array.prototype.forEach
```
tablica.forEach( nazwaFunkcji() );
```

Użycie tablic

Przykłady

```
var tablica = [5,10,15];


var ileElem = tablica.length;//3
for( var i=0; i<ileElem; i++){
    console.log(tablica[i]);
}
```

```
for( let element in tablica){
    console.log(element);
}
for( let key in tablica){
    console.log(„Klucz: ”+key);
    console.log(„W”+tablica[key]);
}
```

```
var iterator = tablica.values();
// zawartość zmiennej iterator?
for( let element of iterator ){
    console.log(element);
}
```

```
for(const [key, val] of
            tablica.entries()){
    console.log(„Klucz: ”+key);
    console.log(„Wartość: ”+val);
}
```

Obiekty

Podstawowe obiekty

- Object
- Function
- Boolean
- Symbol
- Error:
  - EvalError
  - InternalError
  - RangeError
  - ReferenceError
  - SyntaxError
  - TypeError
  - URIError

Liczby i daty:

- Number
- Math
- Date

Przetwarzanie tekstu:

- String
- RegExp

Kolekcje:

- Array, ArrayBuffer
- Map, Set
- DataView, JSON

Obiekty

Składowe i budowa

- ✔ *„Rozbudowana tablica”*
- ✔ Podstawowe (*predefiniowane w JS*) oraz Własne (*użytkownika*)
- ✔ Strukturalnie odpowiadają tablicom asocjacyjnym - element składa się z:
  - ➔ klucza
  - ➔ wartości
- ✔ Posiadają właściwości oraz metody (*właściwości dziedziczone, jeżeli są*)
- ✔ Pozwala grupować w/w elementy
- ✔ Object.prototype – wyświetla strukturę obiektu

```
var obiekt = {
   klucza: true,
   kluczb: 123,
   kluczc: "wartość",
   kluczd: [5,10,15],
   klucze: {},
   kluczf: function(){
      // ciało metody
   } // brak przecinka!
}    // brak średnika !

obiekt.kluczc;
obiekt.kluczf();
console.log(obiekt);
```

Porównanie

Kod funkcyjny oraz obiektowy

- ✗ Trudny w zarządzaniu
- ✗ Kod spaghetti - jest „wszędzie”
- ✗ Konieczna kontrola zmiennych, żeby się nie powtarzały

```
var czyStartPc = false;
var czyStartLcd = false;
var czyStartHdd = false;

function startPc(){startHdd}
function startLcd(){startPc}
function startHdd(){startLcd}

function uruchom(){
    startPc(); startLcd();
    startHdd();
}
```

- ✔ *Prosty w zarządzaniu i utrzymaniu*
- ✔ *Kod jest czytelny oraz zrozumiały*
- ✔ *Kontrola kluczy w obrębie obiektu, poza nim mogą się powtarzać*

```
let pc = {
    czyStart: false,
    uruchom: function(){
    lcd.start();
    hdd.start();
}, … }
let lcd={ czyStart: false,
          start: function(),…}
let hdd={ czyStart: false,
          start: function(),…}
pc.uruchom();
```

Iterowanie po obiekcie

Klucze i ich wartości

```
var mojObiekt = {
    klucz1: true,
    klucz2: 123,
    klucz3: 'wartość',
    klucz4: [5,10,15],
    klucz5: {},
    klucz6: function(){
        console.log('metoda');
    }
}
```

```
Klucz: klucz1
Wartość: true
Klucz: klucz2
Wartość: 123
Klucz: klucz3
Wartość: wartość
Klucz: klucz4
Wartość: 5,10,15
Klucz: klucz5
Wartość: [object Object]
Klucz: klucz6
Wartość: function()
{console.log('metoda');}
```

```
for( let [klucz,wartosc] of Object.entries(mojObiekt) )
{
        console.log('Klucz: '+klucz);
        console.log('Wartość: '+wartosc);
}
```

```
dialog(
   'Pytania?'
);
```