ECMA SCRIPT 2015

Frontend Development

Ecma International

- European Association for Standardizing Information and Communication Systems – stowarzyszenie, które powstało w 1961 roku, by ustandaryzować systemy informatyczne w Europie. Członkami ECMA są firmy zajmujące się systemami informatycznymi i telekomunikacyjnymi w Europie.

ECMAScript

- Ustandaryzowany przez ECMA obiektowy skryptowy język programowania, którego najbardziej znanymi implementacjami są JavaScript, JScript oraz ActionScript.

ECMAScript 2015

- Trwają prace nad siódmą edycją ECMAskryptu, ale obecnie w użyciu jest edycja szósta - ECMAScript 2015.
  Korzystają z niego między innymi popularne przeglądarki – Firefox, Chrome, Opera, Internet Explorer, a oprócz tego produkty firmy Adobe: rozszerzenia programów After Effect i Photoshop.

- ECMAScript 2015 dodaje nową składnię pozwalającą na pisanie złożonych aplikacji przy użyciu klas.

- Do nowych funkcji należą także:

```
class SkinnedMesh extends THREE.Mesh {
  constructor(geometry, materials) {
    super(geometry, materials);

    this.idMatrix = SkinnedMesh.defaultMatrix();
    this.bones = [];
    this.boneMatrices = [];
    //...
  }
  update(camera) {
    //...
    super.update();
  }
  static defaultMatrix() {
    return new THREE.Matrix4();
  }
}
```

Moduły

```
1  // ---- lib/math.js ----
2  let notExported = 123;
3
4  export function sum(x, y) {
5      return x + y;
6  }
7
8  export var pi = 3.141593;
9
10 // ---- app.js ----
11 // nazwa modułu na podstawie strutkury folderów
12 import { sum as sumarize, pi } from 'lib/math'
13
14 console.log('2n = ' + sumarize(pi, pi));
```

W jednym pliku możemy definiować co ma zostać wyeksportowane za pomocą słowa kluczowego export, a w innym możemy to zaimportować za pomocą słowa kluczowego import. Jak widać można przezywać importowane funkcje i właściwości aby uniknąć konfliktów nazw itp.

Iteratory, pętle for..of

```
let fibonacci = {
[Symbol.iterator]() {
let pre = 0, cur = 1;
return {
next() {
      [pre, cur] = [cur, pre + cur];
      return { done: false, value: cur }
} } } }

for (var n of fibonacci) {
// truncate the sequence at 1000
if (n > 1000)
break;
console.log(n);
}
```

Poprzednio istniała tylko pętla for..in która zastosowana na obiekcie bez wartości enumerable ustawionej na true zwracała tylko kolejne elementy, a nie ich wartość. Teraz z pętlą for..of możemy operować na prawdziwych wartościach.

Generatory w stylu Pythona

- Funkcja zadeklarowana jako function* i zawierająca przynajmniej jeden yield staje się generatorem. Generatory to takie funkcje gdzie przepływ działań może pyć spauzowany i kontynuowany w celu stworzenia sekwencji wartości.

```
1  function* idMaker(){
2    var index = 0;
3    while(true)
4      yield index++;
5  }
6
7  var gen = idMaker();
8
9  console.log(gen.next().value); // 0
10 console.log(gen.next().value); // 1
11 console.log(gen.next().value); // 2
12 // ...
```

Funkcje z użyciem strzałek

Poprzednio trzeba było robić tak:

```
var result,
numbers = [1,2,3,4,5];
result = numbers.map(function(x) {
    return x + x;
});
```

To funkcja wywołania zwrotnego przekazywana do funkcji 'map'. Teraz można napisać ją w ten sposób:

```
var result,
numbers = [1,2,3,4,5];
result = numbers.map(x => x + x);
```

Widzimy że zamiast funkcji występuje wyrażenie ze =>. Zamiast pisać całą funkcję wystarczy jedna linijka, która robi to samo.

Liczby binarne i ósemkowe

Dodane są dwie formy pozwalające operować na liczbach w systemie binarnym i ósemkowym.

```
0b111110111 === 503 // true
0o767 === 503 // true
```

Promises

Obietnice są wykorzystywane przy asynchronicznym programowaniu; reprezentują zmienne które mogą być dostępne w przyszłości.

```
function doSomething() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      console.log('robię coś...');
      resolve();
    }, 1500);
  });
}

doSomething()
  .then(() => console.log('a teraz robię coś jeszcze'));
```

Maps, sets oraz weak maps

```
// Sets
var s = new Set();
s.add("hello").add("goodbye").add("hello");
s.size === 2;
s.has("hello") === true;

// Maps
var m = new Map();
m.set("hello", 42);
m.set(s, 34);
m.get(s) == 34;

// Weak Maps
var wm = new WeakMap();
wm.set(s, { extra: 42 });
wm.size === undefined

// Weak Sets
var ws = new WeakSet();
ws.add({ data: 42 });
// Because the added object has no other references, it will not be held in the set
```

Większa dowolność przy tworzeniu stringów

```
// Basic literal string creation
`This is a pretty little template string.`

// Multiline strings
`In ES5 this is
 not legal.`

// Interpolate variable bindings
var name = "Bob", time = "today";
`Hello ${name}, how are you ${time}?`

// Unescaped template strings
String.raw`In ES5 "\n" is a line-feed.`
```