Technologie Internetu - wykład

Programowanie obiektowe w JavaScript

Obiektowość w JavaScript

Wszystko jest obiektem

- Tablica, string, funkcja… wszystko to tak naprawdę tworzy obiekt w JS
- Elastyczne podejście do obiektowości

Obiekty w JS

- Jako zmienne do przechowywania danych
- Jako funkcje
- Jako instancje klas

Obiekt jako zmienna

Struktura, przechowująca dane

- Nie ma konstruktora, instancji.
- Pola i metody.
- Klucze i wartości.
- Wszystko publiczne.

```
var ob = {

  imie: “Jan”,
  nazwisko: “Kowalski”,
  wiek:29,
  wzrost:194

};
```

Obiekt jako zmienna

Do wszystkiego mamy pełen dostęp z zewnątrz.

- Odwołanie do pola obiektu.
- Możliwość nadpisania wartości.
- Możliwość dodania nowych pól.

```
var ob = {

  imie: “Jan”,
  nazwisko: “Kowalski”,

};

let x = ob.imie;
ob.nazwisko = “zmienione”;
ob.adres = “Ulica testowa”;
```

Kopiowanie obiektów

Obiekty i tablice w JS przekazujemy przez referencje

- Przypisując lub przekazując obiekt, przekazujemy tak naprawdę adres do niego.
- W takim wypadku zmiana właściwości będzie widoczna w obu obiektach.

```
var ob = {

  imie: "Jan",
  nazwisko: "Kowalski",

};

var ob2 = ob;

console.log(ob2.imie);
//zmienione
```

Kopiowanie obiektów

Obiekty musimy kopiować ręcznie

- Po kolei, wartość po wartości.
- Pętla for in może iterować po obiekcie z danymi.

```
var ob = {
  imie: "Jan",
  nazwisko: "Kowalski",
};

var ob3 = {};
for(i in ob) {
    ob3[i] = ob[i];
}
ob.imie = "zmienione";
console.log(ob3.imie); //Jan
```

Pola i metody

Właściwościami obiektu mogą być też funkcje

```
var ob = {
  imie: "Jan",
  nazwisko: "Kowalski",
  hello: function() {
    console.log("Jestem " + this.imie);
  }
};

ob.hello();
```

JSON

- JavaScript Object Notation.
- Sposób na przechowywanie danych w JS.
- JSON to tak naprawdę ciąg znaków (string), zawierający wpisany obiekt JS.
- JSON to lekki i czytelny format zapisu danych.
- Dlatego przyjął się jako standard, także poza JavaScriptem.
- Na przykład w przesyłaniu danych między klientem a serwerem.

JSON

Obiekt JS zapisany do stringa.

```
var ob = {imie: "Jan", nazwisko: "Kowalski"};

var st = `{"imie":"Jan","nazwisko":"Kowalski"}`;
```

JSON

Obiekt JS zapisany do stringa.

- JSON.parse() wczytuje JSON-owy string do obiektu.
- JSON.stringify() zamienia JavaScriptowy obiekt w stringa.

```
var ob = {
  imie: "Jan",
  nazwisko: "Kowalski",
};

var st = JSON.stringify(ob);
console.log(st);

var ob2 = JSON.parse(st);
console.log(ob2);
```

Obiekt - inny sposób

Obiekt jako funkcja

- Definicja obiektu jako funkcji, która go tworzy.
- Prosty konstruktor.
- Odwołanie do właściwości przez this.
- Uruchamianie poprzez new.
- Można stworzyć dowolną ilość obiektów bez kopiowania.

```
function samochod(nazwa, moc){
  this.nazwa = nazwa;
  this.moc = moc;
  this.pokaz = function(){
 return this.nazwa+' '+this.moc;
  }
}

var s = new samochod("fiat",99);
var x = s.pokaz();
```

Obiekt - ECMA6

Definicja klasy zgodna ze współczesnymi standardami obiektowości.

- Słowo kluczowe class.
- Słowo kluczowe constructor.
- Pola i metody.
- Odwołanie przez this.
- Tworzenie przez new.

```
class Car{
    brand;
    model;
    power;
    constructor(brand, model){
        this.brand = brand;
        this.model = model;
    }
    go = function(x){
        this.power = x;
    }
}
```

Obiekt - ECMA6

Definicja klasy zgodna ze współczesnymi standardami obiektowości.

- Słowo kluczowe class.
- Słowo kluczowe constructor.
- Pola i metody.
- Odwołanie przez this.
- Tworzenie przez new.

```
//nowy obiekt klasy
var a = new Car("opel", "adam");

//wywołanie metody obiektu
a.go(10);

//odwołanie do pola obiektu
console.log(auto.power);
```

Obiekt - ECMA6

Pola prywatne.

- Dobra praktyka: pola prywatne, metody publiczne.

```
class Car{
    brand;
    model;
    #power; //private

    go = function(x){
        this.#power = x;
    }
}
```

Obiekt - ECMA6

Pola prywatne.

- Dobra praktyka: pola prywatne, metody publiczne.
- Getter do pobierania wartości. Słowo kluczowe get.
- Setter do ustawiania wartości. Słowo kluczowe set.
- Wywołanie gettera i settera z zewnątrz wygląda jak zwykłe pole.

```
class Car{
    #power;

    get power(){
        return this.#power;
    }
    set power(p){
        this.#power = p;
    }
}

var a = new Car();

a.power = "100"; //wywoła setter
x = a.power; //wywoła getter
```

Dziedziczenie

- Tworzymy obiekt rozbudowujący inny obiekt
- Dzięki temu możemy mieć całą hierarchię obiektów
- Metoda call pozwala wywołać konstruktor obiektu
- W tym wypadku jest to obiekt klasy bazowej

```
//klasa bazowa
function Auto(name) {
this.name = name
this.go = function(){}
}

//klasa dziedzicząca
function Truck(name) {
    Auto.call(this, name);
}

//przykładowe wywołanie
let t = new Truck("test");
t.go();
```

Dziedziczenie ECMA6

```
class Animal{
    constructor(name, size) {
        this.name = name;
        this.size = size;
    }
    hello() {
       console.log(this.name);
    }
}
```

```
class Bird extends Animal{
  constructor(name, size, speed) {

    //super: konstruktor bazowy
    super(name, size);
    this.speed = speed;
  }

  //override - nadpisanie metody
  hello() {
    console.log("i am a bird");
  }
}

var a = new Bird("sikorka", 1, 2);
a.hello();
```

Prototypy

- JS przechowuje w pamięci prototypy obiektów.
- Dzięki temu możemy w dowolnym momencie modyfikować obiekty.

```
function Animal(name, size) {
    this.name = name;
    this.size = size;
}

//dodajemy nową metodę do
obiektu
Animal.prototype.hello =
function() {
    console.log(this.name);
}
```

Prototypy

Możemy modyfikować istniejące obiekty

- Prototypy pozwalają nam zmodyfikować także wbudowane obiekty JS.
- Takie jak array, string, Date, Math.
- Oczywiście modyfikacja jest widoczna tylko w naszym projekcie.

```
//dodajemy funkcję do tablic
//zwracającą jej drugi element

Array.prototype.drugi
= function() {return this[1];}

let a = [10, 20, 30, 40, 50];

console.log(a.drugi())
```

Destrukturyzacja

Szybkie przepisanie właściwości obiektu

- Zamiast po kolei przypisywać właściwości obiektu do zmiennych…
- … można przypisać je wszystkie naraz.
- W jednej instrukcji tworzymy dowolną ilość zmiennych/stałych i przypisujemy do nich obiekt.
- Muszą się nazywać jak pola obiektu.

```
var s = {
    imie: "Jan",
    nazwisko: "Kowalski",
    wiek: 20
}

let {imie, nazwisko, wiek} = s;
```