Tworzenie Aplikacji Internetowych
Laboratorium 5

Programowanie obiektowe w JavaScript

Wraz z wersją ECMAScript 2015 (**ES6**) wprowadzono słowo kluczowe **class** pozwalające tworzyć klasy w języku JavaScript. Wcześniej należało wykorzystywać funkcje typu konstruktor lub tworzyć obiekty dynamicznie bez deklaracji klasy.

Klasy tworzy się i wykorzystuje następująco:

- Do utworzenia konstruktora należy wykorzystać słowo kluczowe **constructor** podając w nawiasie parametry (w JS **nie można** przeciążać metod ani konstruktorów)
- Właściwości należy utworzyć w **konstruktorze** po słowie kluczowym **this** (można je definiować poza konstruktorem jedynie w nowszej wersji JS)
- Aby zdefiniować **metodę** w klasie wystarczy podać jej nazwę, parametry w nawiasie i ciało metody wewnątrz nawiasu klamrowego
- Do właściwości obiektu wewnątrz metod należy odwoływać się przez słowo kluczowe **this**
- Do właściwości oraz metod obiektu po jego utworzeniu należy się odwołać za pomocą **kropki .**
- **Nie ma** możliwości tworzenia właściwości oraz metod **prywatnych**
- Obiekty dla utworzonej klasy tworzy się za pomocą słowa kluczowego **new**

Działanie klas przedstawia następujący przykład:

```
class Pojazd {
   constructor(marka, model) {
      this.marka = marka;
      this.model = model;
      this.przebieg = 0;
   }
   dodajDane(ile) {
      this.przebieg += ile;
   }
}

const s1 = new Pojazd("Toyota", "GT86");
const s2 = new Pojazd("Toyota", "AE86");
const s3 = new Pojazd("Fiat", "126p");
s1.dodajDane(100);
s2.model = "Zmiana modelu";
```

Mechanizm **dziedziczenia** działa następująco:

- Do dziedziczenia należy wykorzystać słowo kluczowe **extends**
- Do konstruktora klasy po której się dziedziczy należy się odwołać za pomocą **super**
- Do metod klasy bazowej należy się odwołać za pomocą **super.nazwaMetody**

Działanie dziedziczenia przedstawia następujący przykład:

```
class Samochod extends Pojazd {
   constructor(marka, model, drzwi) {
      super(marka, model); // konstruktor bazowy i przekazanie parametrów
      this.drzwi = drzwi;
      this.plyn = 0;
   }
   dodajDane(ile, pl) {
      super.dodajDane(ile); // metoda bazowa o tej samej nazwie
      this.plyn += pl;
   }
}
```

Zadanie 1

- Utworzyć klasę **Osoba** (z odpowiednim konstruktorem) zawierającą właściwości: *imie*, *nazwisko*, *rokurodzenia* oraz metodę *wypiszInformacje* wypisującą te informacje w konsoli: (console.log(zmienna1, zmienna2, …).
- Utworzyć kilka obiektów typu Osoba i wywołać dla nich metodę *wypiszInformacje*, wynik zaobserwować w konsoli (F12)
- Utworzyć klasę **Pracownik** **dziedziczącą** po klasie **Osoba**, zawierającą dodatkowo takie informacje jak *pensja* i *premia*. Utworzyć metodę *wypiszInformacje* wypisującą wszystkie informacji w konsoli (do wywołania metody z klasy bazowej w konstruktorze oraz *wypiszInformacje* wykorzystać słowo **super**)
- Dla klasy **Pracownik** utworzyć metodę (dowolna nazwa) zwracającą obiekt DOM reprezentujący wiersz tabeli (czyli znacznik **tr** z umieszczonymi wewnątrz znacznikami **td**). W zwracanym wierszu tabeli powinny zostać zawarte kolejno:
    1. Imię oraz pierwsza litera nazwiska – komórka 1
    2. Pensja – komórka 2
    3. Premia – komórka 3

  Zamiast obiektu DOM można w wersji uproszczonej zwrócić kod HTML. Dla przypomnienia obiekty DOM tworzy się następująco: let ob = document.createElement("znacznik");
- Utworzyć na stronie tabelę o id="dane". W skrypcie utworzyć kilka obiektów klasy Pracownik, dla każdego pobrać wiersz z danymi (utworzona w poprzednim punkcie metoda) i umieścić w tabeli o id="dane" (obiekty DOM po ID można pobrać: let gdzie = document.getElementById("id"))

Zadanie 2

Celem tego zadania jest zaznajomienie się z praktycznym przykładem wykorzystującym obiekty. Przykład ten pokazuje początek implementacji gry typu arkanoid – ze względu na wiele uproszczeń system kolizji jest niedokładny (po kolizji obiekt zostaje w nieuaktualnionym miejscu, co pomoże powodować pewne błędy, jednak nie ma znaczenia w kontekście rozważanego zadania).

W celu realizacji zadania należy:

- Przekopiować i uruchomić plik HTML zaczynający się na następnej stronie (objętość 2 stron)
- Przyjrzeć się klasie **Bazowy**, w szczególności metodzie **usun** (metoda ta jest wywołana po kolizji klocka z piłką – ponieważ zawsze zwraca *true* klocek zostanie zawsze usunięty po kolizji)
- Przyjrzeć się klasie **Zwykly** – różnica względem klasy Bazowy to jedynie przypisanie klasy zwykły (w stylach CSS definuje ona kolor klocka)
- Analogicznie do klasy **Zwykly** utworzyć klasę **Niezniszczalny**, nadając także nowy styl CSS (inny kolor klocka). W klasie **Niezniszczalny** **przeciążyć** metodę usun, tak aby zwracała zawsze false. Umieścić kilka takich obiektów na planszy i przetestować aplikację.
- Analogicznie do klasy **Zwykly** utworzyć klasę **Kolorowy**, która zamiast przypisywać styl CSS, przypisze utworzonemu klockowi losowy kolor: this.ksztalt.style.fill = "rgb("+r+","+g+","+b+")"; Zmienne r, g, oraz b wylosować z zakresu od 0 do 255: let r = Math.floor(Math.random() * 256) Zastąpić klocki **Zwykle** na planszy klockami **Kolorowy** i przetestować działanie aplikacji.
- (**opcjonalnie**) Dodać klocki które znikają po trzech kolizjach (a po każdej z nich zmieniają kolor lub tracą przeźroczystość)
- (**opcjonalnie**) Dowolnie rozbudować aplikację (np. dodając klocki, które po zbiciu dodają dodatkowe piłki na planszy, poprawiając kolizję, dodając paletkę sterowaną za pomocą myszki, itp.)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <style>
            #plansza { width: 1080px; height: 520px; border: 1px solid silver; }
            #plansza rect { stroke: silver; stroke-width: 1px; }
            #pilka { width: 16px; height: 16px; fill: gray; }
            .zwykly { fill: navy; }
        </style>
    </head>
    <body>
        <svg id="plansza"></svg>
        <script>
            const nssvg = "http://www.w3.org/2000/svg";
            const plansza = document.getElementById("plansza");
            const pwidth = 1080; // szerokość okna gry
            const pheight = 520; // wysokość okna gry
            const kwidth = 64; // bazowa szerokość klocka
            const kheight = 32; // bazowa wysokość klocka

            class Pilka { // bazowa piłka
                constructor(x, y, r, p, k) {
                    this.x = x; // położenie obiektu w poziomie
                    this.y = y; // położenie obiektu w pionie
                    this.r = r; // wielkość piłki
                    this.p = p; // prędkość piłki w poziomie
                    this.k = k; // kąt poruszania się piłki
                    // wygląd obiektu zdefiniowany za pomocą SVG
                    this.ksztalt = document.createElementNS(nssvg, "circle");
                    this.ksztalt.setAttribute("cx", this.x);
                    this.ksztalt.setAttribute("cy", this.y);
                    this.ksztalt.setAttribute("r", this.r);
                }
                umiesc() { // umieszczenie piłki na planszy
                    plansza.appendChild(this.ksztalt);
                }
                rusz() { // przemieszczenie piłki
                    this.x += Math.sin(this.k) * this.p;
                    this.y += Math.cos(this.k) * this.p;
                    if ((this.x - this.r) < 0) this.odwrocPoziom();
                    else if ((this.x + this.r) > pwidth) this.odwrocPoziom();
                    if ((this.y - this.r) < 0) this.odwrocPion();
                    else if ((this.y + this.r) > pheight) this.odwrocPion();
                }
                rysuj() { // narysowanie piłki w nowej pozycji
                    this.ksztalt.setAttribute("cx", this.x);
                    this.ksztalt.setAttribute("cy", this.y);
                }
                odwroc(klocek) { // odwrócenie kierunku ruchu piłki po kolizji
                    if ((this.x > klocek.x) && (this.x < (klocek.x + klocek.w))) {
                        this.x -= Math.sin(this.k) * this.p;
                        this.y -= Math.cos(this.k) * this.p;
                        this.odwrocPion();
                    } else {
                        this.x -= Math.sin(this.k) * this.p;
                        this.y -= Math.cos(this.k) * this.p;
                        this.odwrocPoziom();
                    }
                }
                odwrocPion() { this.k = Math.PI - this.k; }
                odwrocPoziom() { this.k = -this.k; }
            }

            class Bazowy { // bazowy klocek
                constructor(x, y, w, h) {
                    this.x = 28 + x * kwidth; // położenie obiektu w poziomie
                    this.y = y * kheight; // położenie obiektu w pionie
                    this.w = w; // szerokość klocka
                    this.h = h; // wysokość klocka
                    // wygląd obiektu zdefiniowany za pomocą SVG
                    this.ksztalt = document.createElementNS(nssvg, "rect");
                    this.ksztalt.setAttribute("x", this.x);
                    this.ksztalt.setAttribute("y", this.y);
                    this.ksztalt.setAttribute("width", this.w);
```

```
                    this.ksztalt.setAttribute("height", this.h);
                }
                usun() { return true; } // czy usunąć po zbiciu
                umiesc() { // umieszczenie klocka na planszy
                    plansza.appendChild(this.ksztalt);
                }
                kolizja(pilka) { // sprawdzenie kolizji
                    // czy na zewnątrz klocka? (przyśpieszenie obliczeń)
                    if ((pilka.x + pilka.r) < this.x) return false;
                    if ((pilka.x - pilka.r) > (this.x + this.w)) return false;
                    if ((pilka.y + pilka.r) < this.y) return false;
                    if ((pilka.y - pilka.r) > (this.y + this.h)) return false;
                    // czy wewnątrz klocka?
                    if ((pilka.x >= this.x) && (pilka.x <= (this.x + this.w))
                    && (pilka.y >= this.y) && (pilka.y <= (this.y + this.h))) return true;
                    // dokładna kolizja z najbliższymi krawędziami
                    let cx = pilka.x;
                    let cy = pilka.y;
                    if (cx < this.x) cx = this.x;
                    else if (cx > (this.x + this.w)) cx = this.x + this.w;
                    if (cy < this.y) cy = this.y;
                    else if (cy > (this.y + this.h)) cy = this.y + this.h;
                    let dx = pilka.x - cx;
                    let dy = pilka.y - cy;
                    let dist = dx * dx + dy * dy;
                    if (dist < (pilka.r * pilka.r)) return true;
                    return false; // brak kolizji
                }
            }

            class Zwykly extends Bazowy { // zwykły klocek - znika po uderzeniu
                constructor(x, y) {
                    super(x, y, kwidth, kheight);
                    this.ksztalt.classList.add("zwykly");
                }
            }

            let klocki = []; // tablica klocków
            for (let j = 0; j < 4; j++) {
                for (let i = 0; i < 16; i++) {
                    let klocek = new Zwykly(i, j + 1); // utworzenie klocka
                    klocki.push(klocek); // dodanie klocka do tablicy
                    klocek.umiesc(); // umieszczenie klocka na planszy
                }
            }

            let pilki = []; // tablica piłek - domyślnie jedna piłka
            let pilka = new Pilka(pwidth / 2, pheight - 64, 16, 5.5, 2.5);
            pilki.push(pilka); // dodanie piłki do tablicy
            pilka.umiesc(); // umieszczenie piłki na planszy

            setInterval(function() {
                for (let i = 0; i < pilki.length; i++) { // aktualizacja piłek
                    pilki[i].rusz(); // oblicz nową pozycję piłki
                    for (let k = 0; k < klocki.length; k++) { // sprawdź kolizję z klockami
                        if (klocki[k].kolizja(pilki[i])) { // sprawdzenie kolizji
                            pilki[i].odwroc(klocki[k]); // odwrócenie lotu piłki
                            if (klocki[k].usun()) {
                                plansza.removeChild(klocki[k].ksztalt); // usunięcie z obrazka
                                klocki.splice(k, 1); // usunięcie klocka z tablicy
                            }
                            break; // koniec sprawdzania kolizji dla tej piłki
                        }
                    }
                    pilki[i].rysuj(); // narysuj piłkę
                }
            }, 10);
        </script>
    </body>
</html>
```