# JavaScript - Drzewo rekurencyjne



## Ogólne

Całe zadanie realizowane jest w jednym pliku (np. **drzewo.html**). Funkcja wywołuje samą siebie tak długo, aż gałęzie osiągną minimalną długość (w tym celu muszą być krótsze za każdym kolejnym przebiegiem rekurencji).

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii i tzw. twarde odstępy. Kod należy przepisać ze zrozumieniem.

## Kod - wersja podstawowa

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Drzewo</title>
</head>
<body>
<canvas width="600" height="600" id="miejsceNaDrzewoSymetryczne"></canvas>
<script>
function rysujGalaz(dlugoscGalezi, kierunek) {
	rysowanyObiekt.save();
	rysowanyObiekt.rotate(kat * kierunek);
	rysowanyObiekt.fillRect(-dlugoscGalezi / 20, 0, dlugoscGalezi / 10, -dlugoscGalezi);
	if (dlugoscGalezi > minimalnaDlugoscGalezi) {
		rysowanyObiekt.translate(0, -dlugoscGalezi);
		rysujGalaz(dlugoscGalezi * redukcja, -1);
		rysujGalaz(dlugoscGalezi * redukcja, 1);
	}
	rysowanyObiekt.restore();
}

var rysowanyObiekt = miejsceNaDrzewoSymetryczne.getContext("2d");
var szerokoscObiektu = rysowanyObiekt.canvas.width;
var wysokoscObiektu = rysowanyObiekt.canvas.height;
var kat = 0.35;
var dlugoscPierwszejGalezi = 75;
var minimalnaDlugoscGalezi = 5;
var redukcja = 0.75;
rysowanyObiekt.translate(szerokoscObiektu/2, wysokoscObiektu-10);
rysujGalaz(dlugoscPierwszejGalezi, 0);
</script>
</body>
</html>
```

# Kod - wersja rozszerzona

Uwaga: Do poniższego kodu *JavaScript* musisz jeszcze dopisać fragment kodu HTML, który będzie wyświetlał drzewo (*canvas*).

```
<script>
// FUNKCJA RYSUJĄCA DRZEWO:
function rysujGalaz(dlugoscPoprzedniejGalezi, kierunek) {
    // Zapisujemy wyniki poprzedniego przebiegu rekurencyjnego (te wyniki zapisywane są na stosie stanów, nie są nadpisywane!):
    rysowanyObiekt.save();
    // Wprowadzamy trochę chaosu, generujemy różne kąty odchylenia przy każdym przebiegu rekurencji:
    katOdchyleniaGalezi=(Math.random()*60)*Math.PI/180;
    // Tylko przy pierwszym przebiegu rekurencji rotacja jest równa 0 (pień drzewa), bo 0.35*0=0. Przy każdym następnym przebiegu, rotacja równa jest 0.35 albo -0.35 (czyli 20 stopni w jedną bądź drugą stronę), bo zmienna "kierunek" przyjmuje wartość 1 lub -1:
    rysowanyObiekt.rotate(katOdchyleniaGalezi * kierunek);
    // Gałęzie tak naprawdę są prostokątami (x, y, szerokość, wysokość)... czyli pierwsza gałąź przyjmie wartości: -3.75, 0, 7.5, -75:
    rysowanyObiekt.fillRect(-dlugoscPoprzedniejGalezi / 20, 0, dlugoscPoprzedniejGalezi / 10, -dlugoscPoprzedniejGalezi);
    // Rekurencja zostanie wykonana tylko wtedy, gdy nie osiągnięto minimalnej długości gałęzi:
    if (dlugoscPoprzedniejGalezi > minimalnaDlugoscGalezi) {
        // Przyjęcie nowego względnego układu odniesienia, początku współrzędnych x,y (czubek pnia jest teraz nowym układem odniesienia):
        rysowanyObiekt.translate(0, -dlugoscPoprzedniejGalezi);
        // Rekurencyjne wywołanie samej siebie i przekazanie nowych wartości argumentów (długość gałęzi zostaje za każdym wywołaniem zmniejszona do 75%, a kierunek jej odchylenia (rotacja) zmieniony na 0.35. Zostaną narysowane najpierw wszystkie lewe odgałęzienia aż dojdzie do minimalnej długości gałęzi i zatrzyma się (nie zostanie już wykonana wtedy poniższa prawostronna rekurencja):
        rysujGalaz(dlugoscPoprzedniejGalezi * procentRedukcjiPoprzedniejGalezi, -1);
        // W tym momencie funkcja zatrzyma się, gdyż osiągnięto minimalną długość gałęzi. Ale poniżej zostanie zrobiony restore() do poprzedniego stanu, więc przy następnym przebiegu uruchomi się i zostaną dorysowane wszystkie prawe odgałęzienia (prawostronna rekurencja), ale od czubka struktury zaczynając i schodząc w dół (stos LIFO):
        rysujGalaz(dlugoscPoprzedniejGalezi * procentRedukcjiPoprzedniejGalezi, 1);
    }
    // Gdy długość gałęzi jest mniejsza od minimalnej (czyli gałęzie zostały już narysowane),
    // wtedy robimy jeden przebieg rysowania liści (czyli na samych czubkach gałęzi):
    else {
        rysowanyObiekt.fillStyle = "#00f000"; // kolor liści
        rysowanyObiekt.fillRect(-minimalnaDlugoscGalezi / 2, 0, minimalnaDlugoscGalezi / 2, -1); // x, y, szerokość, wysokość
        rysowanyObiekt.fillStyle = "#00a000";
        rysowanyObiekt.fillRect(0, -minimalnaDlugoscGalezi, 1, minimalnaDlugoscGalezi);
        rysowanyObiekt.fillStyle = "#006000";
        rysowanyObiekt.fillRect(dlugoscPoprzedniejGalezi, -1, -dlugoscPoprzedniejGalezi, minimalnaDlugoscGalezi);
    }
```

```
        // Przywraca poprzednio zapisany stan obiektu (sprzed poprzedniego przebiegu rekurencji), czyli wraca do punktu save(), cofa się jakby o krok do tyłu (na stosie), w którym wartość długości poprzedniej gałęzi nie osiągnęła jeszcze minimum. To jak kliknięcie w "Cofnij" / "Wstecz".
        rysowanyObiekt.restore();
}

// FUNKCJA CZYSZCZĄCA PŁÓTNO:
function czyszczeniePlotna() {
        // Resetujemy układ odniesienia do początkowego / domyślnego:
        rysowanyObiekt.resetTransform();
        // Czyszczenie całego płótna:
        rysowanyObiekt.clearRect(0, 0, szerokoscObiektu, wysokoscObiektu);
        // Powrót do punktu, gdzie rysowane będzie kolejne drzewo:
        rysowanyObiekt.translate(szerokoscObiektu/2, wysokoscObiektu-20);
}


// Rysowanie 2-wymiarowe (2D). Inna opcja (3D) to "webgl":
var rysowanyObiekt = miejsceNaDrzewoChaotyczne.getContext("2d");

// Wstępna szerokość i wysokość drzewa to szerokość i wysokość płótna:
var szerokoscObiektu = rysowanyObiekt.canvas.width;
var wysokoscObiektu = rysowanyObiekt.canvas.height;

// Kolor wypełnienia:
rysowanyObiekt.fillStyle="#8b4513";

// Z każdej gałęzi wychodzą dwie następne pod kątem 20 stopni (wartość 0.35 radiana = 20 stopni, gdyż 0.35x180/3.14=20 oraz 20x3.14/180=0.35). Poniżej stopnie przeliczone na radiany:
var katOdchyleniaGalezi = 0.35;

// Długość pierwszej gałęzi / pnia drzewa (w pikselach):
var dlugoscPierwszejGalezi =  75;

// Długość ostatniej minimalnej gałęzi (w pikselach). Po osiągnięciu tej minimalnej długości rekurencja zostanie przerwana:
var minimalnaDlugoscGalezi = 5;

// Gdyby kolejne odgałęzienia były ciągle tej samej długości (100%), co pierwsza gałąź - rekurencja nigdy nie zostałaby przerwana, bo gałąź nigdy nie osiągnęłaby zakładanej minimalnej długości. Dlatego właśnie przy każdym kolejnym wywołaniu, gałąź powinna być skracana (jej długość będzie wynosić 75% poprzedniej długości). Im stopień redukcji długości będzie mniejszy, tym wystąpi więcej wywołań rekurencyjnych w celu osiągnięcia długości minimalnej (większa gęstość drzewa). Im większy stopnień redukcji długości, tym szybciej minimalna długość zostanie osiągnięta (drzewo będzie miało mniejszą gęstość, mniej rozgałęzień). Podsumowując: mniejsza wartość = mniej gałęzi. Poniżej, procenty przeliczone na wartość liczbową:
var procentRedukcjiPoprzedniejGalezi = 0.75;
```

```
// Współrzędne liczone są od lewego górnego narożnika płótna. Poniżej, pień rysowany jest na środku płótna, 10px od dołu. Ten punkt staje się nowym względnym układem odniesienia:
rysowanyObiekt.translate(szerokoscObiektu/2, wysokoscObiektu-20);

// Uruchamiamy funkcję, przekazujemy jej długość pierwszej gałęzi (pnia) oraz kąt obrotu pierwszej gałęzi równy 0 (brak obrotu):
rysujGalaz(dlugoscPierwszejGalezi, 0);
</script>

<button type="button" onclick="rysujGalaz(dlugoscPierwszejGalezi, 0)">Więcej gałęzi...</button>
<button type="button" onclick="czyszczeniePlotna(); rysujGalaz(dlugoscPierwszejGalezi, 0)">Nowe drzewo...</button>
```

Ostatnia aktualizacja: 3 kwietnia 2024.