# **Chart.js**

## Въведение

Chart.js е библиотека за изграждане на графики в уеб приложения с помощта на **HTML5 Canvas**. Тя е написана на **JavaScript** и предоставя лесен начин за създаване на различни типове графики, като линейни, кръгови, стълбови и други. Библиотеката е разработена за пръв път през **2013 г.** от **Nick Downie** и оттогава тя е продължила да се развива и подобрява. Тя е с отворен код, което позволява на разработчиците да я променят и да я използват в своите проекти безплатно. [**[6]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#6th_source)  
  
Позволява на разработчиците да настроят всички елементи на графиката, включително цветове, легенди, оси и много други. Тя изисква само малко HTML и JavaScript код, за да се добави към уеб страница и да се генерира графика. Библиотеката е съвместима със всички съвременни уеб браузъри, включително **Chrome, Firefox, Safari** и **Internet Explorer**.  
  
Chart.js е създадена с използване на **модулен подход**, който позволява на разработчиците да използват само тези функции и модули, които са им необходими. Това я прави изключително гъвкава и лесна за използване. Всички графики, генерирани с Chart.js, могат да се визуализират на мобилни устройства, като те са напълно **респонсивни** и се приспособяват към размера на екрана.

## Как да използвате Chart.js?

**Chart.js** е лесен за използване.  
  
Първо, добавете връзка към предоставящата **CDN** (мрежа за доставка на съдържание): [**[2]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#2nd_source)

**Код 1.**Включване на Chart.js библиотеката чрез връзка към CDN

<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>

Важно е да споменем, че този **скрипт** трябва да бъде добавен в края на **html** документа, преди затварянето на **body** тагa.

След това добавете **canvas** към мястото, където искате да нарисувате диаграмата(в **body** частта):

**Код 2.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="myChart"></canvas>

Елементът **canvas** трябва да има уникален идентификатор.  
  
След като добавите елемента **canvas**, можете да започнете да конфигурирате диаграмата. Първо, трябва да определите контекста на елемента **canvas**. Това може да стане с помощта на следния код:

**Код 3.** Получаване на контекст на canvas

var ctx = document.getElementById('myChart').getContext('2d');

След това, можете да създадете нов обект от тип **Chart** и да конфигурирате типа на диаграмата и масивите с данни, както в следния пример:

**Код 4.** Създаване на примерна линейна диаграма с Chart.js

var myChart2 = new Chart(ctx, {

type: "line",

data: {

labels: [

"Януари",

"Февруари",

"Март",

"Април",

"Май",

"Юни",

"Юли",

],

datasets: [

{

label: "Посещения на проект Chart.js",

data: [8, 10, 10, 14, 19, 19, 15],

borderColor: "rgb(255, 99, 132)",

fill: false,

},

],

},

options: {},

});

В този пример създадохме **линейна диаграма** с масив от месеци за хоризонталната ос и масив от стойности за вертикалната ос. Допълнително, поставихме етикет и настроихме цвета на границата. След малко ще разберем повече за опциите, между които можем да избираме при създаването на нашата диаграма.  
  
А сега нека видим как изглежда горният пример:

**Фигура 1.** Примерна линейна диаграма

С помощта на **Chart.js** можете да създавате множество различни видове диаграми, включително кръгови, стълбови, графики с точки и други. Различните опции и настройки позволяват персонализиране на визуалния стил на диаграмите.

## Добри практики при използване на Chart.js

1. **Изберете правилния тип графика за визуализиране на вашите данни** - Chart.js предоставя много типове графики като линейни, стълбови, кръгови, диаграми за разпределение и т.н. Изберете този, който е най-подходящ за вашите данни и изисквания за визуализация.
2. **Подгответе данните си** - Chart.js изисква определен формат на данните за визуализация. Уверете се, че вашите данни са в правилния формат, преди да ги подадете към библиотеката.
3. **Изберете правилните настройки за графиката** - Chart.js предлага много настройки за персонализиране на графиката, като цветове, шрифтове, легенда и много други. Изберете настройките, които са най-подходящи за вашата графика и приложение.
4. **Използвайте анимацията на графиката** - Chart.js предоставя вградена анимация за графиките, която може да бъде изключена или персонализирана. Използвайте анимацията, за да направите вашата графика по-динамична и привлекателна.
5. **Използвайте събитията на графиката**- Chart.js поддържа различни събития като натискане на елемент, преместване на мишката върху елемент и други. Използвайте тези събития, за да добавите повече функционалност и интерактивност към вашата графика.
6. **Използвайте Chart.js с други библиотеки**- Chart.js може да бъде използван в комбинация с други библиотеки като jQuery и React. Използвайте тези комбинации, за да добавите още функционалност към вашата графика и да я интегрирате по-добре във вашето приложение.

## Видове диаграми

### Хистограма

Хистограмите се създават, като се зададе тип **bar** (за да обърнете посоката на баровете, задайте тип **horizontalBar**). Цветовете на лентите се задават чрез подаване на един цвят към **backgroundColor** (всички ленти ще имат един и същ цвят) или масив от цветове. Ако подавате масив (както в примера по-долу), цветовете се присвояват на етикета и номера, които имат един и същ индекс в съответните масиви. Т.е. по-долу, **София** е първият етикет, ще бъде зададен на **rgba(255, 99, 132, 0.2)** (първият цвят) и **1221172** (първото число). [**[3]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#3rd_source)

**Код 5.** Създаване на елемент canvas за Chart.js графика

<canvas id="barChart"></canvas>

**Код 6.** Създаване на примерна хистограма с Chart.js

let ctx = document.getElementById('barChart').getContext('2d');

let myChart = new Chart(ctx, {

type: 'bar',

data: {

labels: ['София', 'Пловдив', 'Варна', 'Бургас', 'Русе', 'Стара Загора'],

datasets: [{

label: 'Преброявания на населението - 2021 г.',

data: [1221172, 349070, 332686, 198035, 133813, 124599],

backgroundColor: [

'rgba(255, 99, 132, 0.2)',

'rgba(54, 162, 235, 0.2)',

'rgba(255, 206, 86, 0.2)',

'rgba(75, 192, 192, 0.2)',

'rgba(153, 102, 255, 0.2)',

'rgba(255, 159, 64, 0.2)',

],

borderColor: [

'rgba(255, 99, 132, 1)',

'rgba(54, 162, 235, 1)',

'rgba(255, 206, 86, 1)',

'rgba(75, 192, 192, 1)',

'rgba(153, 102, 255, 1)',

'rgba(255, 159, 64, 1)',

],

borderWidth: 1

}]

},

options: {

legend: { display: false },

title: {

display: true,

text: 'Преброявания на населението - 2021 г.'

}

}

});

В **options** имаме опцията да конфигурираме **legend**, която задава дали легендата на графика ще се показва или не. В този случай **display: false** означава, че легендата няма да се показва.

В **options** също можем да конфигурираме **title** като тази опция задава заглавието на графика. В случая **display: true** означава, че заглавието ще се показва, а **text: 'Преброявания на населението - 2021 г.'** е текстът на заглавието, който се изобразява.

**Фигура 2.** Примерна хистограма

### Линейна диаграма

Линейните диаграми се създават, като се зададе типът на **'line'**. Тъй като вече видяхме един по-прост пример за линейна диаграма, нека разгледаме следния пример:

**Фигура 3.** Примерна линейна диаграма

По подразбиране линиите са с тъмно прозрачно запълване, което покрива областта между линията и оста x. Смятам, че тези запълвания са склонни да затъмняват други редове, затова можем да ги премахнем чрез **fill: false**. [**[3]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#3rd_source) Диаграмата сега изглежда по следния начин:

**Фигура 4.** Примерна линейна диаграма

Нека сега разгледаме кода:

**Код 7.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="lineChart"></canvas>

**Код 8.** Създаване на примерна линейна диаграма с Chart.js

new Chart(document.getElementById("lineChart"), {

type: 'line',

data: {

labels: [1500,1600,1700,1750,1800,1850,1900,1950,1999,2050],

datasets: [{

data: [86,114,106,106,107,111,133,221,783,2478],

label: "Африка",

borderColor: "#3e95cd",

fill:false

}, {

data: [282,350,411,502,635,809,947,1402,3700,5267],

label: "Азия",

borderColor: "#8e5ea2",

fill:false

}, {

data: [168,170,178,190,203,276,408,547,675,734],

label: "Европа",

borderColor: "#3cba9f",

fill:false

}, {

data: [40,20,10,16,24,38,74,167,508,784],

label: "Латинска Америка",

borderColor: "#e8c3b9",

fill:false

}, {

data: [6,3,2,2,7,26,82,172,312,433],

label: "Северна Америка",

borderColor: "#c45850",

fill:false

}

]

},

options: {

title: {

display: true,

text: 'Световна популация за регион (в милиони)'

}

}

});

Даденият код е свързан с изобразяване на световната популация на различните региони в милиони за определен период от време. Многото данни в **data** представлява редица от числови данни за всяка една година, която се определя от масива на етикетите **(labels)** в горната част на кода. Тези числови данни показват световната популация за съответния регион за всяка една година, която е избрана в масива на етикетите. В общия случай, данните **(data)** са представени като масив от обекти, като всеки обект съдържа информация за един регион, съответния масив от данни за популацията му и визуални настройки за представянето му на диаграмата **(label,backgroundColor и fill)**.

### Диаграма на разсейване

Диаграмите на разсейване се основават на линейните диаграми, като **оста x** е променена на **линейна ос**. За да се използва диаграма на разсейване, данните трябва да се предадат като обекти, съдържащи свойства **X** и **Y**. Примерът по-долу създава диаграма на разсейване с 4 точки.[**[1]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#1st_source)

**Фигура 4.** Примерна диаграма на разсейване

Нека сега разгледаме кода:

**Код 9.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="scatterChart"></canvas>

**Код 10.** Създаване на примерна диаграма на разсейване с Chart.js

new Chart(document.getElementById("scatterChart"), {

type: 'scatter',

data: {

datasets: [{

label: 'Точки',

data: [{

x: -0.5,

y: 0.3

}, {

x: 0.1,

y: 1

}, {

x: 1,

y: 0.5

}, {

x: -0.8,

y: 0.2

}, {

x: -0.1,

y: 0.3

}, {

x: 0.9,

y: 0.05

}, {

x: 0.5,

y: 0.5

}],

backgroundColor: 'rgb(255, 99, 132)'

}],

},

options: {},

});

Задаваме **type** на диаграмата да е **"scatter"** (разсейване) . Имаме група от точки в двумерна координатна система, като във всяка точка са зададени две стойности(**x** и **y**), като **x** е позицията по хоризонталната ос (абсциса), а **y** е позицията по вертикалната ос (ордината). В диаграмата има общо седем точки, които са обозначени с различни цветове. В този конкретен случай, в опциите на диаграмата не са зададени никакви допълнителни настройки, което означава, че те са зададени по подразбиране.

### Кръгова и поничкова диаграма

Най-често използваните диаграми са кръговата и поничковата диаграма. Те са разделени на сегменти, като дъгата на всеки сегмент показва пропорционалната стойност на всяка част от данните. Отлично се справят да показват съотношенията между данните. Диаграмите **"Pie"** и **"Doughnut"** на практика са един и същ клас в Chart.js, но имат една различна стойност по подразбиране - изрязването им **(cutout)**. Това обозначава каква част от вътрешността трябва да бъде изрязана. По подразбиране е **'0%'** за кръговите диаграми и **'50%'** за поничковите. Те също така са регистрирани под два псевдонима в ядрото на библиотеката. Освен различната стойност по подразбиране и различния псевдоним, те са напълно еднакви.[**[1]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#1st_source)

**Фигура 6.** Примерна кръгова диаграма

**Фигура 7.** Примерна поничкова диаграма

**Код 11.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="pieChart"></canvas>

**Код 12.** Създаване на примерна кръгова диаграма с Chart.js

new Chart(document.getElementById("pieChart"), {

type: 'pie',

data: {

labels: ["Африка", "Азия", "Европа", "Латинска Америка", "Северна Америка"],

datasets: [{

label: "Популация (в милиони)",

backgroundColor: ["#3e95cd", "#8e5ea2","#3cba9f","#e8c3b9","#c45850"],

data: [2478,5267,734,784,433]

}]

},

options: {

title: {

display: true,

text: 'Прогнозирано световно население (в милиони) за 2050 г.'

}

}

});

Единствената разлика в кода на **Doughnut** диаграмата е именно полето **type**, където като стойност е поставено **"doughnut"**. Освен тази разлика, всичко останало в двата кода е идентично.

### Радарна диаграма

Радарните диаграми - известни още като уеб диаграми, паякови диаграми, звездни диаграми - се създават чрез задаване на **'type'** на **'radar'**. Радарните диаграми обикновено изискват повече вертикално пространство от другите графики, за да бъдат четливи, така че може да се наложи да промените пропорциите на графиката. [**[3]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#3rd_source) Ето пример за такава диаграма:

**Фигура 8.** Примерна радарна диаграма

**Код 13.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="radarChart"></canvas>

**Код 14.** Създаване на примерна радарна диаграма с Chart.js

new Chart(document.getElementById("radarChart"), {

type: 'radar',

data: {

labels: ["Африка", "Азия", "Европа", "Латинска Америка", "Северна Америка"],

datasets: [

{

label: "1950",

fill: true,

backgroundColor: "rgba(179,181,198,0.2)",

borderColor: "rgba(179,181,198,1)",

pointBorderColor: "#fff",

pointBackgroundColor: "rgba(179,181,198,1)",

data: [8.77,55.61,21.69,6.62,6.82]

}, {

label: "2050",

fill: true,

backgroundColor: "rgba(255,99,132,0.2)",

borderColor: "rgba(255,99,132,1)",

pointBorderColor: "#fff",

pointBackgroundColor: "rgba(255,99,132,1)",

pointBorderColor: "#fff",

data: [25.48,54.16,7.61,8.06,4.45]

}

]

},

options: {

title: {

display: true,

text: 'Разпределение в % от световното население'

}

}

});

В посочения код, нови опции за промяна на визуаните настройки, които са използвани са **pointBorderColor**, **pointBackgroundColor** и **pointBorderColor**. Те са параметри, които управляват визуализацията на **точките**, които се показват на графиката, като **pointBorderColor** определя цвета на рамката на точките, **pointBackgroundColor** задава цвета на фона на точките, a **pointBorderColor** определя цвета на рамката на точките.

### Полярна зонална диаграма

Полярна зонална диаграма се създава като зададем **'type'** да е **'polarArea'**. Полярните зонални диаграми са тясно свързани с кръговите диаграми, с тази разлика, че радиусът на всеки елемент е зададен в зависимост от неговата стойност. [**[3]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#3rd_source)

**Фигура 9.** Примерна полярна зонална диаграма

**Код 15.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="polarChart"></canvas>

**Код 16.** Създаване на примерна полярна зонална диаграма с Chart.js

new Chart(document.getElementById("polarChart"), {

type: 'polarArea',

data: {

labels: ["Африка", "Азия", "Европа", "Латинска Америка", "Северна Америка"],

datasets: [

{

label: "Популация (в милиони)",

backgroundColor: ["#3e95cd", "#8e5ea2","#3cba9f","#e8c3b9","#c45850"],

data: [2478,5267,734,784,433]

}

]

},

options: {

title: {

display: true,

text: 'Прогнозирано световно население (в милиони) за 2050 г.'

}

}

});

### Мехурчеста диаграма

Мехурчестите диаграми могат да бъдат чудесни за визуализиране на много различни типове от данни едновременно. Te показваt три различни данни в една диаграма - **големина на мехурчето, хоризонтална позиция** и **вертикална позиция**. Този тип диаграма се използва за сравняване на данни, като позволява на потребителя да визуализира големината на данните, както и тяхната позиция върху диаграмата.

Примерната диаграма визуализира **Брутния вътрешен продукт** (БВП), населението и продължителността на живота в четири държави: Америка, Китай, Япония и Германия. Всяка държава е представена с мехурче, като **размерът** на мехурчето представлява продължителността на живота, позицията му върху оста **X** представлява БВП, а върху оста **Y** - населението. Колкото по-голямо е мехурчето, толкова по-голяма е продължителността на живота в съответната държава. Колкото по в дясно е позиционирано мехурчето, толкова по-голям е БВП на държавата, а колкото по-наляво е позициониран, толкова по-малко е БВП. Същевременно, колкото по-нагоре е позиционирано мехурчето, толкова по-голямо е населението на държавата.

**Фигура 10.** Примерна мехурчеста диаграма

**Код 17.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="bubbleChart"></canvas>

**Код 18.** Създаване на примерна мехурчеста диаграма с Chart.js

new Chart(document.getElementById("bubbleChart"), {

type: 'bubble',

data: {

datasets: [

{

label: "Америка",

backgroundColor: "rgba(255, 99, 132, 0.5)",

borderColor: "transparent",

data: [

{

x: 19390604, // Брутен вътрешен продукт (БВП)

y: 329064917, // Население

r: 79.1, // Продължителност на живота

}

]

},

{

label: "Китай",

backgroundColor: "rgba(54, 162, 235, 0.5)",

borderColor: "transparent",

data: [

{

x: 14342932, // Брутен вътрешен продукт (БВП)

y: 1403500365, // Население

r: 76.9, // Продължителност на живота

}

]

},

{

label: "Япония",

backgroundColor: "rgba(255, 206, 86, 0.5)",

borderColor: "transparent",

data: [

{

x: 5082467, // Брутен вътрешен продукт (БВП)

y: 126860301, // Население

r: 84.6, // Продължителност на живота

}

]

},

{

label: "Германия",

backgroundColor: "rgba(75, 192, 192, 0.5)",

borderColor: "transparent",

data: [

{

x: 4029143, // Брутен вътрешен продукт (БВП)

y: 83783942, // Население

r: 80.9, // Продължителност на живота

}

]

}

]

},

options: {

title: {

display: true,

text: 'Брутен вътрешен продукт, население и продължителност на живота',

},

scales: {

yAxes: [{

scaleLabel: {

display: true,

labelString: 'Популация'

},

ticks: {

min: -600000000, // минимална стойност за оста у

max: 2000000000, // максимална стойност за оста у

stepSize: 200000000 // стъпка между стойностите на оста у

}

}],

xAxes: [{

scaleLabel: {

display: true,

labelString: 'БВП (щ.д.)'

},

ticks: {

min: 2240000, // минимална стойност за оста x

max: 21170000, // максимална стойност за оста x

stepSize: 10000 // стъпка между стойностите на оста x

}

}],

}

}

});

В **options** на тази диаграма, имаме обект **scales**, който позволява да се настроят различни мащабиращи оста параметри за **x** и **y** координатите на графиката. В общия случай, тези параметри са настройки на границите и стъпките на мащабиране, за да се осигури правилната визуализация на данните. В обекта **scales**, опциите за оста на **y** координатата могат да се конфигурират с **yAxes**, докато за оста на **x** координатата - с **xAxes**. В този код, се използва още **scaleLabel** за да се добавят метаданни към остите - това е текстов етикет, който показва каква информация представлява дадената ос.  
Използваните опции за мащабиране са следните:

* **ticks** - конфигурира мащабирането на всяка от стойностите на оста. Тук може да се зададе минимална и максимална стойност, както и стъпката между тях.
  + **min** и **max** - тези параметри задават минималната и максималната стойност на оста.
  + **stepSize** - това е стъпката между стойностите на остата.

Обикновено мащабирането се прави автоматично, но с **scales** можем да променяме тези настройки, за да се получи желаното представяне на данните.

### Смесена диаграма

Смесените диаграми в **Chart.js** успява да комбинира различни видове диаграми (като линейни, стълбови и кръгови диаграми) в една диаграма. Това позволява на потребителя да показва няколко вида данни на една и съща диаграма и може да бъде полезно в ситуации, в които трябва да се сравняват или анализират заедно различни видове данни. Ето и един пример за смесена диаграма обединяваща **линейна диаграма**, показваща тенденцията на броя на посетителите на уебсайта във времето, заедно с **хистограма**, показваща броя на покупките, направени на уебсайта през същия период.

**Фигура 11.** Примерна смесена диаграма

**Код 19.** Създаване на елемент canvas за Chart.js диаграма

<canvas id="mixedChart"></canvas>

**Код 20.** Създаване на примерна смесена диаграма с Chart.js

new Chart(document.getElementById("mixedChart"), {

type: 'bar',

data: {

labels: ["Януари", "Февруари", "Март", "Април", "Май", "Юни", "Юли"],

datasets: [{

label: 'Посетители',

type:'line',

data: [500, 750, 670, 900, 1200, 1050, 1020],

borderColor: 'rgba(255, 99, 132, 1)',

borderWidth: 2

}, {

label: 'Поръчки',

type:'bar',

data: [50, 75, 58, 125, 80, 160, 180],

backgroundColor: 'rgba(54, 162, 235, 1)',

borderWidth: 1

}]

},

options: {

responsive: true,

title: {

display: true,

text: 'Брой посетители на уебсайт и покупки'

},

scales: {

yAxes: [{

ticks: {

beginAtZero: true

}

}]

}

}

});

**Options** обектът представлява настройките на диаграмата. Този обект може да съдържа много различни настройки, като заглавие на диаграмата, мащаби на осите, цветове, форматиране на линии, кутии и т.н.  
В конкретния пример, обектът options съдържа следните настройки:

* **responsive: true** - тази настройка позволява на диаграмата да се реагира на промяната на размера на контейнера, в който се намира. Това ще каже, че когато размерът на контейнера се промени, диаграмата ще се промени автоматично, за да се адаптира към новия размер.
* **title: {display: true, text: 'Брой посетители на уебсайт и покупки'}** - тази настройка задава заглавието на диаграмата. Параметърът **display** определя дали заглавието трябва да бъде показано, а параметърът **text** задава самото заглавие.
* **scales: {yAxes: [{ticks: {beginAtZero: true}}]}** - тази настройка определя мащабите на y-оса на диаграмата. **yAxes** указва, че се използва само една y-ос, а **ticks** определя началото и края на оса. **beginAtZero: true** гарантира, че y-осата ще започне от нулата.

**Options** обектът е много важен за кастъмизирането на визуализацията на данните в диаграмата. Той позволява на потребителя да променя външния вид и функционалността на диаграмата според нуждите и изискванията на конкретния проект.

## Предимства и недостатъци

### Предимства

Chart.js предоставя на потребителите много възможности. А ето и някои от тях:

* Шест различни анимирани изгледа за визуализиране на данните.
* Възможности за персонализиране, както и разширения за интерактивност.
* различни видове интерактивни диаграми за показване на данни.
* Използването му е напълно безплатно.
* Aктуализирането, премахването и модифицирането на данни от диаграма е лесно и безпроблемно.
* Благодарение на простата си структура Chart. js може да съхранява максимален брой данни в минимално пространство.
* Диаграмите са напълно адаптивни.
* Много плъгини са достъпни за използване чрез NPM и дори можете да пишете и собствени плъгини.

### Недостатъци

Освен многобройните предимства, Chart.js все пак има и следните недостатъци:

* Chart. js показва само подобен тип данни с вариации.
* Инсталацията на Chart.js поддържа показване само на графики и диаграми. Липсва му гъвкавостта, предлагана от други опции.
* Елементът canvas е базиран на растерно изображение. Той има същите проблеми като невекторните формати. [**[7]**](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#7th_source)

## Заключение

Chart.js е една от най-добрите библиотеки за визуализация на данни на пазара, която може да бъде използвана от уеб разработчици на всички нива. Тя предоставя бърз и лесен начин за създаване на интерактивни графики и предоставя много опции за персонализиране на външния вид и функционалността на графиките.  
  
Освен това, тази библиотека е с отворен код и има голяма общност от разработчици, които активно я поддържат и разширяват. Това означава, че съществува голям избор от документация, ръководства и примери, които могат да бъдат полезни при работата с нея.  
  
Също така, Chart.js е съвместима със всички модерни уеб браузъри и може да бъде интегрирана лесно във всеки проект. Ако искате да добавите визуални елементи към вашето уеб приложение, Chart.js може да бъде един от най-добрите инструменти, които можете да използвате.

## Използвани източници:

[1] [**Chart.js**](https://www.chartjs.org/), "Simple yet flexible JavaScript charting for designers & developers", последно посетен на 2023-04-17.

[2] [**W3Schools**](https://www.w3schools.com/ai/ai_chartjs.asp), "AI - Chart.js", последно посетен на 2023-04-17.

[3] [**Tobias Ahlin**](https://tobiasahlin.com/blog/chartjs-charts-to-get-you-started), "10 Chart.js example charts to get you started", последно посетен на 2023-04-17.

[4] [**Keenthemes Start HTML Chart.js Documentation**](https://preview.keenthemes.com/start-html-free/documentation/charts/chartjs.html), "Documentation on how to use Chart.js, a flexible JavaScript charting library for designers and developers", последно посетен на 2023-05-01.

[5] [**CoreUI React Chart.js Component Documentation**](https://coreui.io/react/docs/components/chart/), "Documentation on how to use the Chart.js component within the CoreUI React library", последно посетен на 2023-05-01.

[6] [**Chart.js Wikipedia page**](https://en.wikipedia.org/wiki/Chart.js), "A free and open-source JavaScript library for data visualization, particularly for charts on websites", последно посетен на 2023-05-01.

[7] [**Chart.js Tutorialspoint page**](https://www.tutorialspoint.com/chartjs/index.htm), "A comprehensive tutorial on Chart.js, a JavaScript library for creating interactive and customizable charts and graphs", последно посетен на 2023-05-01.

## Списък с фигури:

[**Фигура 1.** Примерна линейна диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_1)

[**Фигура 2.** Примерна хистограма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_2)

[**Фигура 3.** Примерна линейна диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_3)

[**Фигура 4.** Примерна линейна диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_4)

[**Фигура 5.** Примерна диаграма на разсейване](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_5)

[**Фигура 6.** Примерна кръгова диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_6)

[**Фигура 7.** Примерна поничкова диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_7)

[**Фигура 8.** Примерна радарна диаграм](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_8)

[**Фигура 9.** Примерна полярна зонална диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_9)

[**Фигура 10.** Примерна мехурчеста диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_10)

[**Фигура 11.** Примерна смесена диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#figure_11)

## Списък с кодове:

[**Код 1.** Включване на Chart.js библиотеката чрез връзка към CDN](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_1)

[**Код 2.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_2)

[**Код 3.** Получаване на контекст на canvas](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_3)

[**Код 4.** Създаване на примерна линейна диаграма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_4)

[**Код 5.** Създаване на елемент canvas за Chart.js графика](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_5)

[**Код 6.** Създаване на примерна хистограма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_6)

[**Код 7.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_7)

[**Код 8.** Създаване на примерна линейна диаграма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_8)

[**Код 9.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_9)

[**Код 10.** Създаване на примерна диаграма на разсейване с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_10)

[**Код 11.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_11)

[**Код 12.** Създаване на примерна кръгова диаграма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_12)

[**Код 13.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_13)

[**Код 14.** Създаване на примерна радарна диаграма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_14)

[**Код 15.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_15)

[**Код 16.** Създаване на примерна полярна зонална диаграма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_16)

[**Код 17.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_17)

[**Код 18.** Създаване на примерна мехурчеста диаграма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_18)

[**Код 19.** Създаване на елемент canvas за Chart.js диаграма](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_19)

[**Код 20.** Създаване на примерна смесена диаграма с Chart.js](https://w20ref.w3c.fmi.uni-sofia.bg/referat/85309_5_115/index.php?action=view&back=review#code_20)

**Изготвен от:** Йорданка Стоянова

**Последна модификация:** Май 10 2023 г.