# Упражнения: Прости пресмятания

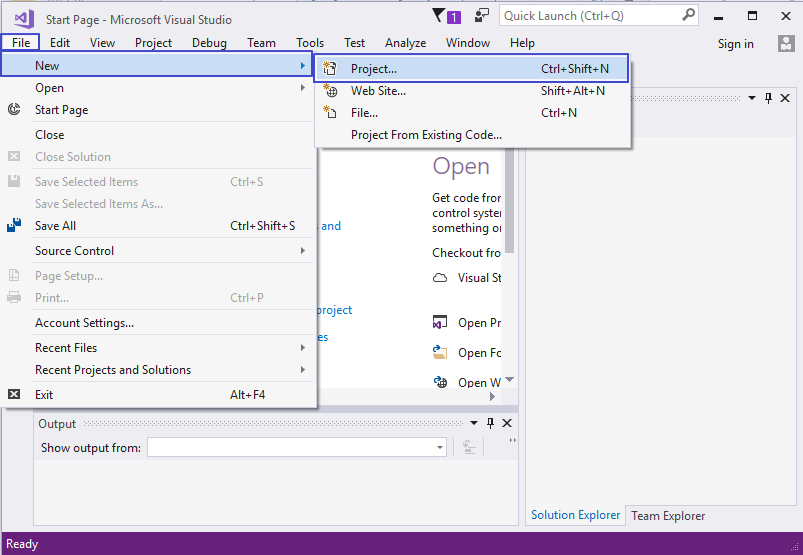
Задачи за упражнение в клас и за домашно към курса [„Основи на програмирането“ @ СофтУни](https://softuni.bg/courses/programming-basics).

## Празно Visual Studio решение (Blank Solution)

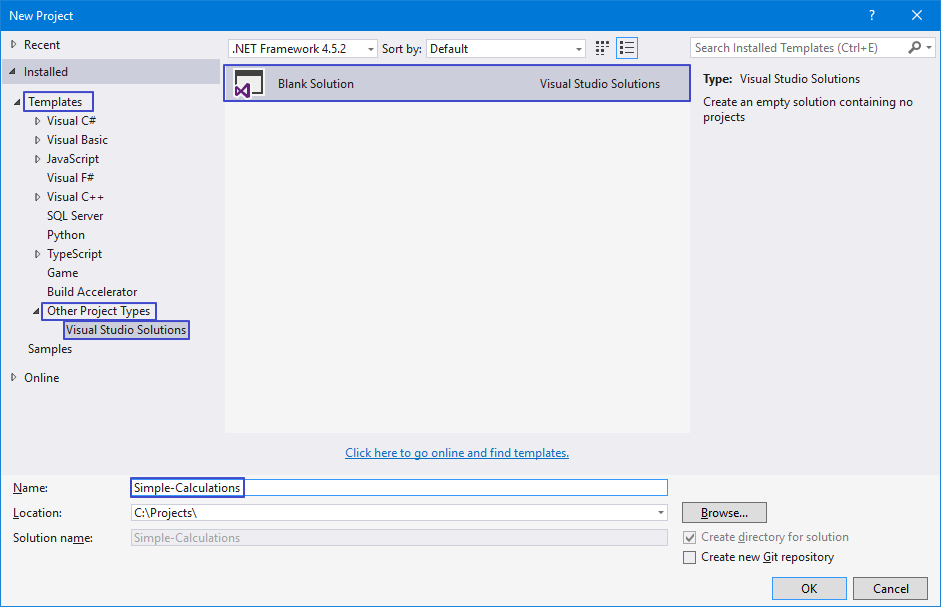
Създайте празно решение (**Blank Solution**) във Visual Studio. Решенията (solutions) във Visual Studio обединяват **група проекти**. Тази възможност е изключително удобна, когато искаме да работим по няколко проекта и бързо да превключваме между тях или искаме да обединим логически няколко взаимосвързани проекта.

В настоящото практическо занимание ще използваме **Blank Solution с няколко проекта** за да организираме решенията на задачите от упражненията – всяка задача в отделен проект и всички проекти в общ solution.

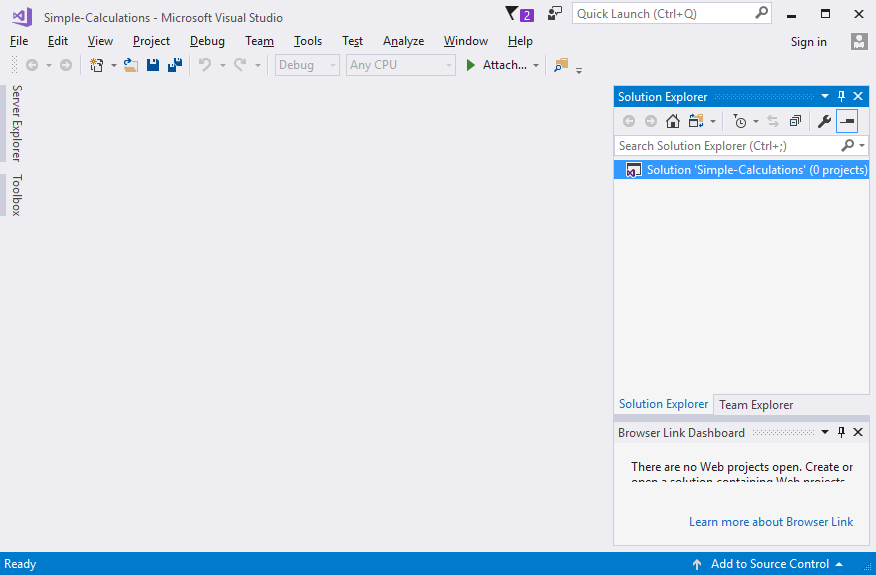
1. Стартирайте Visual Studio.
2. Създайте нов **Blank Solution**: [File]🡪 [New] 🡪 [Project].



1. Изберете от диалоговия прозорец [Templates] 🡪 [Other Project Types] 🡪 [Visual Studio Solutions] 🡪 [**Blank Solution**] и дайте подходящо име на проекта, например “Simple-Calculations”:



Сега имате създаден **празен Visual Studio Solution** (с 0 проекта в него):

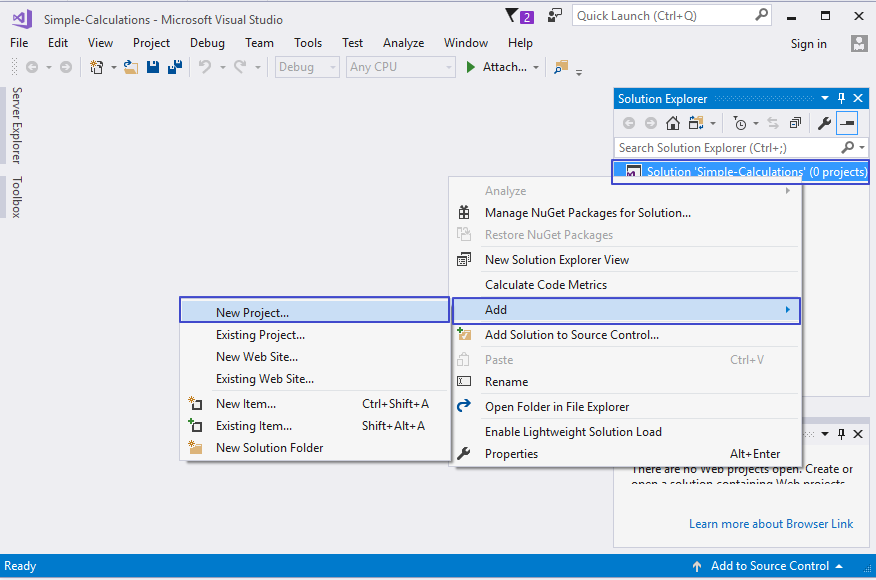


Целта на този blank solution e да добавяте в него **по един проект за всяка задача** от упражненията.

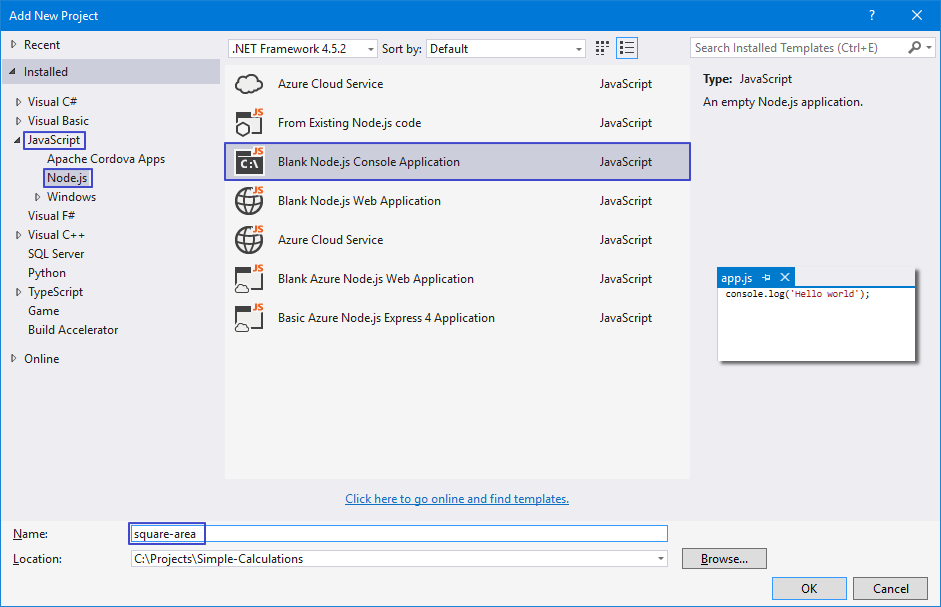
## Пресмятане на лице на квадрат

Първата задача от тази тема е следната: да се напише **конзолна програма**, която **прочита цяло число** „a“, въведено от потребителя, и **пресмята лицето на квадрат** **със страна** „a“. Задачата е тривиално лесна: въвеждате число, умножавате го само по себе си и печатате получения резултат на конзолата.

1. Създайте **нов проект** в съществуващото Visual Studio решение. В Solution Explorer кликнете с десен бутон на мишката върху **Solution 'Simple-Calculations'**. Изберете [Add] 🡪 [New Project…]:



1. Ще се отвори диалогов прозорец за избор на тип проект за създаване. Изберете Node.js конзолно приложение с име “square-area”:



Вече имате solution с едно конзолно приложение в него. Остава да напишете кода за решаване на задачата.

1. Копирайте скелета на функцията squareArea():

function squareArea([arg1]) {

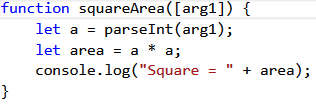
//Your code goes here.

}

// Call the function to execute the code inside

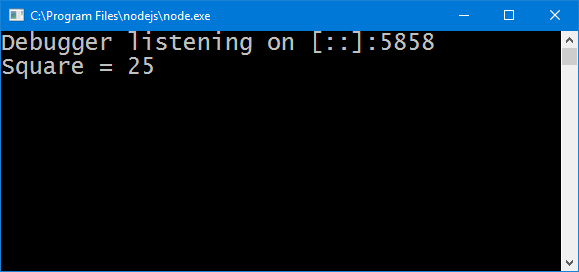
squareArea(["5"]);

1. Отидете в тялото на фунцкията squareArea([arg1]) и напишете кода от картинката по-долу:

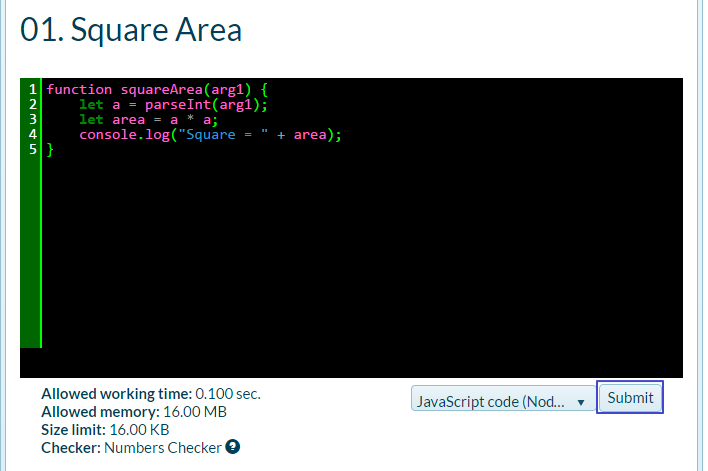


Кодът прочита цяло число с a = parseInt(arg1), след това изчислява area = a \* a и накрая печата стойността на променливата area.

1. **Стартирайте** програмата с [F5] и я **тествайте** с различни входни стойности:



1. **Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#0>. Сменете стратегията на judge с **"JavaScript code(NodeJS)"**. Трябва да получите 100 точки (напълно коректно решение):

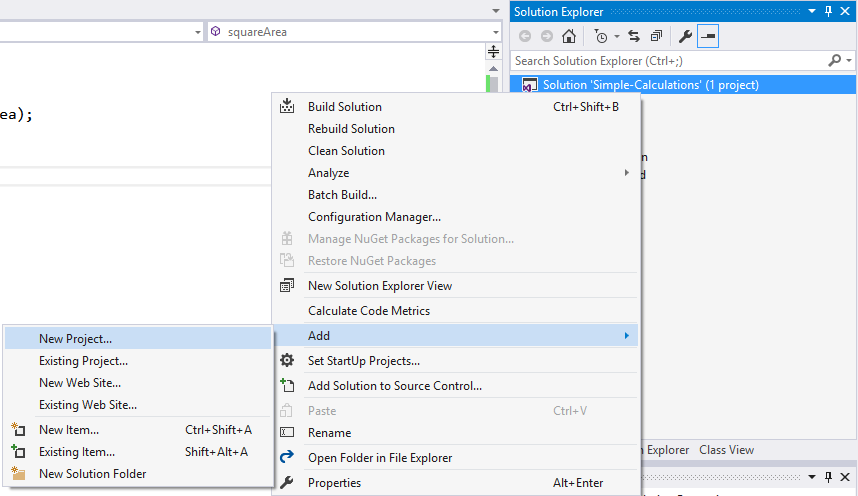




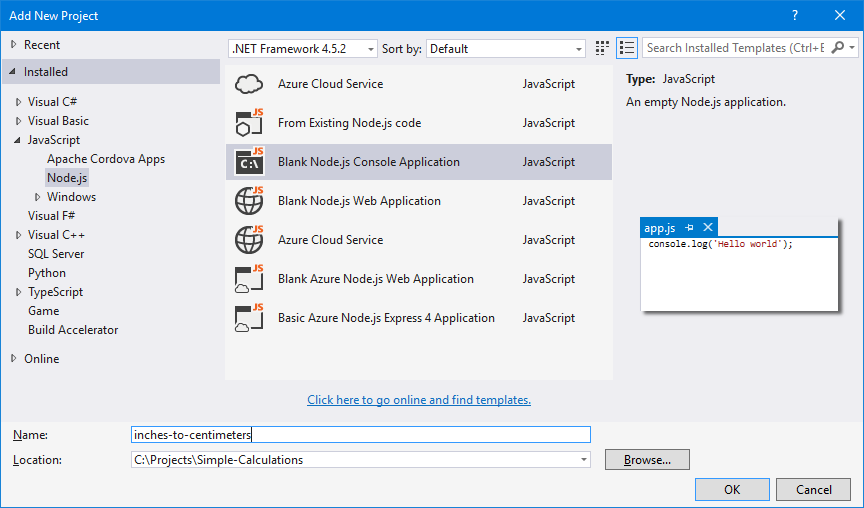
## От инчове към сантиметри

Да се напише програма, която **чете число** (не непременно цяло), подадено от потребителя, и преобразува числото **от инчове в сантиметри**. За целта **умножава инчовете по 2.54** (защото 1 инч = 2.54 сантиметра).

1. Първо създайте **нов Node.js конзолен проект** в решението “**Simple-Calculations**”. Кликнете с мишката върху решението в Solution Explorer и изберете [Add] 🡪 [New Project…]:



Изберете [JavaScript] 🡪 [Node.js] 🡪 [Blank Node.js Console Application] и задайте име “inches-to-centimeters”:



1. Копирайте скелета на функцията inchesToCentimeters():

function inchesToCentimeters([arg1]) {

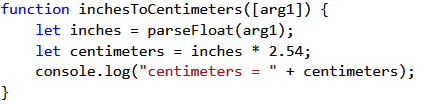
//Your code goes here.

}

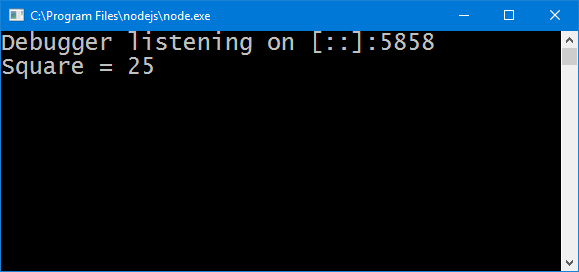
// Call the function to execute the code inside

inchesToCentimeters(["5"]);

1. **Напишете кода** на програмата. Може да си помогнете с примерния код от картинката:

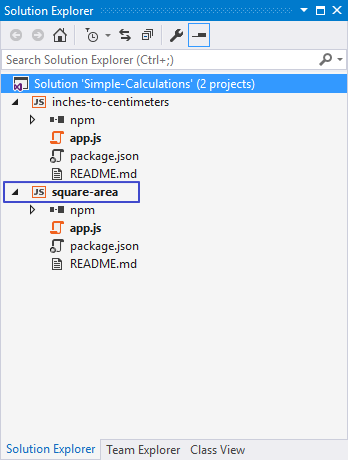


1. **Стартирайте програмата**, както обикновено с [F5]:

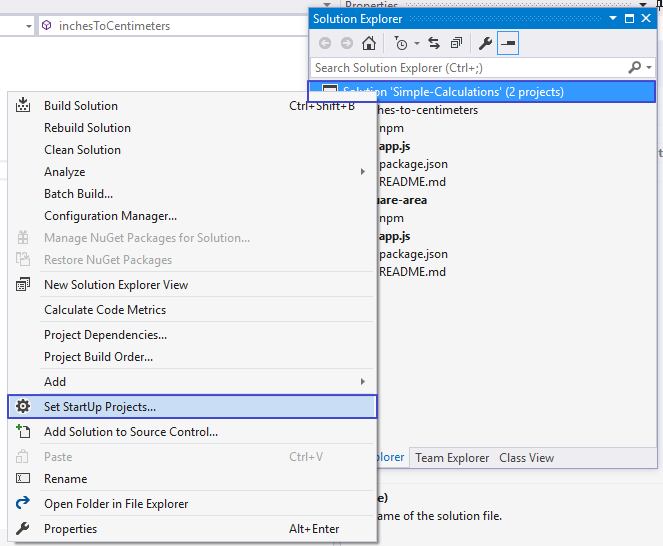


Изненада! Како става? Програмата не работи правилно… Всъщност това не е ли предходната програма?

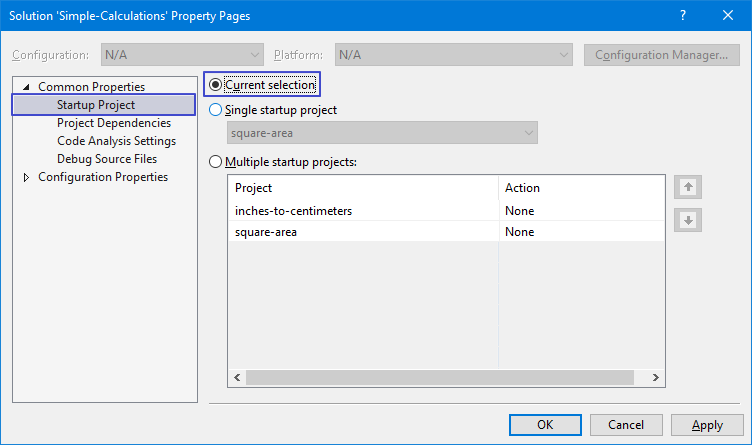
Във Visual Studio **текущият активен проект** в един solution е маркиран в получерно и може да се сменя:



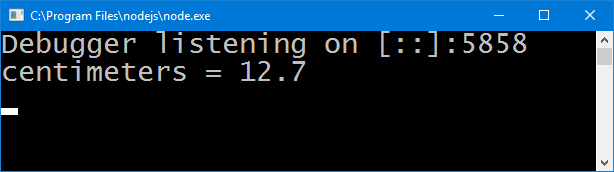
1. За да включите режим на **автоматично преминаване към текущия проект**, кликнете върху главния solution с десния бутон на мишката и изберете **[Set StartUp Projects…]**:



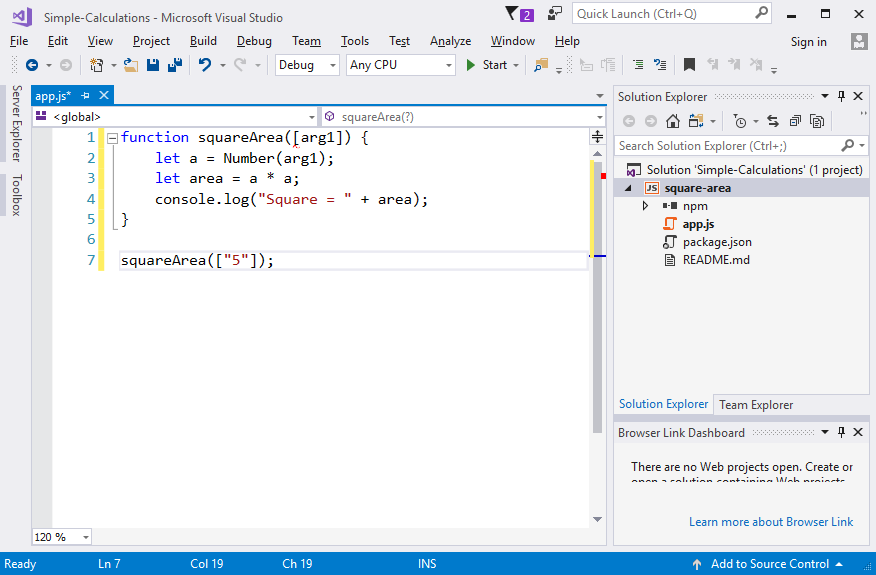
Ще се появи диалогов прозорец, от който трябва да се избере **[Startup Project]** 🡪 [**Current selection**]:



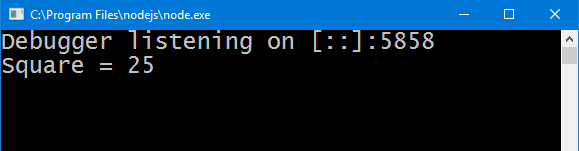
1. Сега отново **стартирайте програмата**, както обикновено с [F5]. Този път ще се стартира текущата отворена програма, която преобразува инчове в сантиметри. Изглежда работи коректно:



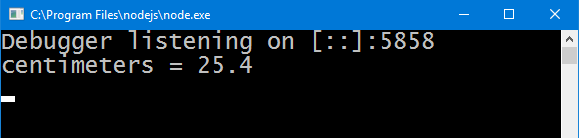
1. Сега **превключете към предходната програма** (лице на квадрат). Това става с ляв клик на мишката върху файла app.js от предходния проект “square-area” в панела [Solution Explorer] на Visual Studio:



1. Натиснете пак **[F5]**. Този път трябва да се стартира другият проект:

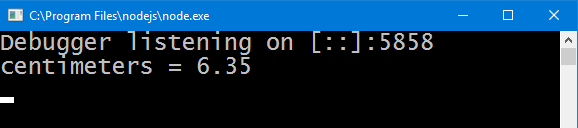


1. Превключете обратно към проекта “inches-to-centimeters” и го стартирайте с **[F5]**:

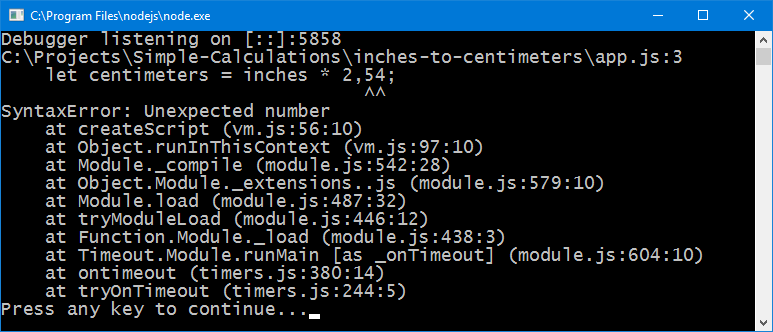


**Превключването между проектите** е много лесно, нали? Просто избираме файла със сорс кода на програмата, кликваме го с мишката и при стартиране тръгва програмата от този файл.

1. Тествайте с **дробни числа**, например с 2.5:



**Внимание:** в зависимост от регионалните настройки на операционната система, е възможно вместо **десетична точка** (US настройки) да се използва **десетична запетая** (BG настройки). Ако програмата очаква десетична точка и бъде въведено число с десетична запетая или на обратно (бъде въведена десетична точка когато се очаква десетична запетая), ще се получи следната грешка:



Препоръчително е **да промените настройките на компютъра си**, така че да се използва **десетична точка**:





1. Вече е време за **тестване в judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#1>. Решението би трябвало да бъде прието като напълно коректно:



## Поздрав по име

Да се напише програма, която **чете име на човек**, подадено от потребителя, и отпечатва “Hello, <name>!”, където <name> е въведеното преди това име.

1. Първо създайте **нов Node.js конзолен проект** с име “greeting” в решението “Simple-Calculations” и използвайте скелета на функцията greeting():

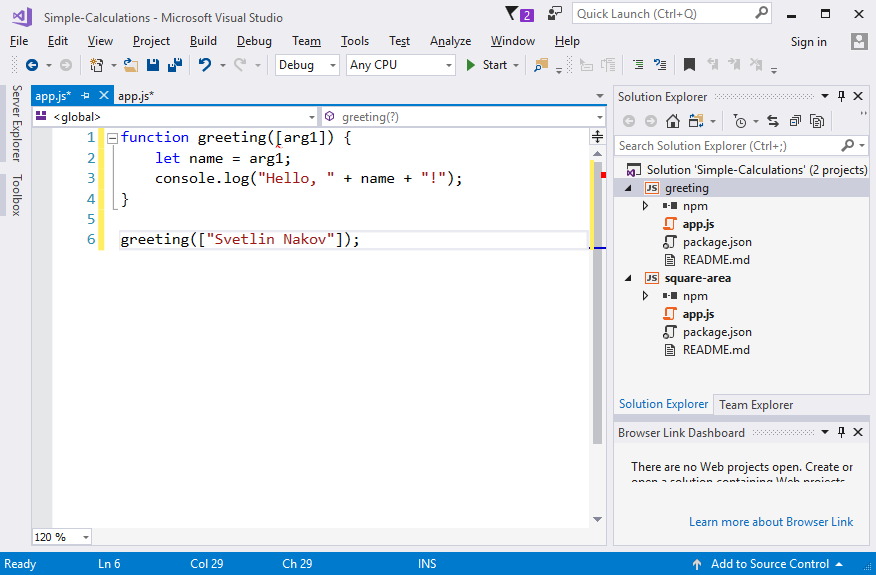
function greeting([arg1]) {

//Your code goes here.

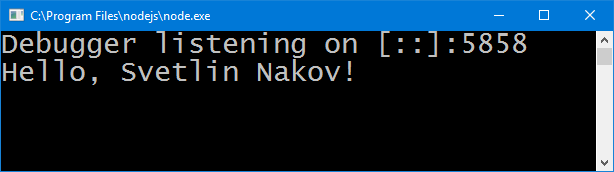
}

greeting(["Svetlin Nakov"]);

1. **Напишете кода** на програмата. Ако се затруднявате, може да ползвате примерния код по-долу:



1. **Стартирайте** програмата с **[F5]** и я тествайте:



1. Тествайте в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#2>.

## Съединяване на текст и числа

Напишете JavaScript програма, която прочита от конзолата име, фамилия, възраст и град, въведени от потребителя като аргументи, и печата съобщение от следния вид: “You are <firstName> <lastName>, a <age>-years old person from <town>”.

1. Добавете към текущото Visual Studio решение още един **конзолен Node.js проект** с име “concatenate-data”.
2. Използвайте скелета, в който да напишете кода си:

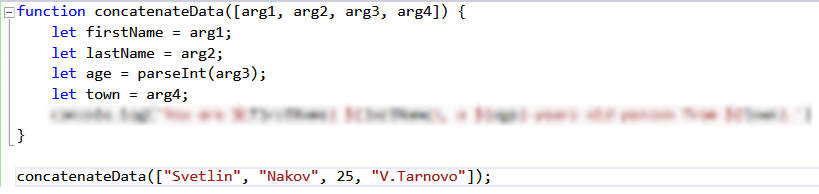
function concatenateData([arg1, arg2, arg3, arg4]) {

//Your code goes here.

}

concatenateData(["Svetlin", "Nakov", "25", "V. Tarnovo"]);

1. **Напишете кода**, който чете входните данни:



1. **Допишете код**, който отпечатва описаното в условието на задачата съобщение.

На горната картинка кодът е нарочно даден размазан, за да помислите как да си го напишете сами.

1. **Тествайте** решението локално с [F5] и въвеждане на примерни данни.
2. Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#3>.

## Лице на трапец

Напишете програма, която чете три числа b1, b2 и h, подадени от потребителя като аргументи, и **пресмята лицето на трапец** с основи b1 и b2 и височина h. **Формулата за лице на трапец** е (b1 + b2) \* h / 2.

На фигурата по-долу е показан трапец със страни 8 и 13 и височина 7. Той има лице (8 + 13) \* 7 / 2 = 73.5.



1. Добавете към текущото Visual Studio решение още един **конзолен Node.js проект** с име “trapezoid-area”.
2. Използвайте скелета по-долу, в който да напишете кода си:

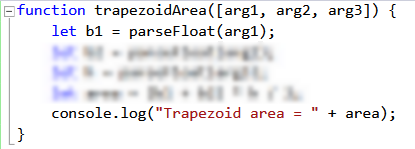
function trapezoidArea([arg1, arg2, arg3]) {

//Your code goes here.

}

trapezoidArea(["8", "13", "7"]);

1. **Напишете кода**, който чете входните данни, пресмята лицето на трапеца и го отпечатва:



Кодът на картинката е нарочно размазан, за да си го доизмислите и допишете сами.

1. **Тествайте** решението локално с [F5] и въвеждане на примерни данни.
2. Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#4>.

## Периметър и лице на кръг

Напишете програма, която чете **число** r(радиус), подадено от потребителя, и пресмята и отпечатва **лицето** и **периметъра на кръг** / окръжност с радиус r.

|  |  |
| --- | --- |
| **вход** | **изход** |
| 3 | Area = 28.274333882308138  Perimeter = 18.84955592153876 |
| 4.5 | Area = 63.61725123519331  Perimeter = 28.274333882308138 |

За изчисленията можете да използвате следните формули:

* area = Math.PI \* r \* r
* perimeter = 2 \* Math.PI \* r

Използвайте следния код:

function circleAreaPerimeter([arg1]) {

// Your code goes here

}

// Call the function to execute the code inside

circleAreaPerimeter(["3"]);

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#5>.



## Лице на правоъгълник в равнината

**Правоъгълник** е зададен с **координатите** на два от своите срещуположни ъгъла (x1, y1) – (x2, y2). Да се пресметнат **площта** и **периметъра** му. **Входът** се въвежда от потребителя. Числата x1, y1, x2 и y2 са дадени по едно на ред. **Изходът** се извежда на конзолата и трябва да съдържа два реда с по едно число на всеки от тях – лицето и периметъра.

|  |  |
| --- | --- |
| **вход** | **изход** |
| 60  20  10  50 | 1500  160 |
| 30  40  70  -10 | 2000  180 |
| 600.25  500.75  100.50  -200.5 | 350449.6875  2402 |

Използвайте следния код:

function rectangleAreaParameter([arg1, arg2, arg3, arg4]) {

// Your code goes here

}

// Call the function to execute the code inside

rectangleAreaParameter(["60", "20", "10", "50"]);

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#6>.

## Лице на триъгълник

Напишете програма, която чете **страна** и **височина** на **триъгълник**, подадени от потребителя като аргументи,и пресмята неговото лице. Използвайте **формулата** за лице на триъгълник: area = a \* h / 2. Закръглете резултата до **2 знака след десетичната точка** използвайки метода [**toFixed()**](https://www.w3schools.com/jsref/jsref_tofixed.asp).

|  |  |
| --- | --- |
| **вход** | **изход** |
| 20  30 | Triangle area = 300.00 |
| 15  35 | Triangle area = 262.50 |
| 7.75  8.45 | Triangle area = 32.74 |
| 1.23456  4.56789 | Triangle area = 2.82 |

Използвайте следния код:

function triangleArea([arg1, arg2]) {

// Your code goes here

}

// Call the function to execute the code inside

triangleArea(["20", "30"]);

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#7>.

## Конвертор: от градуси °C към градуси °F

Напишете програма, която чете **градуси по скалата на Целзий** (°C), въведени от потребителя като аргумент, и ги преобразува до **градуси по скалата на Фаренхайт** (°F). Потърсете в Интернет подходяща [формула](http://bfy.tw/3rGh), с която да извършите изчисленията. Закръглете резултата до **2 знака след десетичната точка**. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 25 | 77 | 0 | 32 | -5.5 | 22.1 | 32.3 | 90.14 |

Използвайте следния код:

function celsiusToFahrenheit([arg1]) {

// Your code goes here

}

// Call the function to execute the code inside

celsiusToFahrenheit(["25"]);

Използвайте формулата: Math.round(num \* 100) / 100,за да закръглите до 2 знака след десетичната точка.

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#8>.

## Конвертор: от радиани в градуси

Напишете програма, която чете **ъгъл в [радиани](https://bg.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D0%B8%D0%B0%D0%BD)** (rad), въведен от потребителя, и го преобразува в **[градуси](https://bg.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D1%83%D1%81_(%D1%8A%D0%B3%D1%8A%D0%BB))** (deg). Потърсете в Интернет подходяща формула. Числото **π** в JavaScript програми е достъпно чрез Math.PI. Закръглете резултата до най-близкото цяло число използвайки Math.round(). Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3.1416 | 180 | 6.2832 | 360 | 0.7854 | 45 | 0.5236 | 30 |

function radiansToDegrees([arg1]) {

// Your code goes here

}

// Call the function to execute the code inside

radiansToDegrees(["3.1416"]);

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#9>.

## Конвертор: USD към BGN

Напишете програма за **конвертиране на щатски долари** (USD) **в български лева** (BGN). **Закръглете** резултата до **2 цифри** след десетичната запетая. Използвайте фиксиран **курс** между долар и лев: **1 USD** = **1.79549 BGN**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **Изход** |
| 20 | 35.91 BGN | 100 | 179.55 BGN | 12.5 | 22.44 BGN |

Използвайте следния код:

function usdToBGN([arg1]) {

// Your code goes here

}

// Call the function to execute the code inside

usdToBGN(["20"]);

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#10>.

## \* Междувалутен конвертор

Напишете програма за **конвертиране на парична сума от една валута в друга**. Трябва да се поддържат следните валути: **BGN**, **USD**, **EUR**, **GBP**. Използвайте следните фиксирани валутни курсове:

|  |  |  |  |
| --- | --- | --- | --- |
| **Курс** | **USD** | **EUR** | **GBP** |
| **1 BGN** | 1.79549 | 1.95583 | 2.53405 |

**Входът** e **сума за конвертиране**, **входна валута** и **изходна валута**, въведени от потребителя. **Изходът** е едно число – преобразуваната сума по посочените по-горе курсове, закръглен до **2 цифри** след десетичната точка. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **Изход** |  | **вход** | **изход** |
| 20  USD  BGN | 35.91 BGN | 100  BGN  EUR | 51.13 EUR | 12.35  EUR  GBP | 9.53 GBP | 150.35  USD  EUR | 138.02 EUR |

Използвайе следния код:

function currencyConverter([arg1, arg2, arg3]) {

// Your code goes here

}

// Call the function to execute the code inside

currencyConverter(["20", "USD", "BGN"]);

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#11>.

## \*\* Пресмятане с дати: 1000 дни на Земята

Напишете програма, която чете **рождена дата** във формат “dd-MM-yyyy”, въведена от потребителя, и пресмята датата, на която се навършват **1000 дни** от тази рождена дата и я отпечатва в същия формат.

|  |  |
| --- | --- |
| **вход** | **изход** |
| 25-02-1995 | 20-11-1997 |
| 07-11-2003 | 02-08-2006 |
| 30-12-2002 | 24-09-2005 |
| 01-01-2012 | 26-09-2014 |
| 14-06-1980 | 10-03-1983 |

**\* Подсказки**: Обработване на date string: http://stackoverflow.com/questions/7151543/convert-dd-mm-yyyy-string-to-date

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#12>. Не печатайте нищо допълнително на конзолата освен изискваната дата!

# Упражнения: Web и Desktop приложения

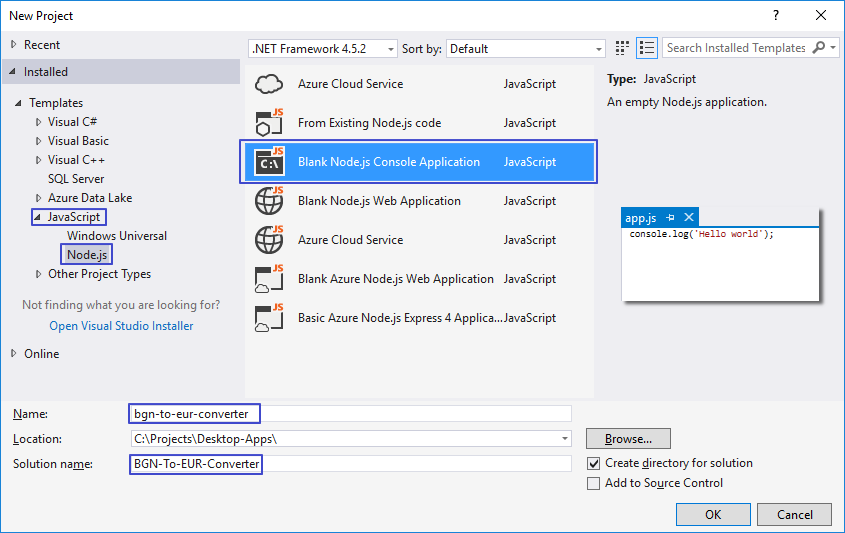
## Графично приложение: конвертор от BGN към EUR

Създайте **графично приложение**, което пресмята стойността в **евро** (EUR) на парична сума, зададена в **лева** (BGN). При промяна на стойността в лева, равностойността в евро трябва да се преизчислява автоматично. Използвайте курс лева / евро: 1.95583.



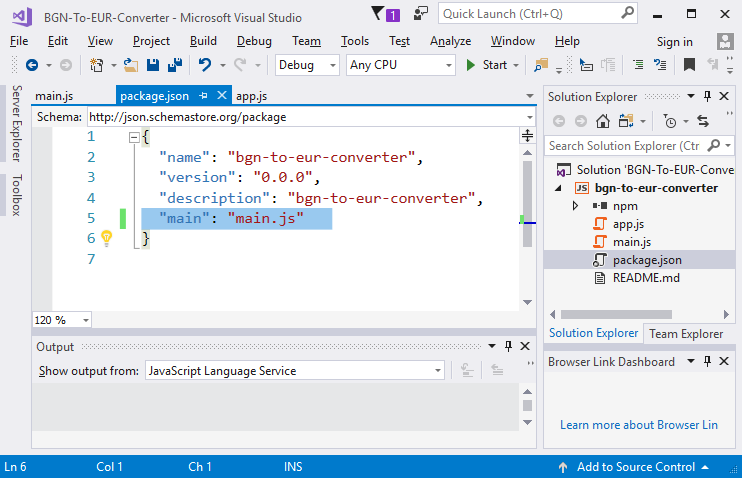
За нашето приложение ще използваме технологията **Electron.js**, която позволява създаване на графични приложения за всички операционни системи с езика за програмиране **JavaScript, HTML и CSS**.

1. Създайте нов Visual Studio проект. Изберете от диалоговия прозорец [Templates] 🡪 [JavaScript] 🡪 [Node.js] 🡪 [**Blank Node.js Console Application**] и дайте подходящо име на проекта, например “BGN-To-EUR-Converter”:



В случай, че не сте инсталирали Electron.js, преминете през стъпки **2.** и **3.** от Упражнение "**1. First steps in coding**", задача "**7. Десктоп приложение "Суматор за числа"** ".

1. Отворете файла "package.json", който съдържа настройките на проекта и променете името на стартовия скриптов файл с "main.js" :



1. Кодът, описан в "main.js", управлява събитията и създава нови прозорци в приложението. Трябва да изглежда по следния начин:

const {app, BrowserWindow} = require('electron');

const path = require('path');

const url = require('url');

let win;

function createWindow () {

win = new BrowserWindow({width: 750, height: 350, resizable: false});

win.loadURL(url.format({

pathname: path.join(\_\_dirname, 'index.html'),

protocol: 'file:',

slashes: true

}));

win.on('closed', ()=>{

win = null;

});

}

app.on('ready', createWindow);

app.on('window-all-closed', () => {

app.quit()

})

app.on('activate', () => {

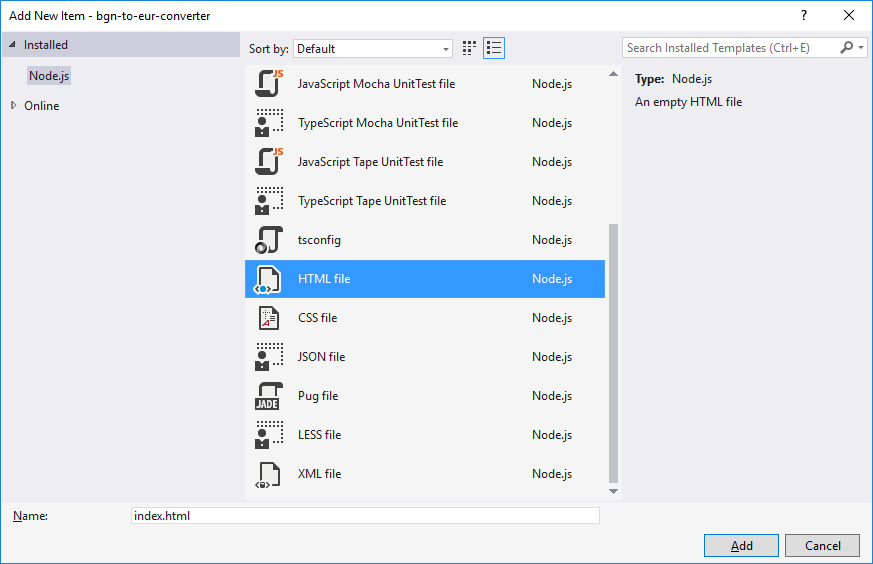
if (win === null) {

createWindow()

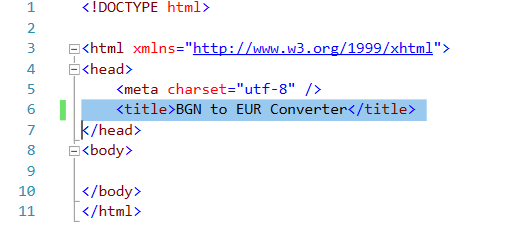
}

})

1. Създайте нов \*.html файл с име "index.html":



Тагът "**<title>**" е задължителен за всеки html документ и дефинира заглавието му. Влезте в него и напишете "**BGN to EUR Converter**":



Преименувайте файла "app.js" на "converter.js" и добавете следния код под тага <title> в "index.html":

<script src="converter.js" type="text/javascript"></script>

По този начин се осъществява връзката между файловете **index.html** и **converter.js**.

Тагът "**<body>**" дефинира тялото на html документа. Напишете в него следния код:

1. Отворете **cmd** с десен клик на мишката върху проекта и стартирайте electron с командата "**electron .**" :

<body>

<form class="content-form">

<h2 class="title">BGN to EUR Converter</h2>

<section class="items">

<label for="bgn" class="currency">

<span class="item-currency">BGN: </span>

<input class="currency-value" type="number" id="bgn" value="0" />

</label>

<label for="euro" class="currency">

<span class="item-currency">EUR: </span>

<input class="currency-value" type="text" id="euro" readonly />

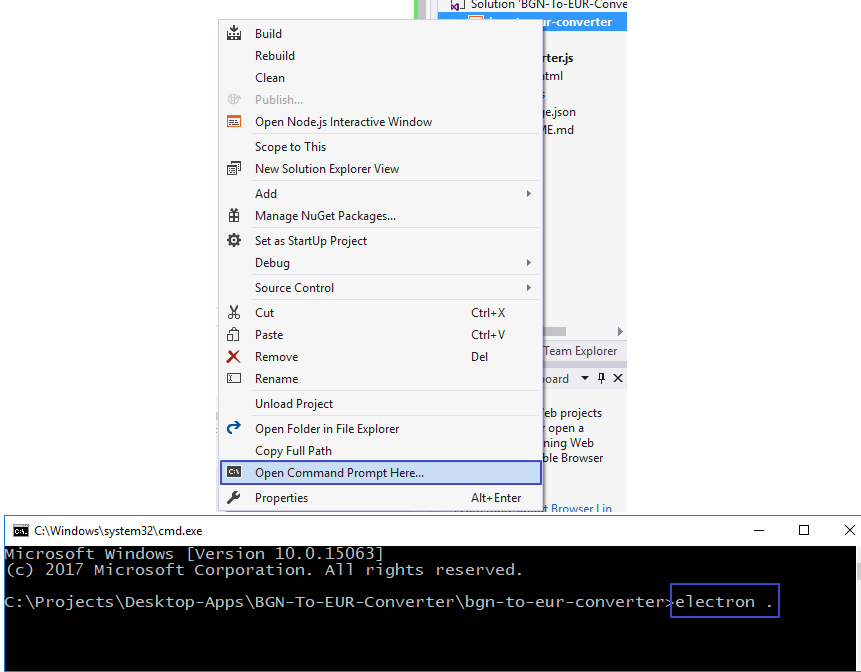
</label>

<input class="primary-btn" type="button" value="Convert!" />

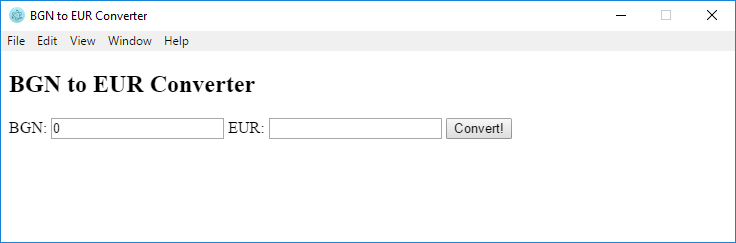
</section>

</form>

</body>



Приложението би трябвало да изглежда така:



При натискане на бутона "Convert!" не се случва нищо, тъй като все още няма функция, която да изпълнява пресмятането. За целта отворете файла "converter.js" и напишете в него следния код:

function eurConverter() {

let bgn = document.getElementById("bgn").value;

let eur = (bgn / 1.95583).toFixed(2);

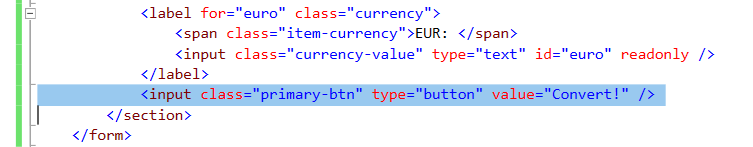
document.getElementById("euro").value = eur;

}

На първия ред функцията взима стойността на полето с **id="bgn"** и я записва в променливата "**bgn**". На втория ред превръща левовете в евро спрямо зададения курс и ограничава записа до два символа след десетичната запетая. На последния ред записва получената стойност за евро в полето с **id="euro"**.

За да заработи приложението, остава да закачим функцията за бутона "Convert!".

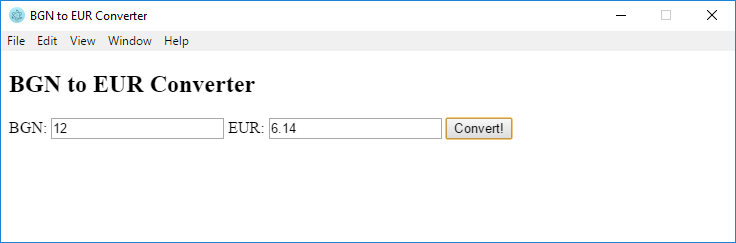
1. Отворете файла **index.html** и намерете input полето с тип "**button**":



Добавете атрибут за събитието "**onclick**", който да извиква функцията "**eurConverter()**" при натискане на бутона:

<input class="primary-btn" type="button" onclick="eurConverter()" value="Convert!" />

Отново стартирайте приложението и го тествайте:



1. Създайте нов файл с разширение **\*.css** и име "**index.css**". [CSS](https://www.w3schools.com/html/html_css.asp) служи за стилизиране на елементите в HTML. Отворете файла "index.html" и добавете следния ред в тага "<head>":

<link rel="stylesheet" href="index.css" type="text/css" />

Във файла "**index.css**" сложете следния код:

body {

font-family: 'Lato', sans-serif;

color: #FFFFFF;

}

.content-form {

width: 50%;

margin: 5% auto;

background: #234465;

padding: 5px 10px 10px;

border-radius: 15px;

box-shadow: 5px 5px 10px #808080, 5px 5px 10px #6793c1 inset;

}

.currency-value {

border: none;

padding: 5px;

border-radius: 5px;

}

.title {

text-align: center;

}

.item-currency {

font-weight: 700;

}

.currency {

margin: auto;

padding-bottom: 15px;

}

.items {

display: flex;

flex-direction: column;

justify-content: flex-start;

}

.primary-btn {

margin: auto;

border: none;

padding: 10px 30px;

border-radius: 10px;

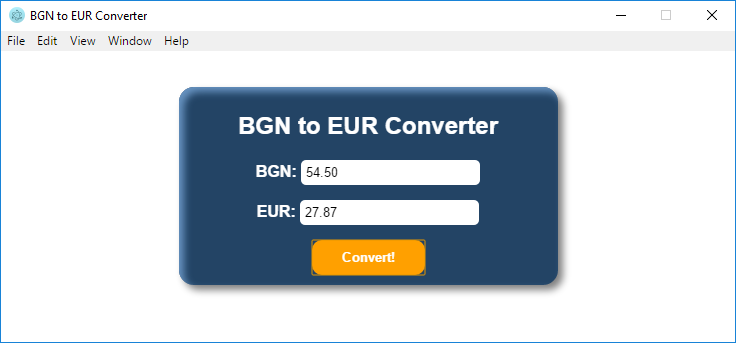
background-color: #ffa000;

color: #FFFFFF;

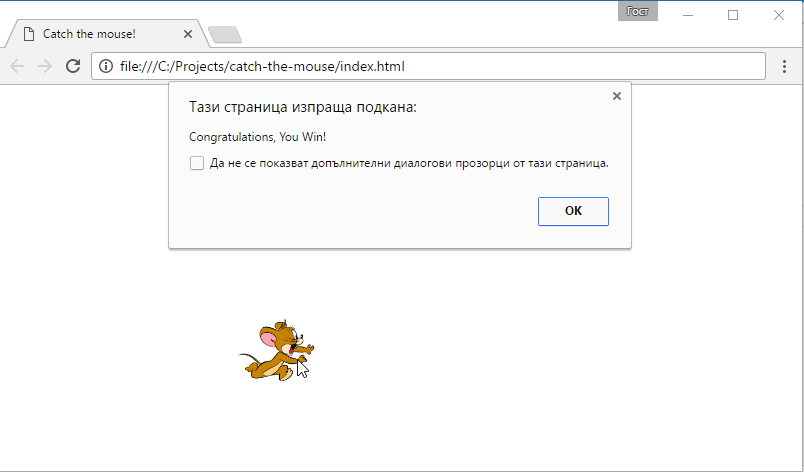
font-weight: 700;

}

Стартирайте приложението:



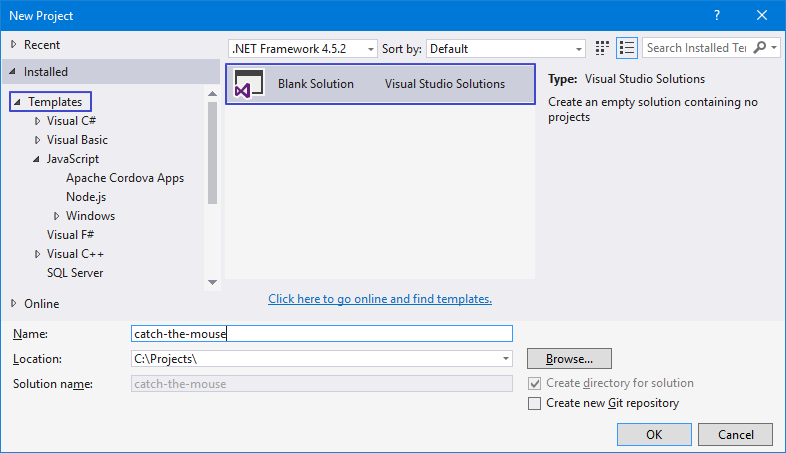
## \*\*\* Уеб приложение: хвани мишката!



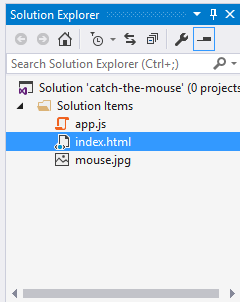
При преместване на курсора на мишката върху изображението, то се премества на случайна позиция. Така се създава усещане, че „**изображението бяга от курсора** и е трудно да се хване“. При „хващане“ на изображението, се извежда съобщение-поздрав.

**\* Подсказка**: напишете обработчик за събитието mouseover и премествайте изображението на случайна позиция. Използвайте генератор за случайни числа Math.random(). Позицията на изображението се задава от свойството style.position. За да „хванете мишката“, напишете функция за събитието onclick.

1. Създайте нов празен **Visual Studio Solution** с име „**catch-the-mouse**“:



1. В проекта създайте два файла: **index.html** и **app.js**.Добавете изображението “mouse.jpg”. Структурата на проекта трябва да изглежда по следния начин:



1. Можете да си помогнете с кода по-долу:

index.html:

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title>Catch the mouse!</title>

<script src="app.js" type="text/javascript"></script>

</head>

<body>

<img id="image" style="position:absolute;top:30%;left:30%; width:10%; height:auto; " src="images/mouse.jpg" />

</body>

</html>

app.js:

function chaseMouse() {

let img = document.getElementById("image");

img.style.position = "absolute";

img.style.left = (Math.random() \* 300) + "px";

img.style.top = (Math.random() \* 300) + "px";

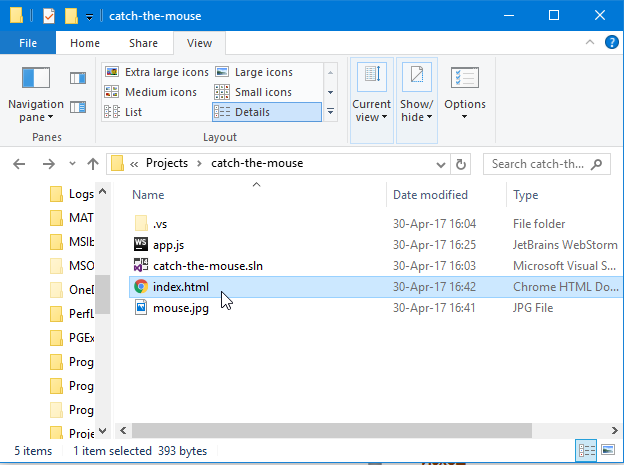
}

function catchMouse() {

alert("Congratulations, You Win!")

}

Тествайте приложението като отворите папката на проекта в **explorer** и стартирате файла **index.html:**



# Изпитни задачи от минали издания на курса

## \*Учебна зала

*Първа задача от междинния изпит на 6 март 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/169#0)*.*

**Учебна зала** има правоъгълен размер w на h метра, без колони във вътрешността си. Залата е разделена на две части – лява и дясна, с коридор приблизително по средата. В лявата и в дясната част има **редици с бюра**. В задната част на залата има голяма **входна врата**. В предната част на залата има **катедра** с подиум за преподавателя. Едно **работно място** заема **70 на 120 cm** (маса с размер 70 на 40 cm + място за стол и преминаване с размер 70 на 80 cm). **Коридорът** е широк поне **100 cm**. Изчислено е, че заради **входната врата** (която е с отвор 160 cm) се губи точно **1 работно място**, а заради **катедрата** (която е с размер 160 на 120 cm) се губят точно **2 работни места**. Напишете програма, която прочита размерите на учебната зала и изчислява **броя работни места в нея** при описаното разположение (вж. фигурата).

### Вход

Програмата чете 2 **числа**, по едно на ред: h (дължина в метри) и w (широчина в метри), въведени от потребителя.

Ограничения: **3** ≤ h ≤ w ≤ **100**.

### Изход

Да се отпечата на конзолата едно цяло число: **броят места** в учебната зала.

### Примерен вход и изход

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Чертеж** | **Обяснения** |
| 15  8.9 | 129 |  | Залата е дълга 1500 cm. В тях могат да бъдат разположени **12 реда** (12 \* 120 cm = 1440 + 60 cm остатък).  Залата е широка 890 cm. От тях 100 cm отиват за коридора в средата. В останалите 790 cm могат да се разположат по **11 бюра** **на ред** (11 \* 70 cm = 770 cm + 20 cm остатък).  **Брой места** = **12 \* 11 - 3** = 132 - 3 = **129** (имаме 12 реда по 11 места = 132 минус 3 места за катедра и входна врата). |
| 8.4  5.2 | 39 |  | Залата е дълга 840 cm. В тях могат да бъдат разположени **7 реда** (7 \* 120 cm = 840, без остатък).  Залата е широка 520 cm. От тях 100 cm отиват за коридора в средата. В останалите 420 cm могат да се разположат по **6 бюра** **на ред** (6 \* 70 cm = 420 cm, без остатък).  **Брой места** = **7 \* 6 - 3** = 42 - 3 = **39** (имаме 7 реда по 6 места = 42 минус 3 места за катедра и входна врата). |

## \*Зеленчукова борса

*Първа задача от изпита на 26 март 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/179#0)***.***

Градинар продавал реколтата от градината си на зеленчуковата борса. Продава **зеленчуци за** **N лева на килограм** и **плодове за M лева за килограм**. Напишете програма, която да **пресмята приходите от реколтата в евро** ( ако приемем, че **едно евро** е равно на **1.94лв**).

### Вход

Програмата чете **4 числа**, по едно на ред, въведени от потребителя:

* Първи ред (аргумент) – Цена за килограм зеленчуци – число с плаваща запетая
* Втори ред (аргумент) – Цена за килограм плодове – число с плаваща запетая
* Трети ред (аргумент) – Общо килограми на зеленчуците – цяло число
* Четвърти ред (аргумент) – Общо килограми на плодовете – цяло число

**Ограничения**: **Всички числа ще са в интервала от 0.00 до 1000.00**

### Изход

Да се отпечата на конзолата **едно число с плаваща запетая**: **приходите от всички плодове и зеленчуци в** **евро**. Резултатът **да се форматира до втория знак след запетаята.**

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 0.194  19.4  10  10 | 101.00 | Зеленчуците струват – 0.194лв. \* 10кг. = 1.94лв.  Плодовете струват – 19.4лв. \* 10кг. = 194лв.  Общо – 195.94лв. = 101евро |
| 1.5  2.5  10  10 | 20.62 |  |

## \*Ремонт на плочки

*Първа задача от изпита на 24 април 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/181#0)***.***

**На** **площадката** пред жилищен блок трябва да се **поставят плочки**. **Площадката** **е** с форма на **квадрат** **със страна N метра**. **Плочките** са **широки „W“ метра** и **дълги „L“ метра**. На площадката има **една** **пейка с ширина M метра** и **дължина O метра**. Под нея **не е нужно** да се слагат плочки. **Всяка плочка** се поставя за **0.2 минути**.

Напишете **програма**, която **чете размерите** на **площадката**, **плочките** и **пейката** и пресмята **колко плочки са необходими** да се покрие площадката и пресмята **времето за поставяне на всички плочки**.

**Пример:** **площадка** с **размер** **20м.** има **площ 400кв.м**. **Пейка** широка **1м.** и дълга **2м.**, заема **площ 2кв.м**. Една **плочка** е **широка** **5м.** и **дълга 4м.** има **площ = 20кв.м.** **Площта**, която трябва да се покрие е **400 – 2 = 398 кв.м.** Необходими са **398 / 20 = 19.90 плочки**. Необходимото **време** е **19.90 \* 0.2 = 3.98 минути**.

### Вход

Програмата чете **5 числа**, въведени от потребителя:

* N – **дължината** на **страна** от **площадката** в интервала **[1...100]**
* W – **широчината** на една **плочка** в интервала **[0.1...10.00]**
* L – **дължината** на една **плочка** в интервала **[0.1...10.00]**
* М – **широчината** на **пейката** в интервала **[0...10]**
* О – **дължината** на **пейката** в интервала **[0...10]**

### Изход

Да се отпечатат на конзолата две числа: **броят плочки**, необходим за ремонта и **времето за поставяне**, всяко на нов ред. Резултатът **да се форматира до втория знак след запетаята.**

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 20  5  4  1  2 | 19.90  3.98 | **Обща площ** = 20 \* 20 = **400**; площ на **пейката** = 1 \* 2 = **2**  **Площ** **за покриване** = 400 – 2 = **398**  **Площ на плочки** = 5 \* 4 = **20**  **Необходими плочки** = 398 / 20 = **19.9**  **Необходимо време** = 19.9 \* 0.2 = **3.98** |
| 40  0.8  0.6  3  5 | 3302.08  660.42 | |

## \*Парички

*Първа задача от изпита на 17 юли 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/233#0)***.***

Преди време **Пешо си е купил биткойни**. Сега ще ходи на екскурзия из Европа и **ще му трябват евра**. Освен биткойни **има и китайски юанa**. Пешо иска да **обмени парите** си **в евро** за екскурзията. Напишете програма, която да **пресмята колко евра може да купи спрямо следните валутни курсове:**

* **1 биткойн** = **1168 лева**.
* **1 китайски юан** = **0.15 долара**.
* **1 долар** = **1.76 лева**.
* **1 евро** = **1.95 лева**.

Обменното бюро има **комисионна от 0 до 5 процента от крайната сума в евро.**

### Вход

Програмата чете **3 числа**, въведени от потребителя:

* На **първия ред(аргумент)** – **броят биткойни**. **Цяло число в интервала** **[0…20]**
* На **втория ред(аргумент)** – **броят китайски юана**. **Реално число в интервала [0.00… 50 000.00]**
* На **третия ред(аргумент)** – **комисионната. Реално число в интервала [0.00 ... 5.00]**

### Изход

На конзолата **да се отпечата 1 число** - **резултатът от обмяната на валутите**. Резултатът **да се форматира до втория знак след запетаята.**

### Примерен вход и изход

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| 1  5  5 | 569.67 | 1 биткойн = **1168 лева**  5 юана = 0.75 долара  0.75 долара = **1.32 лева**  **1168 + 1.32 = 1169.32 лева** = **599.651282051282 евро**  **Комисионна:** 5% от 599.651282051282 = **29.9825641025641**  **Резултат:** 599.651282051282 - 29.9825641025641 = **569.668717948718 евро** | |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 20  5678  2.4 | 12442.24 | 7  50200.12  3 | 10659.47 |

## \*Дневна печалба

*Първа задача от изпита на 28 август 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/274#0)***.***

Иван е програмист в **американска компания** и **работи** от вкъщи **средно N дни** **в месеца** като изкарва **средно по M долара на ден**. В края на годината Иван **получава бонус**, който е **равен на 2.5 месечни заплати**. **От спечеленото през годината му се удържат 25% данъци.** Напишете програма, която да **пресмята, колко е чистата средна печалба на Иван на ден в лева**, тъй като той харчи изкараното в България. Приема се, че в **годината има точно 365 дни**. **Курсът на долара** спрямо лева ще **се чете от последния подаден аргумент**.

### Вход

Програмата чете 3 **числа**, въведени от потребителя:

* На **първия ред(аргумент)** – **работни дни в месеца**. **Цяло число в интервала** **[5…30]**
* На **втория ред(аргумент)** – **изкарани пари на ден**. **Реално число в интервала [10.00… 2000.00]**
* На **третия ред(аргумент) – курсът на долара спрямо лева /1 долар = X лева/. Реално число в интервала [0.99… 1.99]**

### Изход

На конзолата **да се отпечата 1 число** – **средната печалба на ден в лева**. Резултатът **да се форматира до втория знак след запетаята.**

### Примерен вход и изход

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| 21  75.00  1.59 | 74.61 | **1 месечна заплата** = 21 \* 75 = **1575 долара**  **Годишен доход** = 1575\*12 + 1575\*2.5 = **22837.5 долара**  **Данък** = 25% от 22837.5 = **5709.375**  **Чист годишен доход** = 17128.125 долара = **27233.71875 лв**  **Средна печалба на ден =** 27233.71875 / 365 = **74.61 лева** | |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 15  105  1.71 | 80.24 | 22  199.99  1.50 | 196.63 |