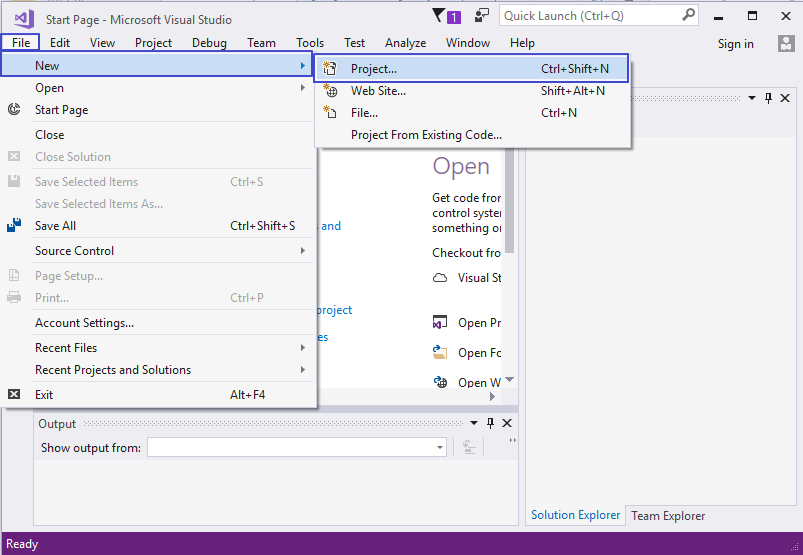
# Упражнения: Първи стъпки в коденето

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

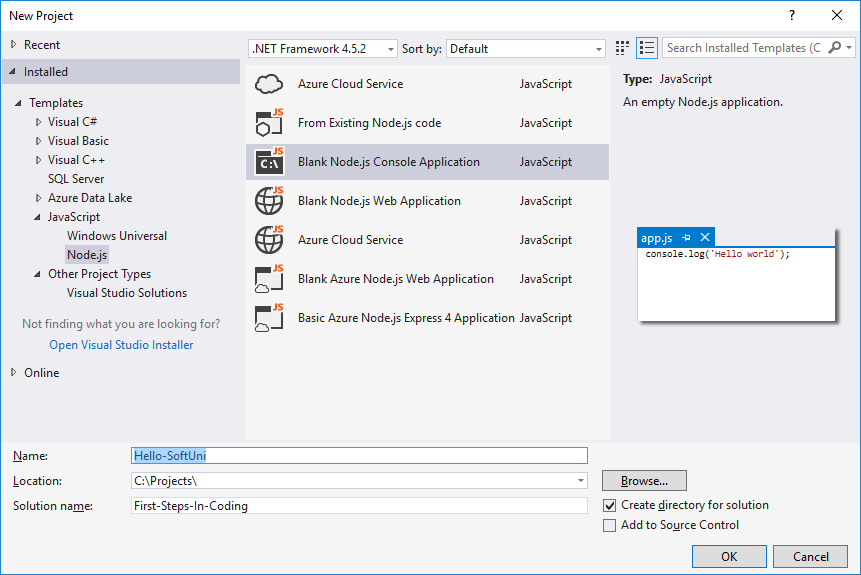
## Конзолна програмка “Hello SoftUni”

Напишете **конзолна Node.js програма**, която отпечатва текста “**Hello SoftUni**”.

1. Стартирайте Visual Studio.
2. Създайте нов конзолен проект: [File]🡪 [New] 🡪 [Project].



1. Изберете от диалоговия прозорец [JavaScript] 🡪 [Node.js] 🡪 [Blank Node.js Console Application] и дайте подходящо име на проекта, например “Hello-SoftUni”:



1. Изтрийте генерирания код във файла "**app.js**" и го заместете с функцията hello()**:**

function hello() {

// Your code goes here

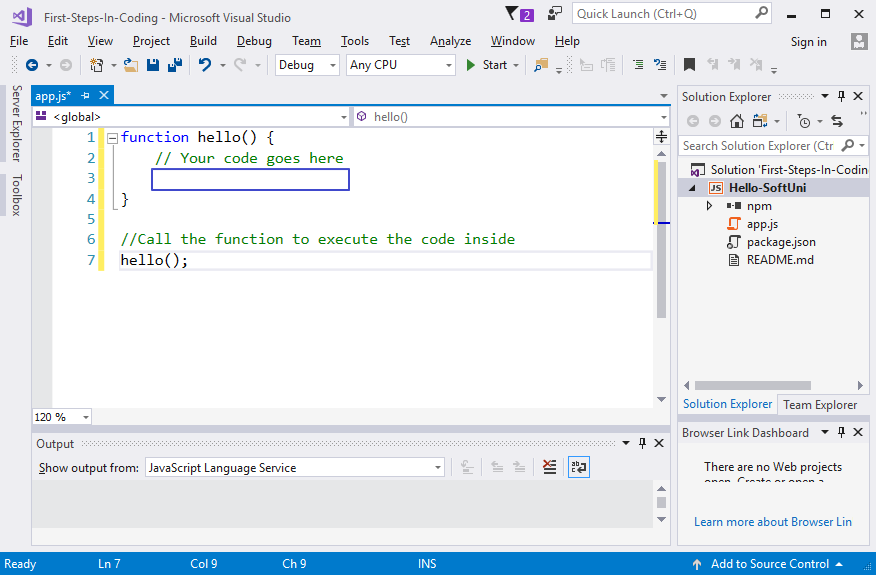
}

// Call the function to execute the code inside

hello();

В тялото на функцията се пише програмен код (команди) на езика JavaScript.

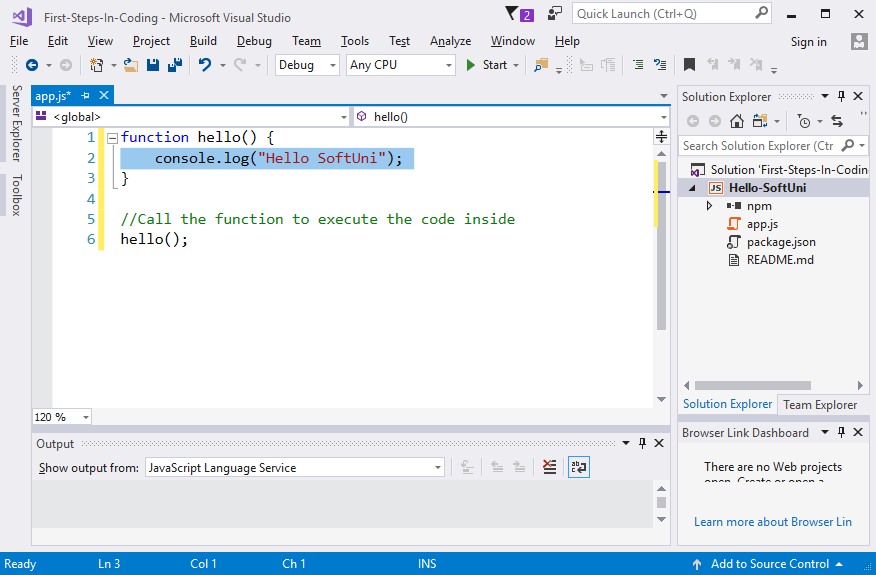
1. Придвижете курсора между отварящата и затварящата скоба { }.



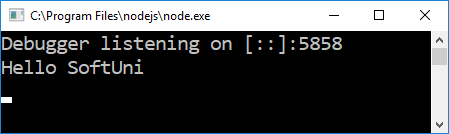
1. Напишете следния програмен код (команда за печатане на текста **"Hello SoftUni"**):

|  |
| --- |
| console.log("Hello SoftUni"); |

Кодът на програмата се пише отместен навътре с една табулация.



1. **Стартирайте** програмата с натискане на **[F5]**. Трябва да получите следния резултат:

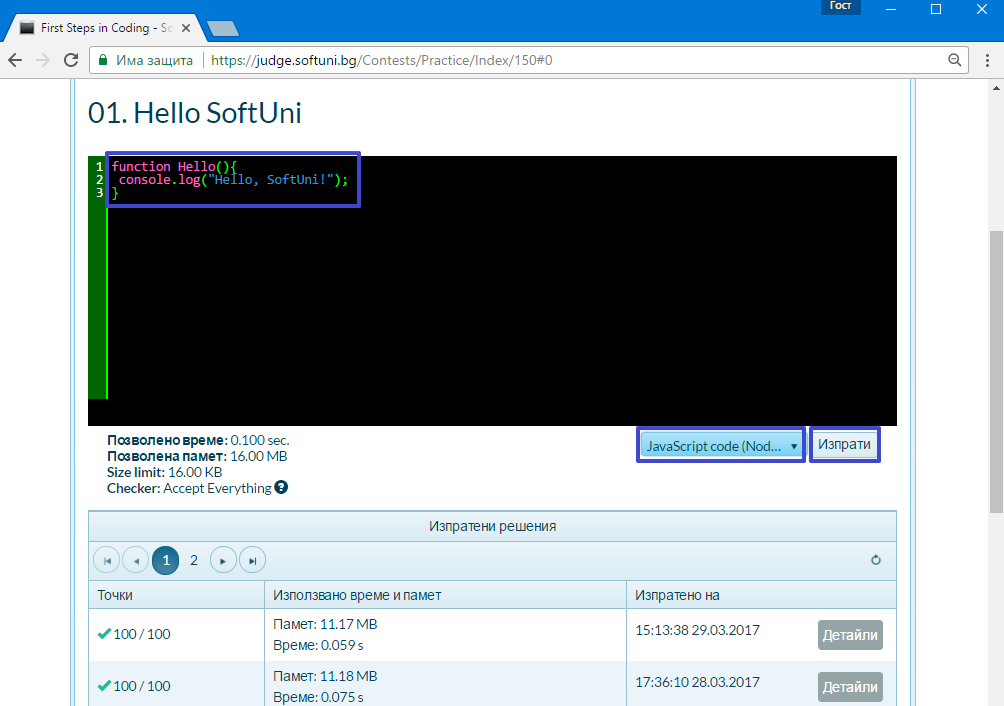


1. **Тествайте** решението на тази задача в онлайн judge системата на СофтУни. За целта първо отворете [https://judge.softuni.bg/Contests/Practice/Index/150#0](https://judge.softuni.bg/Contests/Practice/Index/150" \l "0). Влезте с вашето потребителско име в СофтУни. Ще се появи прозорец за изпращане на решения за задача “**Hello SoftUni**”. Копирайте сорс кода от Visual Studio, без извикването на функцията, и го поставете в полето за изпращане на решения:

function hello() {

console.log("Hello SoftUni");

}

1. От падащото менюдо[Submit]бутона **променете стратегията на "JavaScript code(NodeJS)" и изпратете решението** за оценяване с **бутона [Submit]**. Ще получите резултата след няколко секунди в таблицата с изпратени решения в judge системата: 

## Конзолна програма “Expression”

Напишете **конзолна Node.js програма**, която пресмята и отпечатва стойността на следния **числен израз**:

|  |
| --- |
| (3522 + 52353) \* 23 - (2336 \* 501 + 23432 - 6743) \* 3 |

Забележка: не е разрешено да се пресметне стойността предварително (например с Windows Calculator).

1. Направете нов Node.js конзолен проект с име “Expression”.
2. Копирайте функцията “expression()” и влезте в тялото, между { и }.

function expression() {

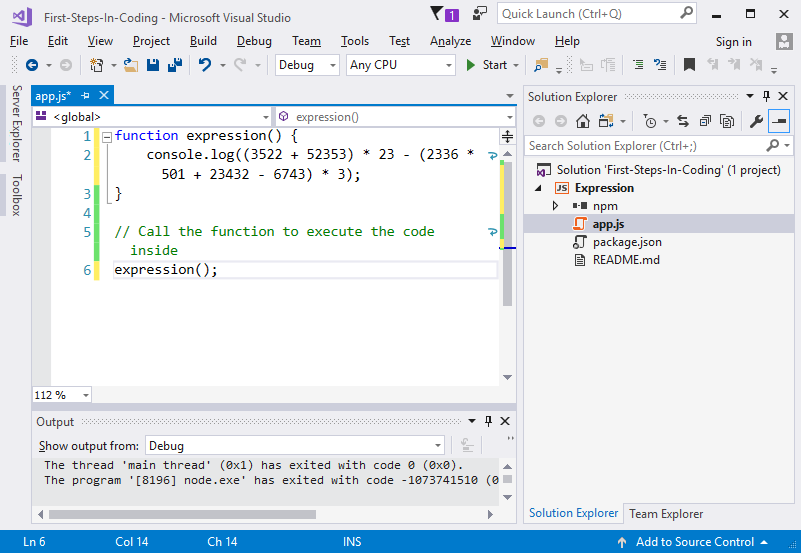
// Your code goes here

}

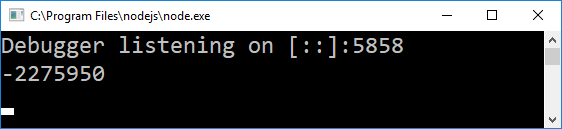
// Call the function to execute the code inside

expression();

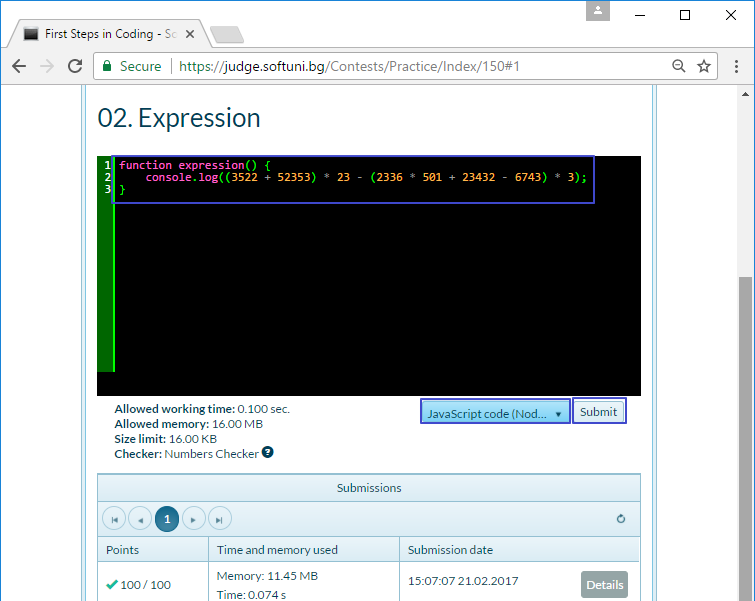
1. Сега трябва да напишете кода, който да изчисли горния числен израз и да отпечата на конзолата стойността му. Подайте горния числен израз в скобите на командата console.log():



1. Стартирайте програмата с [F5] и проверете дали вашият резултат прилича на нашия:



1. Тествайте вашата програма в judge системата: [https://judge.softuni.bg/Contests/Practice/Index/150#1](https://judge.softuni.bg/Contests/Practice/Index/150" \l "1).



## Числата от 1 до 20

Напишете Node.js конзолна програма, която отпечатва числата от 1 до 20 на отделни редове на конзолата.

1. Създайте конзолно Node.js приложение с име "Nums-1-To-20" и използвайте тялото на функцията "nums1To20()":

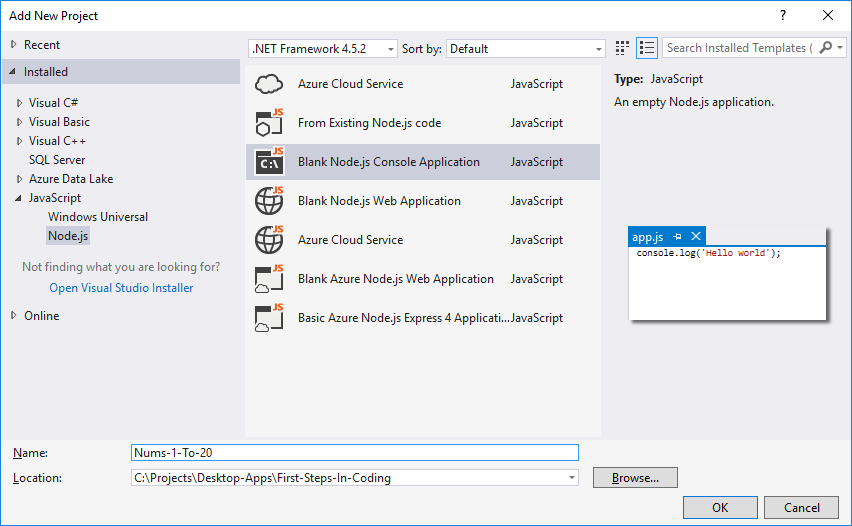
function nums1To20() {

// Your code goes here

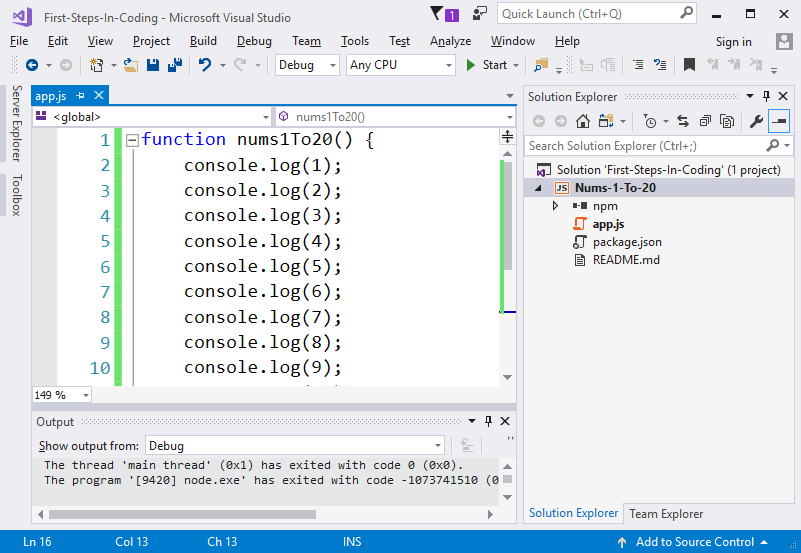
}

// Call the function to execute the code inside

nums1To20();



1. Напишете 20 команди console.log(), една след друга, за да отпечатате числата от 1 до 20.



1. **Тествайте** вашето решение на задачата в judge системата: [https://judge.softuni.bg/Contests/Practice/Index/150#2](https://judge.softuni.bg/Contests/Practice/Index/150" \l "2)
2. Можете ли да напишете програмата по **по-умен начин**, така че да не повтаряте 20 пъти една и съща команда? Потърсете в Интернет информация за „**[for loop JavaScript](https://www.google.bg/search?q=for+loop+javascript&oq=for+loop+jav&aqs=chrome.2.69i57j0l5.3355j0j7&sourceid=chrome&ie=UTF-8)**“.

## Триъгълник от 55 звездички

Напишете Node.js конзолна програма, която отпечатва **триъгълник от 55 звездички**, разположени на 10 реда:

|  |
| --- |
| \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\* |

1. Създайте ново конзолно Node.js приложение с име “Triangle-Of-55-Stars”.
2. Напишете код, който печата триъгълника от звездички, например чрез 10 команди, подобни на console.log("\*"). Използвайте тялото на функцията "triangleOfStars()**"**:

function triangleOfStars() {

// Your code goes here

}

// Call the function to execute the code inside

triangleOfStars();

1. **Тествайте** кода си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/150#3>.
2. Опитайте да подобрите решението си, така че да няма много повтарящи се команди. Може ли това да стане с **for цикъл**?

## Лице на правоъгълник

Напишете Node.js програма, която прочита от конзолата две числа, **a** и **b,** въведени от потребителя, пресмята и отпечатва **лицето на правоъгълник** със страни **a** и **b**. Примерен вход и изход:

|  |  |  |
| --- | --- | --- |
| **a** | **b** | **area** |
| 2 | 7 | 14 |
| 7 | 8 | 56 |
| 12 | 5 | 60 |

1. Направете конзолна Node.js програма. За да прочетете двете числа, използвайте следния код:

function rectangleArea([arg1, arg2]) {

let a = Number(arg1);

let b = Number(arg2);

// Print the area

}

// Call the function to execute the code inside

rectangleArea(["2", "7"]);

rectangleArea(["7", "8"]);

1. Допишете програмата по-горе, за да пресмята лицето на правоъгълника и да го проверява.
2. Тествайте решението си в judge системата: [https://judge.softuni.bg/Contests/Practice/Index/150#4](https://judge.softuni.bg/Contests/Practice/Index/150" \l "4).

## \* Квадрат от звездички

Напишете Node.js конзолна програма, която прочита от конзолата цяло положително число **N**, въведено от потребителя и отпечатва на конзолата **квадрат от N звездички**, като в примерите по-долу:

|  |  |
| --- | --- |
| **вход** | **изход** |
| 3 | \*\*\*  \* \*  \*\*\* |
| 4 | \*\*\*\*  \* \*  \* \*  \*\*\*\* |
| 5 | \*\*\*\*\*  \* \*  \* \*  \* \*  \*\*\*\*\* |

1. Направете конзолна Node.js програма. За да прочетете числото **N** (2 ≤ N ≤100), използвайте следния код:

function squareOfStars([arg1]) {

let n = Number(arg1);

// Print the square

}

// Call the function to execute the code inside

squareOfStars(["3"]);

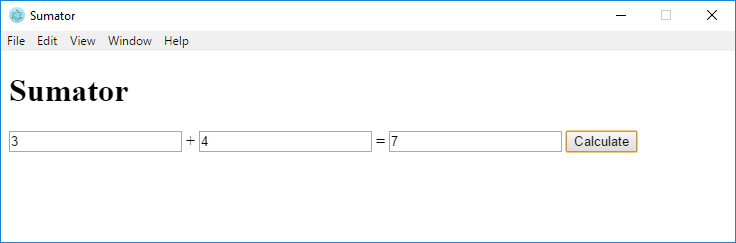
squareOfStars(["4"]);

1. Допишете програмата по-горе, за да отпечатва квадрат, съставен от звездички. Може да се наложи да използвате **for-цикли**. Потърсете информация в интернет.
2. Тествайте решението си в judge системата: [https://judge.softuni.bg/Contests/Practice/Index/150#5](https://judge.softuni.bg/Contests/Practice/Index/150" \l "5).

# Упражнения: Десктоп и Web приложения

## Десктоп приложение „Суматор за числа“

Напишете **графично (GUI) приложение**, което изчислява **сумата на две числа**:

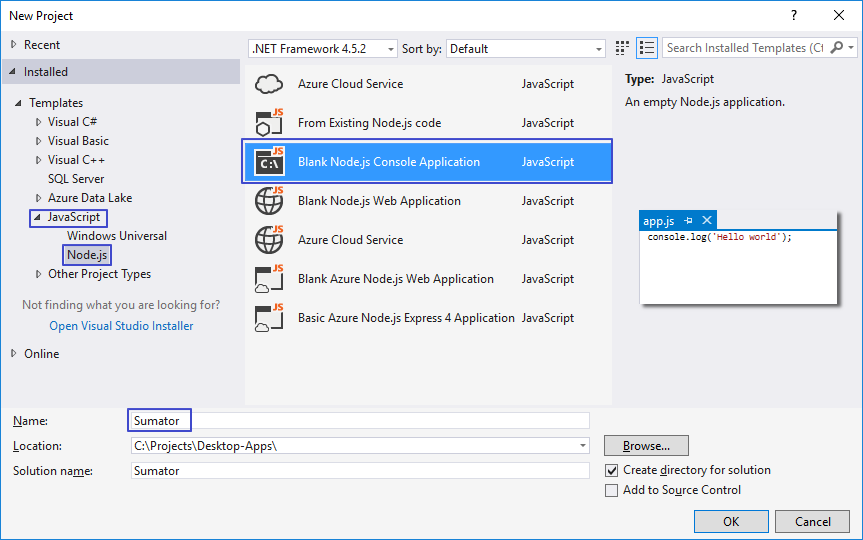


При въвеждане на две числа в първите две текстови полета и натискане на бутона [Calculate] се изчислява тяхната сума и резултатът се показва в третото текстово поле.

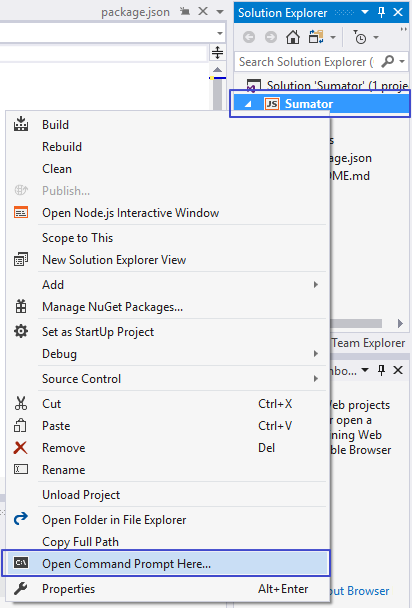
За разлика от конзолните приложения, които четат и пишат данните си във вид на текст на конзолата, **графичните (GUI) приложения** имат визуален потребителски интерфейс. Графичните приложения (настолни приложения, desktop apps) се състоят от един от няколко графични прозореца, в които има контроли: текстови полета, бутони, картинки, таблици и други.

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**.

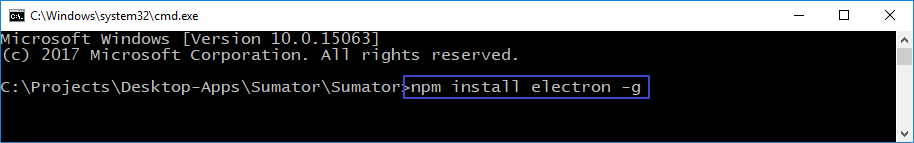
1. Във Visual Studio създайте нов Node.js проект от тип „**Blank Node.js Console Application**“:



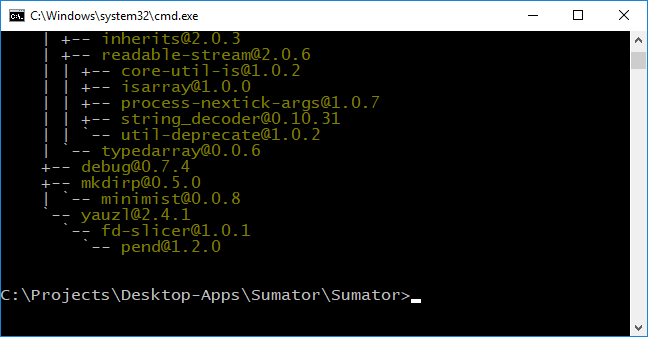
1. Следващата стъпка е инсталиране на платформата Electron.js. За целта стартирайте команден ред (**cmd**) в папката на проекта:



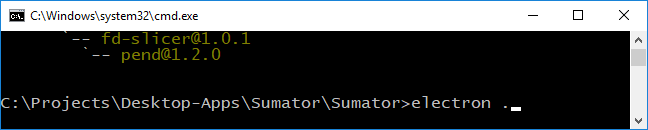
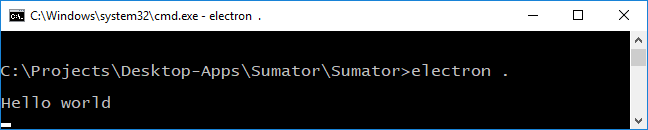
Изпълнете следната команда: "npm install electron -g**"**



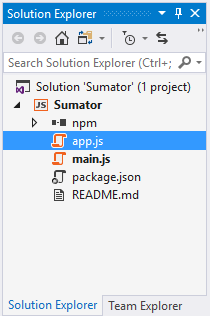
След приключване на инсталацията, прозорецът трябва да изглежда по подобен начин:



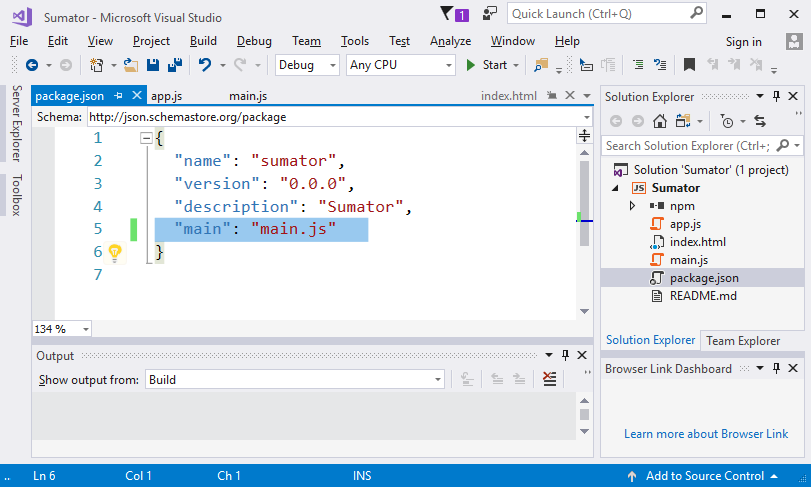
1. Платформата се стартира с командата "electron .**"**.В "**package.json**"за стартов файл на проекта е обозначен скриптът "app.js", който печата в конзолата ред с текст "Hello world". Това означава, че платформата е правилно инсталирана и следващата стъпка е създаването на файл "main.js", който ще бъде отправна точка за стартиране на десктоп приложението.

Структурата на проекта трябва да изглежда по следния начин:



1. В проекта намерете файла "package.json" и променете main скрипта с "main.js":



1. Electron.js стартира от файла "main.js", а в случай, че не е указан стартов файл, стартира от "index.js". Кодът, описан в този файл, 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: 250, 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()

})

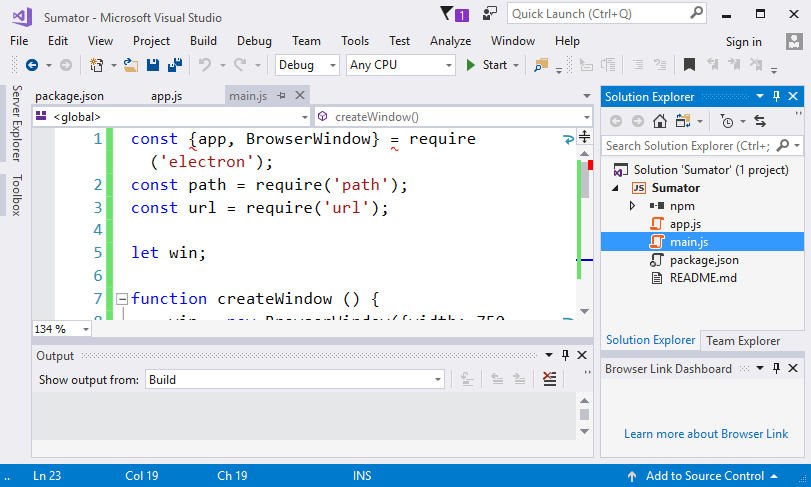
app.on('activate', () => {

if (win === null) {

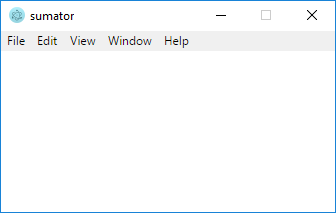
createWindow()

}

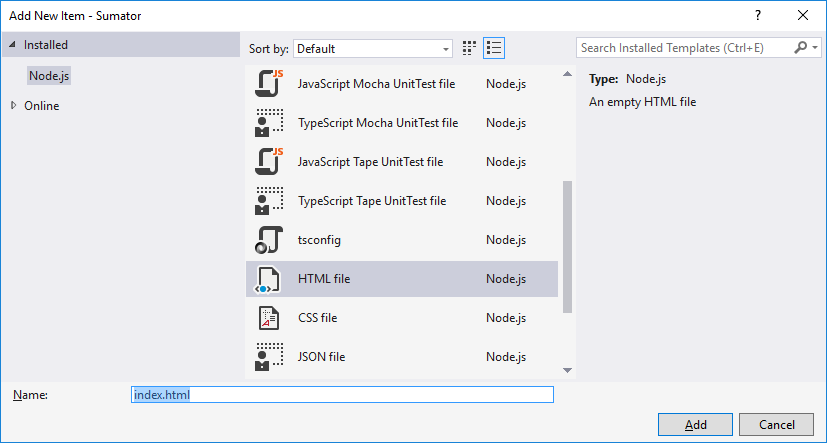
})



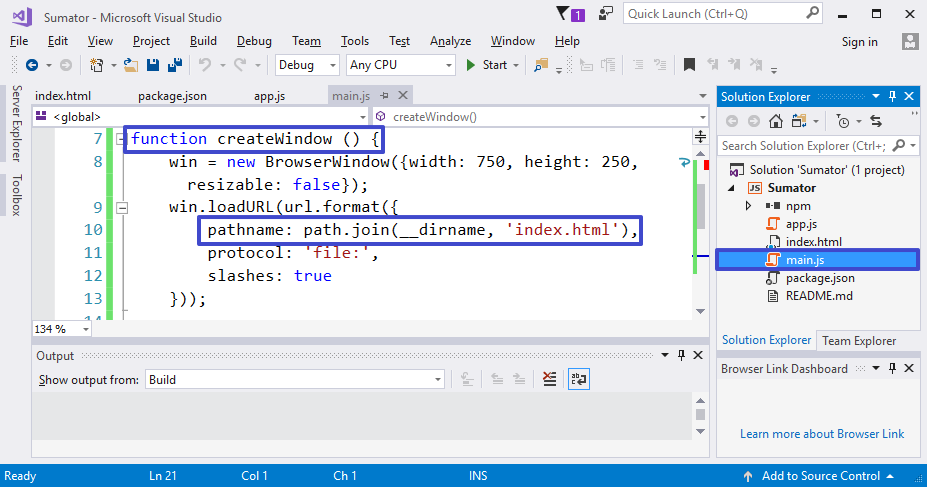
Стартирайте приложението с командата "**electron .**" в командния ред. Тъй като проектът все още не съдържа html файл с елементи за визуализиране, кодът генерира празен прозорец:



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



Това е файлът, който платформата визуализира в прозореца при стартиране на приложението:



Кодът в "index.html" трябва да изглежда по подобен начин:

<!DOCTYPE html>

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

<head>

<meta charset="utf-8" />

<title>Sumator</title>

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

</head>

<body>

<h1>Sumator</h1>

<form>

<input type="number" id="num1"/>

<span>+</span>

<input type="number" id="num2" />

<span>=</span>

<input type="number" readonly="readonly" id="sum"/>

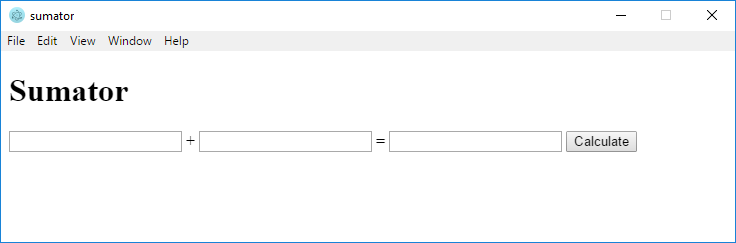
<input type="button" onclick="calc()" value="Calculate"/>

</form>

</body>

</html>

При повторно стартиране на приложението, трябва да се визуализират полетата за въвеждане на числата и бутона за изчисляване на сумата им:



1. Последната стъпка е създаването на функцията calc(), която извършва сумирането на числата и записва получения резултат в полето "sum". Във файла "app.js" напишете следния код:

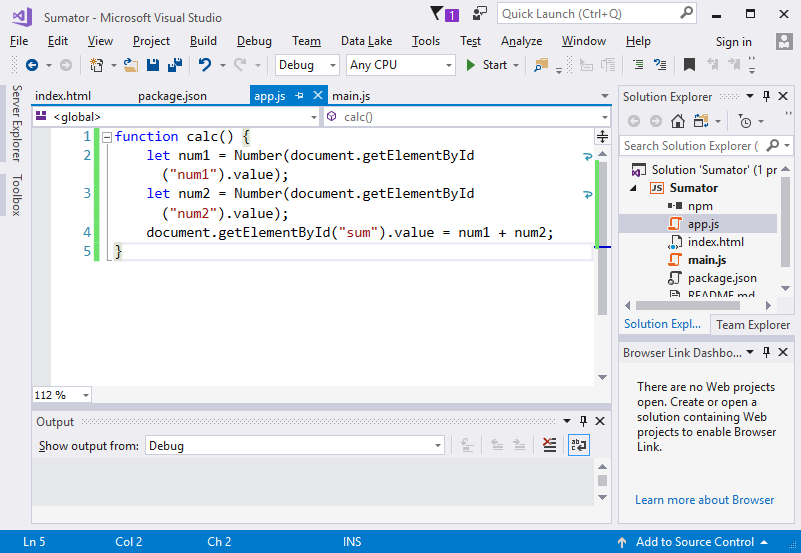
function calc() {

let num1 = Number(document.getElementById("num1").value);

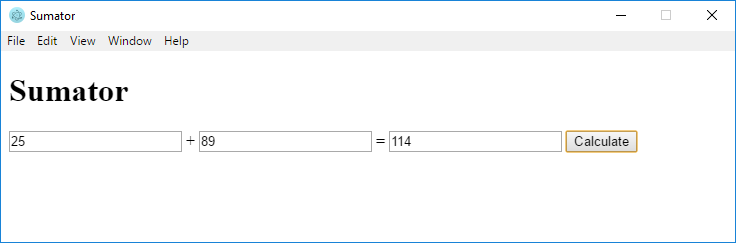
let num2 = Number(document.getElementById("num2").value);

document.getElementById("sum").value = num1 + num2;

}

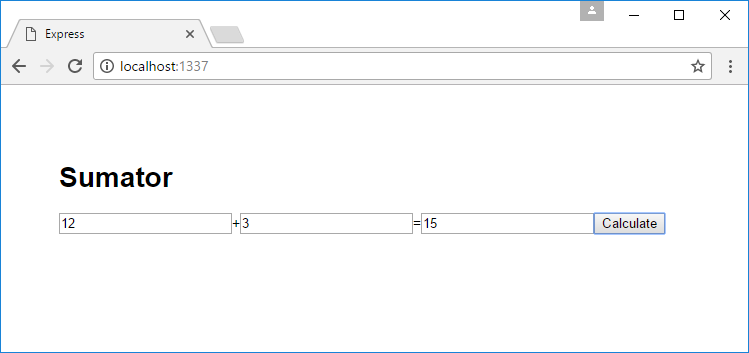


1. Стартирайте приложението и го тествайте :



## Уеб приложение „Суматор за числа“

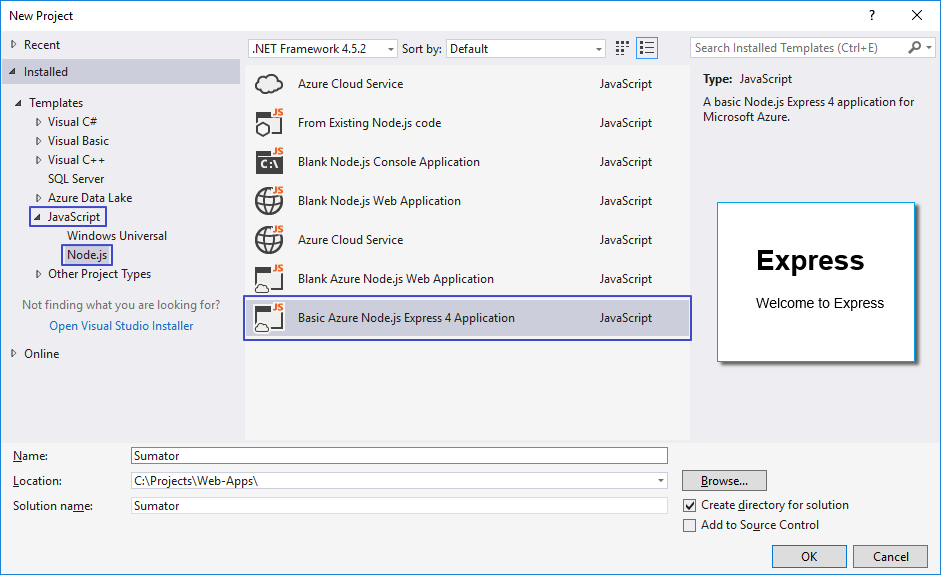
Напишете **уеб приложение**, което изчислява **сумата на две числа**. При въвеждане на две числа в първите две текстови полета и натискане на бутона [Calculate] се изчислява тяхната сума и резултатът се показва в третото текстово поле. Уеб приложението би могло да изглежда по следния начин:



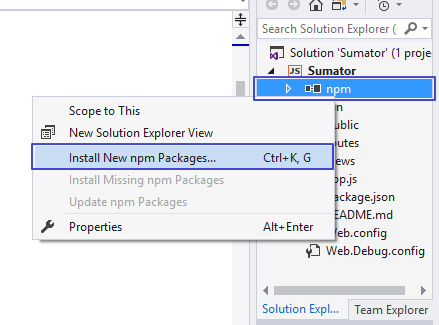
За разлика от конзолните приложения, които четат и пишат данните си във вид на текст на конзолата, **уеб приложения** имат **уеб базиран потребителски интерфейс**. Уеб приложенията се зареждат от някакъв Интернет адрес (URL) чрез стандартен **уеб браузър**. Потребителите пишат входните данни в страница, визуализирана от уеб приложението, данните се обработват на уеб сървъра и резултатите се показват отново в страницата в уеб браузъра.

За нашето уеб приложение ще използваме технологията **Express.js**, която позволява създаване на уеб приложения с езика за програмиране **JavaScript** в средата за разработка **Visual Studio**.

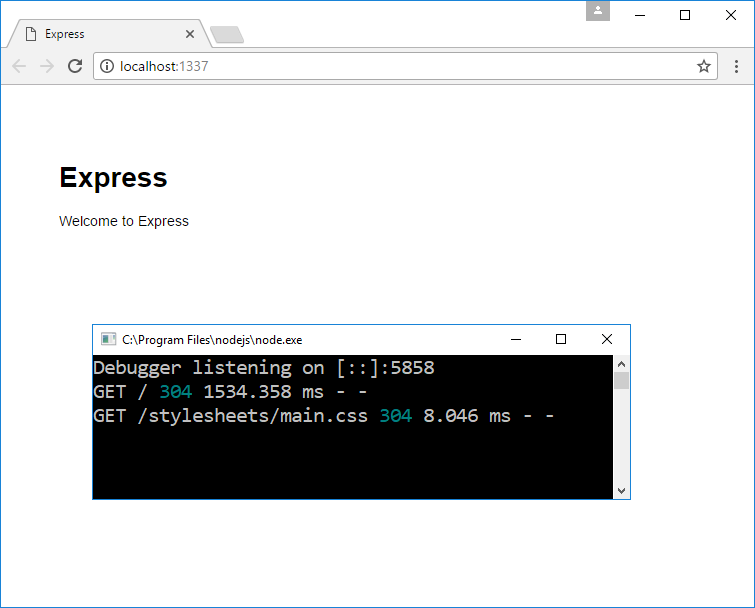
1. Във Visual Studio създайте нов Node.js проект от тип „**Basic Azure Node.js Express 4 Application**“:



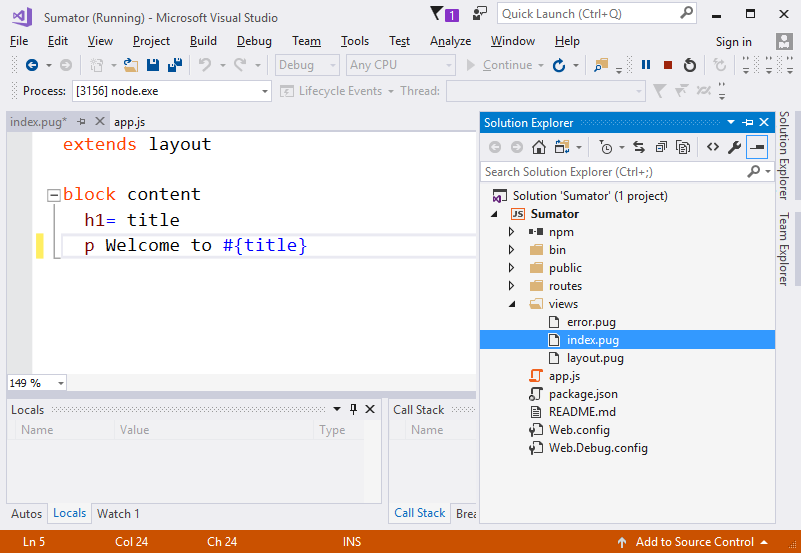
1. Инсталирайте липсващите npm пакети, нужни за стартиране на приложението:



След успешно инсталиране на пакетите, при стартиране на приложението с [F5], то трябва да изглежда по следния начин:



1. Намерете файла views/index.pug. В него стои изгледът (view) за главната страница на уеб приложението:



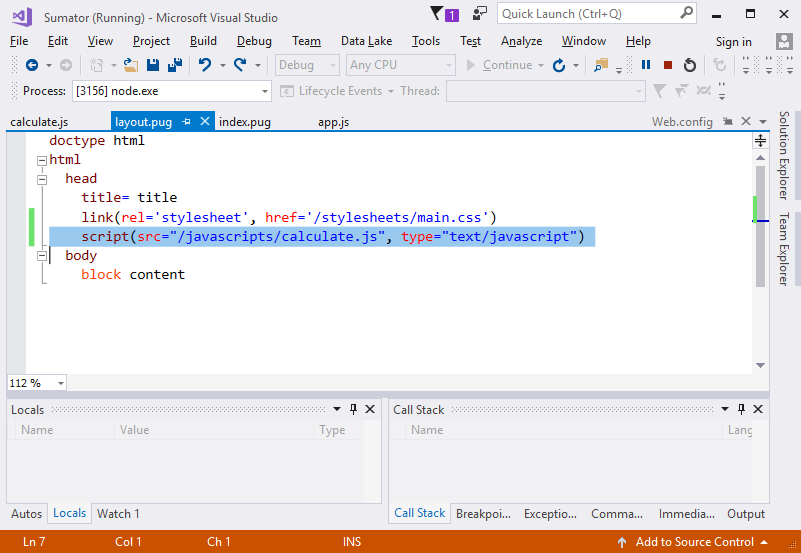
1. Изтрийте стария код от файла index.pug и напишете вместо него следния код:

|  |
| --- |
| extends layout  block content  h1= "Sumator"  form  input(type='number', id='num1')  span +  input(type='number', id='num2')  span =  input(type='number', readonly='readonly', id='sum')  input(type='button', onclick='calc()', value='Calculate') |

Този код създава една **уеб форма** с **три текстови полета** и един **бутон** в нея. Указано е, че при натискане на бутона [Calculate] ще се извика функцията calc().

1. Намерете файла views/layout.pug и добавете следния ред:

script(src="/javascripts/calculate.js", type="text/javascript")



1. Остава да се напише **функцията** (calc()), което сумира числата при натискане на бутона [Calculate]. Създайте файла public\javascripts\calculate.js и напишете следния код:

function calc() {

let num1 = Number(document.getElementById('num1').value);

let num2 = Number(document.getElementById('num2').value);

document.getElementById('sum').value = num1 + num2;

}

Този код осъществява действието “**calculate**”. То обработва два параметъра num1 и num2 и изчислява и записва тяхната **сума**.

1. Приложението е готово. Можете да го стартирате с **[F5]** и да го тествате дали работи:

