

Прости проверки

Логически изрази и проверки Условна конструкция if-else

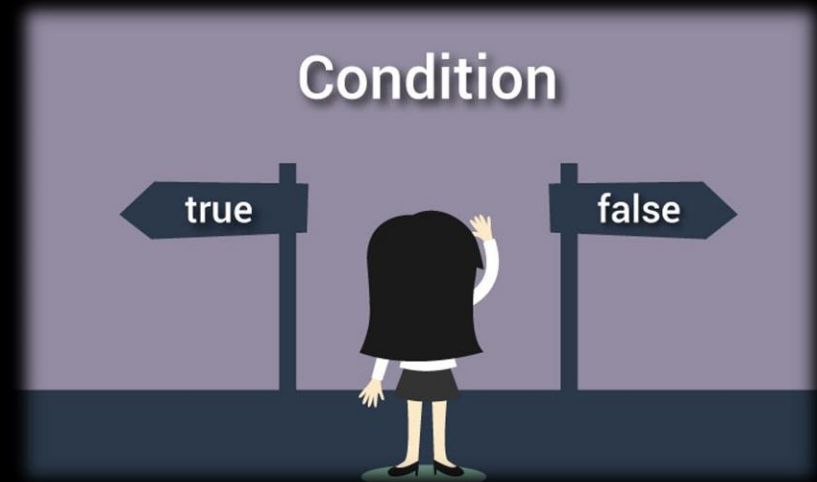
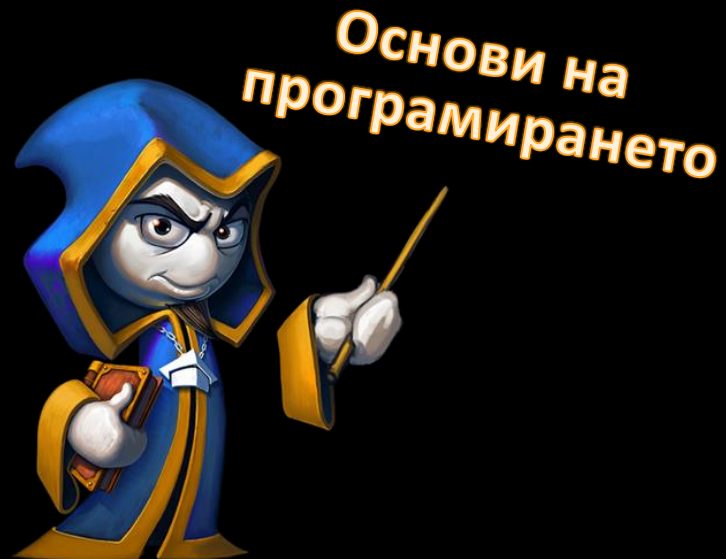


СофтУни

трейнърски екип

Софтуерен университет

<http://softuni.bg>



Have a Question?

sli.do

#TODO

Съдържание

1. Логически изрази и проверки
 - Оператори за сравнение: $<$, $>$, $==$, $!=$, ...
2. Конструкции **if** и **if-else**
3. Живот на променливата
4. Серия от проверки: **if-else-if-else...**
5. Дебъгване





Логически изрази и проверки

Оператори за сравнение

Сравняване на числа

- В програмирането можем да сравняваме стойности:

```
let a = 5;  
let b = 10;  
console.log(a < b);           // True  
console.log(a > 0);           // True  
console.log(a > 100);         // False  
console.log(a < a);           // False  
console.log(a <= 5);          // True  
console.log(b == 2 * a);      // True  
console.log("2" === 2);       // False
```

Оператор < (по-малко)

Оператор >
(по-голямо)

Оператор <=
(по-малко
или равно)

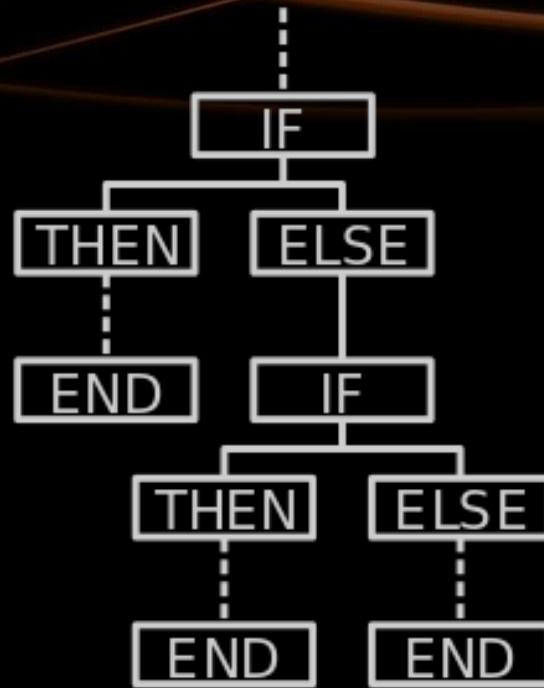
Оператор == (равно)

Оператори за сравнение

Оператор	Означение
Проверка за равенство по стойност (и тип данни)	==, ===
Проверка за различно по стойност (и тип данни)	!=, !==
По-голямо	>
По-голямо или равно	>=
По-малко	<
По-малко или равно	<=

- Пример:

```
let result = 5 != 6;  
console.log(result); // True
```



Прости проверки

Условни конструкции

Прости проверки

- В програмирането често **проверяваме условия** и за извършване на различни действия, според резултата от проверката
- Пример: въвеждаме оценка и проверяваме дали е отлична (≥ 5.50)

```
function isExcellent([arg1]) {  
  let grade = Number(arg1);  
  if (grade >= 5.50) {  
    console.log("Excellent!");  
  }  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#0>

Проверки с if-else конструкция

- Въвеждаме оценка, проверяваме дали е отлична или не е

```
function isExcellent([arg1]) {  
  let grade = Number(arg1);  
  if (grade >= 5.50) {  
    console.log("Excellent!");  
  } else {  
    console.log("Not excellent.");  
  }  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#1>

Блок от код

- Къдравите скоби `{ }` въвеждат **блок** (група команди)

- Без скобите след **if** се изпълнява само

Само този ред ще се отпечата

```
let color = "red";  
if (color == "red")  
    console.log("tomato");  
else if (color == "yellow")  
    console.log("banana");  
console.log("bye");
```

Изпълнява се винаги – не е част от **if/else** конструкцията

```
let color = "red";  
if (color == "red")  
{  
    console.log("tomato");  
}  
else if (color == "yellow")  
{  
    console.log("banana");  
    console.log("bye");  
}
```

Четно или нечетно – пример

- Напишете програма, която проверява дали едно число е **четно** или **нечетно**:
 - ако е четно принтира "**even**"
 - ако е нечетно принтира "**odd**"

Четно или нечетно – решение

```
function isEven([arg1]) {  
  let num = parseInt(arg1);  
  if (num % 2 == 0) {  
    console.log("even");  
  } else {  
    console.log("odd");  
  }  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#2>

По-голямото число – задача

- Напишете програма, която:
 - чете две цели числа
 - извежда по-голямото от тях

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#3>

По-голямото число – решение

```
function greaterNumber([arg1, arg2]) {  
    let num1 = Number(arg1);  
    let num2 = Number(arg2);  
    if (num1 > num2) {  
        console.log("Greater number: " + num1);  
    } else {  
        console.log("Greater number: " + num2);  
    }  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#3>



Живот на променлива

Диапазон на използване на променлива

Живот на променлива

- Обхват, в който дадена променлива може да бъде използвана

```
function printVariable() {  
  let car = "Volvo";  
  if (true) {  
    let secondCar = "Ferrari";  
    console.log(secondCar); // Prints "Ferrari"  
  }  
  console.log(car); // Prints "Volvo"  
  console.log(secondCar); // Error  
}
```



```
if (income < 0) rate = 0.00;  
else if (income < 8925) rate = 0.10;  
else if (income < 36250) rate = 0.15;  
else if (income < 87850) rate = 0.23;  
else if (income < 183250) rate = 0.28;  
else if (income < 398350) rate = 0.33;  
else if (income < 400000) rate = 0.35;  
else rate = 0.396;
```

Серии от проверки

Серии от проверки

- Конструкцията **if/else-if/else...** може да е в серия
 - Пример: Да се провери дали въведеното число е по – голямо от 4 или от 6.

Въведете числото **7** като вход

```
function isBigger([arg1]) {  
  let num = Number(arg1);  
  if(num > 4){ console.log(num + " is bigger than 4"); }  
  if(num > 6){ console.log(num + " is bigger than 6"); }  
}
```

Изход: **7 is bigger than 4**
7 is bigger than 6

Серии от проверки (2)

- Да се провери дали въведеното число е по – голямо от 4 или от 6?

```
function isBigger([arg1]) {  
  let num = Number(arg1);  
  if(num > 4){ console.log(num + " is bigger than 4"); }  
  else if(num > 6){ console.log(num + " is bigger than 6"); }  
}
```

Изход: 7 is bigger than 4

Изписване на число до 10 с думи - задача

- Да се изпише с английски текст дадено число (от 0 до 10)

```
function number0to9([arg1]) {  
  let num = parseInt(arg1);  
  if (num == 1)  
    console.log("one");  
  else if (num == 2)  
    console.log("two");  
  else if (num == 3)  
    console.log("three");  
  // TODO: add more checks  
  else  
    console.log("number too big");  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#4>

Бонус точки – задача

- Дадено е цяло число – брой точки
 - Ако числото е до 100 включително, бонус точките са 5
 - Ако числото е по-голямо от 100, бонус точките са 20%
 - Ако числото е по-голямо от 1000, бонус точките са 10%
 - Допълнителни бонус точки:
 - За четно число \rightarrow 1 т.
 - За число, което завършва на 5 \rightarrow 2 т.
- Да се напише програма, която пресмята бонус точките и общия брой точки след прилагане на бонусите

Бонус точки – решение

```
function scoreCalculator([arg1]) {  
  let num = Number(arg1);  
  let bonusScore = 0;  
  
  if (num > 1000)  
    { bonusScore = num * 0.10; }  
  else // TODO: write more logic here ...  
  
  if (num % 10 == 5)  
    { bonusScore += 2; }  
  else // TODO: write more logic here ...  
  
  console.log("Bonus score: " + bonusScore);  
  console.log("Total score: " + (num + bonusScore));  
}
```

20	→	6 26
----	---	---------

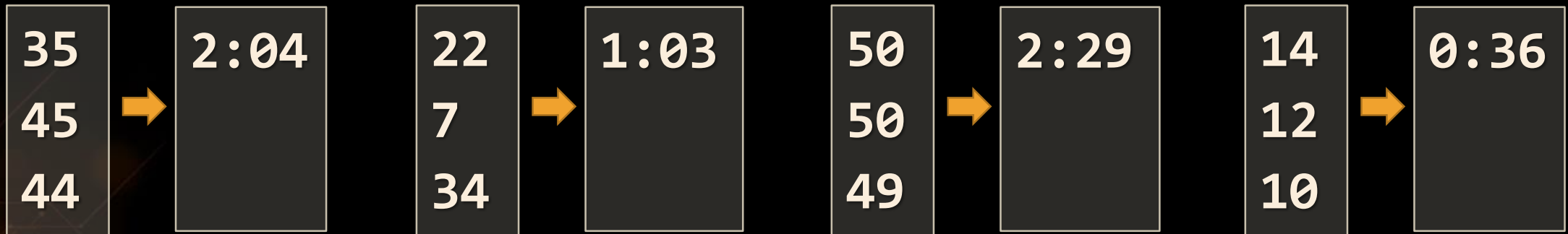
175	→	37 212
-----	---	-----------

2703	→	270.3 2973.3
------	---	-----------------

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#5>

Сумиране на секунди – задача

- Трима спортни състезатели финишират за някакъв брой **секунди** (между 1 и 50). Да се пресметне сумарното им време във формат "**минути:секунди**". Секундите да се изведат с **водеща нула** ($2 \rightarrow "02"$, $7 \rightarrow "07"$, $35 \rightarrow "35"$).
- Примери:



Сумиране на секунди – решение

```
function sumSeconds([arg1, arg2, arg3]) {  
  let sec1 = Number(arg1);  
  // TODO: Read also sec2 and sec3 ...  
  let secs = sec1 + sec2 + sec3;  
  let mins = 0;  
  if (secs > 59)    // TODO: Repeat this 2 times ...  
  { mins++; secs = secs - 60; }  
  if (secs < 10)  
  { console.log(mins + ":" + "0" + secs); }  
  else  
  { console.log(mins + ":" + secs); }  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#6>

Конвертор за мерни единици

- Да се напише програма, която преобразува разстояние между посочените в таблицата **мерни единици**:

- Вход: число +
входна мерна единица +
изходна мерна единица

- Примерен вход и изход:

12

km

ft

39370.0788 ft

входна единица	изходна единица
1 meter (m)	1000 millimeters (mm)
1 meter (m)	100 centimeters (cm)
1 meter (m)	0.000621371192 miles (mi)
1 meter (m)	39.3700787 inches (in)
1 meter (m)	0.001 kilometers (km)
1 meter (m)	3.2808399 feet (ft)
1 meter (m)	1.0936133 yards (yd)

Конвертор за мерни единици – решение

```
function metricConverter([arg1, arg2, arg3]) {  
  let size = Number(arg1);  
  let sourceMetric = arg2.toLowerCase();  
  let destMetric = arg3.toLowerCase();  
  if (sourceMetric == "km")  
    { size = size / 0.001; }  
  // Check the other metrics: mm, cm, ft, yd, ...  
  if (destMetric == "ft")  
    { size = size * 3.2808399; }  
  // Check the other metrics: mm, cm, ft, yd, ...  
  console.log(size + " " + destMetric);  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/152#7>

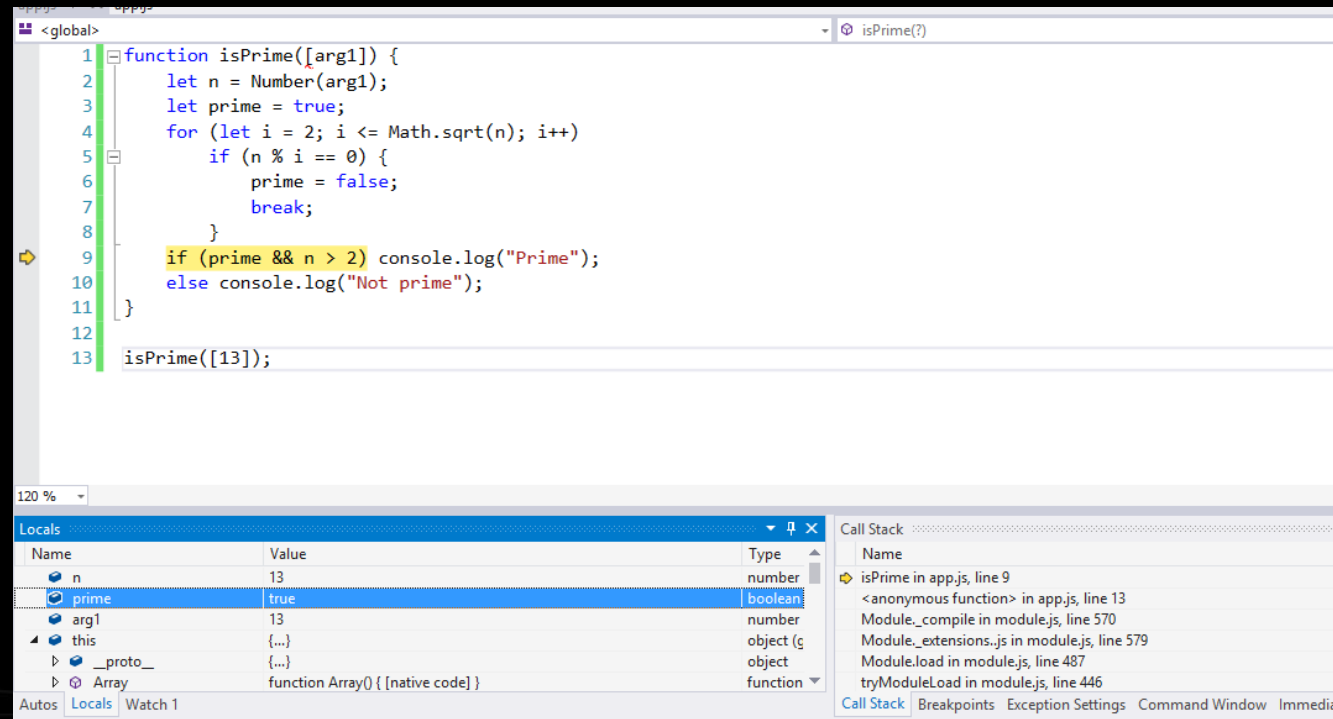


Дебъгване

Прости операции с дебъгер

Дебъгване

- Процес на „закачане“ към изпълнението на програмата, което ни позволява да проследи процеса на изпълнение
 - Това ни позволява да откриваме грешки (бъгове)



The screenshot shows a web browser's developer console with a JavaScript function `isPrime` defined. The function takes an argument `arg1`, converts it to a number, and checks if it is a prime number by testing divisibility from 2 up to the square root of the number. The function logs "Prime" if the number is prime and "Not prime" otherwise. The function is called with `isPrime([13]);`. The console shows the execution state with the `prime` variable set to `true` and the `arg1` variable set to `13`. The call stack shows the function `isPrime` in `app.js` at line 9, called by an anonymous function in `app.js` at line 13.

```
1 function isPrime([arg1]) {
2   let n = Number(arg1);
3   let prime = true;
4   for (let i = 2; i <= Math.sqrt(n); i++)
5     if (n % i == 0) {
6       prime = false;
7       break;
8     }
9   if (prime && n > 2) console.log("Prime");
10  else console.log("Not prime");
11 }
12
13 isPrime([13]);
```

Name	Value	Type
n	13	number
prime	true	boolean
arg1	13	number
this	{...}	object (g
__proto__	{...}	object
Array	function Array() { [native code] }	function

Call Stack

- isPrime in app.js, line 9
- <anonymous function> in app.js, line 13
- Module_compile in module.js, line 570
- Module_extensions.js in module.js, line 579
- Module.load in module.js, line 487
- tryModuleLoad in module.js, line 446

Дебъгване във Visual Studio

- Натискане на [F5] ще стартира програмата в **debug** режим.
- Можем да преминем към следващата **стъпка** отново [F10]
- Можем да създаваме [F9] стопери – **breakpoints**
 - До тях можем директно да стигнем изпозлвайки [F5]


```
C:\Program Files\nodejs\node.exe
Debugger listening on [::]:5858
Not excellent.
```

```
C:\Program Files\nodejs\node.exe
Debugger lis
Excellent!
```

```
C:\Program Files\nodejs\node.exe
Debugger listening on [::]:5858
Greater number: 5
```

Задачи с прости проверки

Работа на живо в клас (лаб)

Какво научихме днес?

- Конструкции за проверка на условие **if** и **if-else**:

```
if (условие) {  
    група команди;  
}  
else if (условие2) {  
    група команди;  
}  
else {  
    група команди;  
}
```

```
if (условие)  
    единична_команда;  
else if (условие2)  
    единична_команда;  
else if (условие3)  
    единична_команда;  
else if (условие4)  
    единична_команда;  
else if (условие5)  
    единична_команда;  
else  
    единична_команда;
```



Прости проверки



Въпроси?



- Настоящият курс (слайдове, примери, видео, задачи и др.) се разпространяват под свободен лиценз "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International"



Безплатни обучения в СофтУни

- Фондация "Софтуерен университет" – softuni.org
- Софтуерен университет – качествено образование, професия и работа за софтуерни инженери
 - softuni.bg
- СофтУни @ Facebook
 - facebook.com/SoftwareUniversity
- СофтУни форуми – forum.softuni.bg

