JAVASCRIPT - ЧАСТ 3

ARROW FUNCTIONS, TEMPLATE LITERALS, SPREAD OPERATOR, DESTRUCTURING ASSIGNMENT, OBJECT LITERAL, STRINGS AND UNICODE, MATH

СЪДЪРЖАНИЕ

- arrow functions
- template literals
- spread operator
- destructuring assignment
- object literal
- strings and unicode
- Math

ДА ПРИПОМНИМ

КАКВО СА ФУНКЦИИТЕ?

- Парчета от код
- Набор от действия, които решават даден проблем
- Могат да бъдат извиквани от различни места
- Могат да приемат параметри и да връщат стойност

QUIZ

```
function logFirstName(name) {
    console.log(name);
}

const logLastName = function (name) {
    console.log(name);
}
```

КАКВО Е ОБХВАТ НА ПРОМЕНЛИВА?

- Обхвата на една променлива показва къде може да се достъпи тя
- Глобални и локални променливи
- Променлива декларирана без ключовата дума var става глобална

QUIZ

```
function test() {
    console.log(a);
    console.log(b);
    console.log(c);

    var a = 5;
    const b = 4;

    if (true) {
        const c = 3;
    }

    console.log(a);
    console.log(b);
    console.log(c);
```

ARROW FUNCTIONS

() => {}

КАКВО СА ARROW FUNCTIONS?

- Lambda expressions в повечето езици за програмиране
- Олекотен вариант на нормалните функции
- Свързани са към техния lexical scope
- Нямат arguments обект

ПРИМЕРИ

```
const logName = function (name) {
    console.log(name);
}

const logName = (name) => console.log(name);

const logName = name => console.log(name);

logName("Boris");
```

ПРИМЕРИ

```
const getFullName = function (fristName, lastName) {
    return fristName + " " + lastName;
}

const getFullName = (fristName, lastName) => fristName + " " +

getFullName("Boris", "Georgiev");
```

ПРИМЕРИ

```
const numbers = [1, 2, 3];

console.log(numbers.map(number => number * 2));
console.log(numbers.map((number, index) => number * index));
```

TEMPLATE LITERALS

ES6

`Tagged ${Template} Literals`

КАКВО СА TEMPLATE LITERALS?

- Нова функционалност в ES6
- Улесняват работата със стрингове и темплейти

КАКВО МОГАТ TEMPLATE LITERALS?

- Могат да се интерполират (interpolate) променливи в тях
- Могат да се интерполират (interpolate) изрази (expressions) в тях
- Могат да се дефинират многоредови стрингове
- Могат да се ползват кавички и апострофи без да се escape-ват

ПРИМЕРИ

```
const name = "First Name";
console.log(`Hello my name is ${name}!`);
```

```
console.log(`Current date and time is: ${new Date()}`);
```

ОЩЕ ПРИМЕРИ

```
const articleTitle = "Article title";
const articleText = "Article text";
const htmlTemplate = `
    <article>
        <h1>${articleTitle}</h1>
        <p>${articleText}</p>
    </article>
`;
```

TAGGED TEMPLATES

```
function normal(template, ...expressions) {
  return template.reduce((accumulator, part, i) => {
    return accumulator + expressions[i - 1] + part
  })
}

const name = 'nico';
const outfit = 'leather jacket';
const text = normal`
    hello ${name},
    you look lovely today in that ${outfit}
`;
console.log(text);
```

ПРИМЕРИ

```
function upperExpr (template, ...expressions) {
  return template.reduce((accumulator, part, i) => {
    return accumulator + expressions[i - 1].toUpperCase() + pa
  })
}
const name = 'nico';
const outfit = 'leather jacket';
const text = upperExpr`
    hello ${name},
    you look lovely today in that ${outfit}
`;
console.log(text);
```

SPREAD OPERATOR

КАКВО Е SPREAD OPERATOR?

- Spread operator позволява iterable (като например масив) да бъде разделен на части
- Spread operator позволява обект да бъде разделен на части

ИЗПОЛЗВАНЕ НА SPREAD OPERATOR

Извикване на функции, които очакват повече от 1 параметър

```
const names = ["Izabela", "Hristova"];
console.log(getFullName(names[0], names[1]));
console.log(getFullName(...names));

const numbers = [1, 2, 3, 4];
console.log(Math.min(...numbers)); //Math.min(1, 2, 3, 4);
console.log(Math.max(...numbers)); //Math.max(1, 2, 3, 4);
```

ИЗПОЛЗВАНЕ НА SPREAD OPERATOR

Комбиниране на масиви

```
const arr1 = [1, 2, 3];
const arr2 = [5, 6, 7];

console.log(...[...arr1, 4, ...arr2]);
```

ИЗПОЛЗВАНЕ НА SPREAD OPERATOR

Клониране на масиви (shallow)

```
const arr3 = [1, 2, 3];
const arr4 = [...arr3];

arr3.push(4);
console.log(...arr3);
console.log(...arr4);
```

ИЗПОЛЗВАНЕ НА SPREAD OPERATOR

Комбиниране на обекти

```
const defaults = {
    isRtl: false,
    lang: "bg"
}
const options = {
    currency: "bgn"
}
console.log({ ...defaults, ...options });
```

ИЗПОЛЗВАНЕ НА SPREAD OPERATOR

Копиране на обекти (shallow)

```
const options = {
    isRtl: false,
    lang: "bg"
}
const copyOptions = { ...options };

options.isRtl = true;
```

ИЗПОЛЗВАНЕ НА SPREAD OPERATOR

Конвертиране на array like към масив

```
function sum () {
    const numbers = [...arguments];
    console.log(numbers.reduce(
        (acc, number) => acc + number,
        0
    ));
}
```

ИЗПОЛЗВАНЕ НА SPREAD OPERATOR

Rest params

```
function multiply(multiplier, ...numbers) {
    const result = numbers.map(number => multiplier * number);
    console.log(...result);
}
```

DESTRUCTURING ASSIGNMENT

КАКВО Е DESTRUCTURING ASSIGNMENT?

- Destructuring assignment е израз на JavaScript, който позволява присвояването на стойности от масиви или свойства от обекти в отделни променливи.

ПРИМЕРИ

```
const [a, b] = [1, 2];
console.log(a, b);
```

ПРИМЕРИ

```
const { age, name } = { age: 27, name: "Name", gender: "male"
console.log(age, name);
```

ПРИМЕРИ

Разменяне на стойности

```
function getMin(min, max) {
    if (min > max) {
        [min, max] = [max, min];
    }

    return min;
}

console.log(getMin(2, 1));
```

ПРИМЕРИ

Улеснява работата с функции, които връщат обекти

```
function getCoords() {
    return {
        x: 1,
        y: 2
    };
}

const { x, y } = getCoords();

console.log(x, y);
```

Стойности по подразбиране

REST & DESTRUCTURING

```
const [a, b, ...c] = [1, 2, 3, 4, 5, 6];
const { age, name, ...rest } = { age: 27, name: "Name", gender

console.log(a, b, c);
console.log(age, name, rest);
```

НОВОСТИ ПРИ ДЕФИНИРАНЕТО НА OBJECT LITERALS

PROPERTY VALUE - КРАТЪК СИНТАКСИС

Позволява дефинирането на ключ, който автоматично взима стойността от променлива със същото име

ПРИМЕРИ

```
const age = 27;
const name = "Name";
const obj = { age, name }; // { age: age, name: name };

console.log(obj);
```

COMPUTED PROPERTY NAMES

Позволява задаването на ключове на обекти, които не са известни преди изпълнение на кода

ПРИМЕРИ

```
function getObject(type) {
    return {
        [type]: {
            age: 27,
            name: "Name"
        }
    };
}

console.log(getObject("person"));
```

НОВОСТИ ПРИ СТРИНГОВЕТЕ И UNICODE

- String.prototype.startsWith
- String.prototype.endsWith
- String.prototype.includes
- String.prototype.repeat

STRING.PROTOTYPE.STARTSWITH

Проверява дали даден стринг започва по определн начин

```
const str = "hello";
const test = "hel";

console.log(`${str} starts with "${test}": ${str.indexOf(test)
console.log(`${str} starts with "${test}": ${str.startsWith(te
```

STRING.PROTOTYPE.ENDSWITH

Проверява дали даден стринг завършва по определн начин

```
const str = "hello";
const test = "lo";

console.log(`${str} ends with "${test}": ${str.lastIndexOf(tes
console.log(`${str} ends with "${test}": ${str.endsWith(test)}
```

STRING.PROTOTYPE.INCLUDES

Проверява дали даден стринг съдържа подстринг

```
const str = "hello";
const test = "ll";

console.log(`${str} contains "${test}": ${str.indexOf(test) !=
console.log(`${str} contains "${test}": ${str.includes(test)}`
```

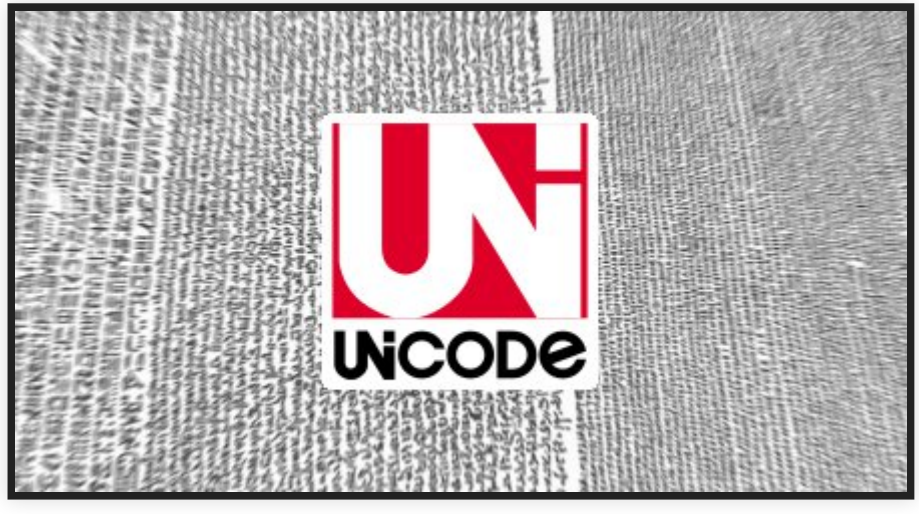
STRING.PROTOTYPE.REPEAT

Повтаря стринг n на брой пъти

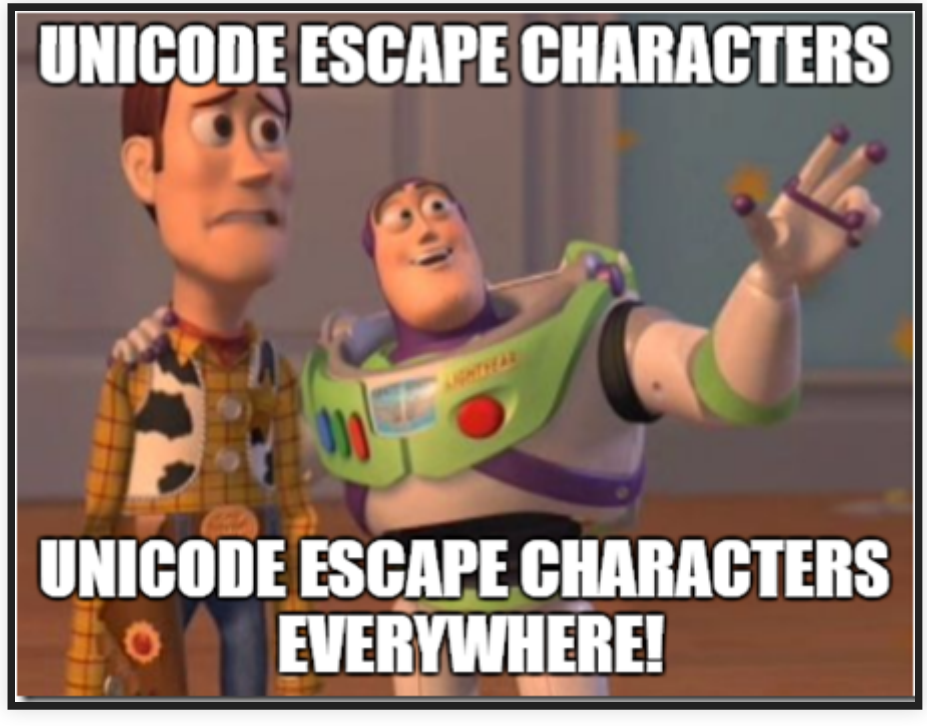
```
const str = "na";

console.log(Array.from(Array(4)).map(() => str).join(""));
console.log(str.repeat(4));
```

UNICODE



Unicode provides an unique number for every character,
no matter what the platform,
no matter what the program,
no matter what the language.
-*www.unicode.org*

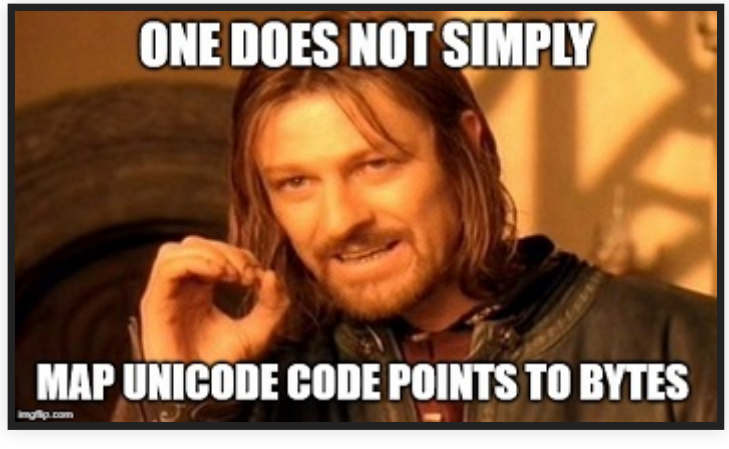

UNICODE ESCAPE CHARACTERS
UNICODE ESCAPE CHARACTERS
EVERYWHERE!

- Unicode е стандарт за кодиране на символи
- Unicode е стандарт за работа с голям диапазон от символи
- "Компютрите просто се справят с числа". Те съхраняват знаци и букви (символи), като присвояват номер за всеки от тях. Преди изобретяването на Unicode е имало стотици различни кодиращи системи(ASCII, ANSI) за присвояване на тези числа. Нито едно кодиране не е можело да съдържа достатъчно символи.

ПОПУЛЯРНИ ЕНКОДИНГИ UTF-8, UTF-16

КАКВО СА ЕНКОДИНГИТЕ?

- Всеки символ си има свое число - "code point" (\u00E9) и тези code points се конвертират към редица от битове (*енкоднати*) чрез различни *енкодинги*, за да бъдат "разбрани" и запазени от компютрите.



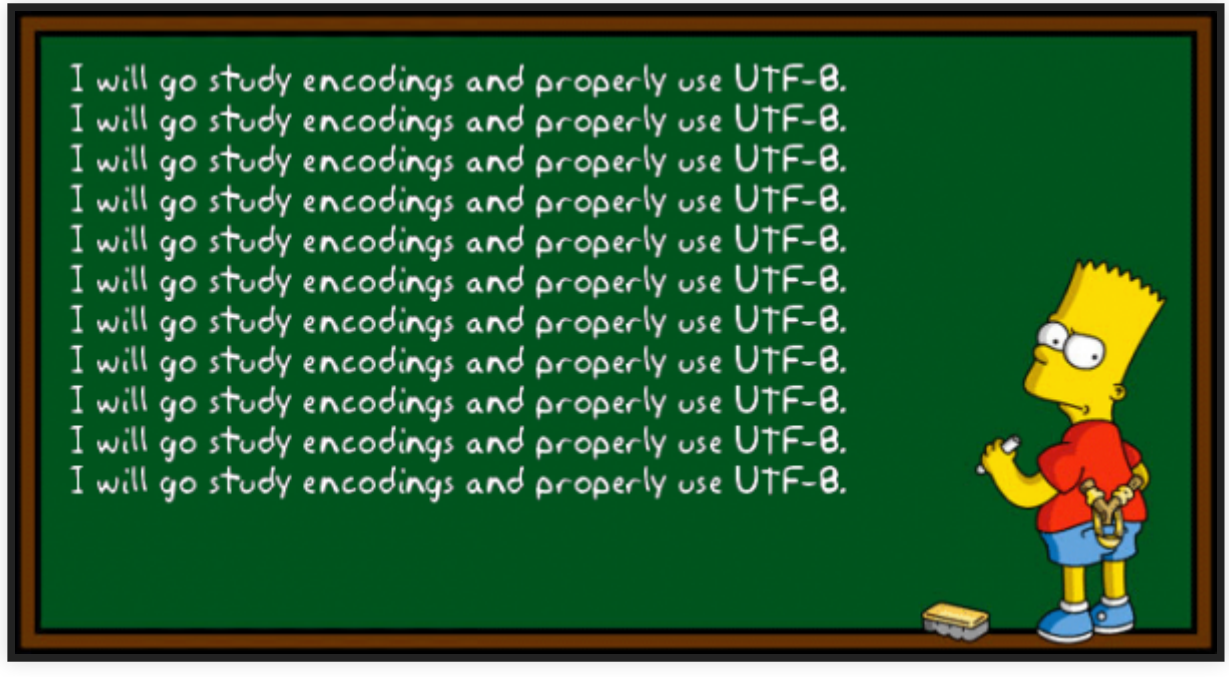
UTF-8

- UTF-8 е сред доминиращите енкодинги в Web. Използва 1байт за първите 128 code points и до 4 за останалите символи. Първите 128 Unicode code points репрезентират ASCII символите, от което следва, че всички ASCII текстове са също UTF-8 текстове

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

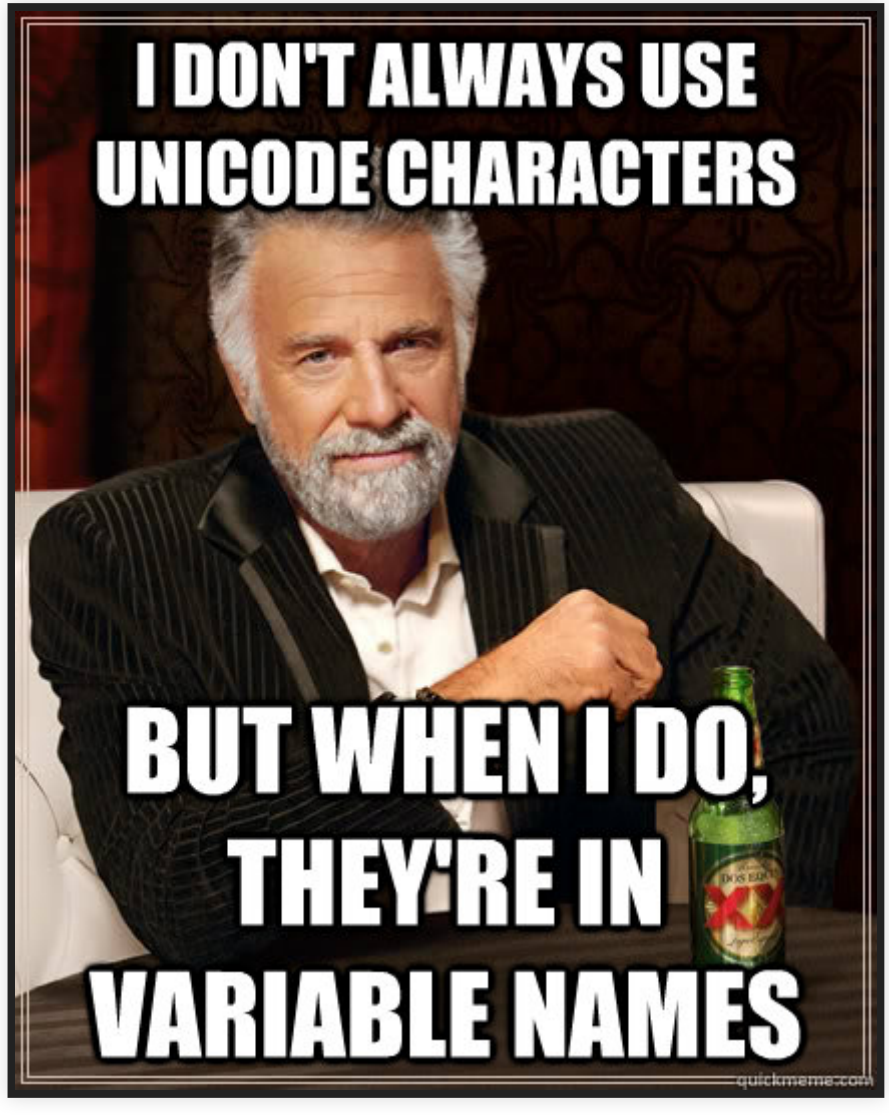
UTF-16

- UTF-16 е протокол, който използва code points с фиксираната дължина от 16 бита
- Той е по-добър там, където ASCII не преобладава, тъй като използва предимно 2 байта на символ. UTF-8 ще започне да използва 3 или повече байта за символите от по-висок ред, където UTF-16 остава само 2 байта за повечето знаци.


I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.

UNICODE В JAVASCRIPT

- Въпреки че JavaScript source файл може да има всеки енкодинг, JavaScript го конвентира вътрешно към UTF-16, преди да го изпълни
- Всички стрингове в JavaScript са UTF-16 поредици, наложено от ECMAScript стандарта, който казва:
  When a String contains actual textual data, each element is considered to be a single UTF-16 code unit


I DON'T ALWAYS USE
UNICODE CHARACTERS
BUT WHEN I DO,
THEY'RE IN
VARIABLE NAMES
quickmeme.com

UNICODE В JAVASCRIPT СТРИНГ

- Unicode поредица може да бъде добавена във всеки стринг, с формат `\uXXXX`:

```
const s1 = '\u00E9' //é
```

- Поредица може да се създаде и от 2 Unicode поредици

```
const s2 = '\u0065\u0301' //é
```

```
s1.length //1
s2.length //2
```

UNICODE В JAVASCRIPT СТРИНГ

- Emojis

```
const s4 = '🐶' // \uD83D\uDC36
```

- Някой emojis са създадени, чрез комбиниране н други:

```
💑 = \uD83D\uDC69\u200D\u2764\uFE0F\u200D\uD83D\uDC69

💑 = 👩 '\uD83D\uDC69' + ❤️ '\u200D\u2764\uFE0F\u200D' + 👩 '\u
```

MATH

ОБЕКТ MATH

- Позволява изпълняването на математически функции върху числа
- За разлика от други глобални обекти Math няма конструктор. Методите и пропъртитата са статични

ЧЕСТО ИЗПОЛЗВАНИ МЕТОДИ

- Math.min() and Math.max() Връщат съответно най-ниската и най-високата стойност от листа с аргументи

```
Math.min(0, 150, 30, 20, -8, -200);  // returns -200
Math.max(0, 150, 30, 20, -8, -200);  // returns 150
```

- Math.random() Връща произволно число от 0(включително) до 1(изключено)

```
Math.random();      // returns a random number
```

ОЩЕ МЕТОДИ

- Math.round() Закръглява числото на по-близкото до него

```
Math.round(4.7);     // returns 5
Math.round(4.4);     // returns 4
```

- Math.ceil() Закръглява нагоре числото

```
Math.ceil(4.4);      // returns 5
```

- Math.floor() Закръглява надолу числото

```
Math.floor(4.7);     // returns 4
```

MATH

ВЪПРОСИ?

БЛАГОДАРИМ ВИ!

names: "Филип Сидеров и Христо Петров"
emails: " filip.siderov@sap.com & h.petrov@sap.com "