



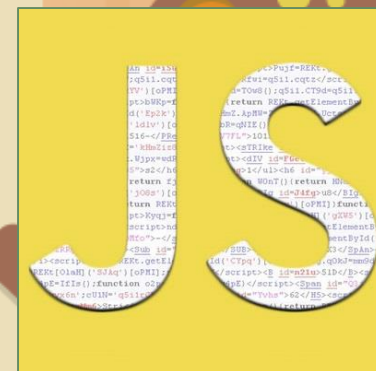
WEB ACADEMY

Front End за Начинаещи JavaScript

An illustration of a wooden desk with various art supplies. On the left, there is a paintbrush with a yellow tip and a brown handle, and a palette with yellow, green, and blue paint. On the right, there is a paintbrush with a yellow tip and a brown handle, and a palette with yellow, green, and blue paint. In the center, the word 'JavaScript' is written in a large, bold, green font. Below it, there is a code snippet in a smaller, green font. To the right of the code snippet, there is a yellow square with a large, stylized 'JS' logo inside it. The background is a light brown color with some green and blue circles scattered around.

JavaScript

```
<script>  
  document.write('Awesome!');  
</script>
```



За JavaScript

5

- JavaScript – интерпретаторен език за програмиране
- Създаден 4.12.1995 година от Netscape
- Изпълнява се в браузъра
- Езикът е чувствителен към регистъра
- Поддържа функционален и обектно ориентиран стил
- Няма нищо общо с Java освен приликата в името
- Поддържа едноредови коментари във формат
// Това е едноредов коментар и няма да се интерпретира от браузъра
- Поддържа многоредови коментари във формат
/* Това е многоредов коментар, който няма да се интерпретира от браузъра */
- Няколко оператора се групират посредством { и }



ВЪЗМОЖНОСТИ

6

- **Какви са възможностите на JavaScript:**
 - Зареждане на данни чрез AJAX
 - Ефекти с изображения и HTML елементи
 - Управление на прозорци и рамки.
 - Разпознаване на възможностите на браузъра.
 - Използване на камерата и микрофона.
 - Създаване на графики
- **Какво не може да прави JavaScript:**
 - Не може да записва информация на потребителския компютър
 - Не може да записва информация на отдалечения сървър
 - Не може да запазва информация директно в база данни.
 - Не може да се стартират локални приложения.



Вмъкване на JavaScript



- чрез деклариране на външен файл вътре в `<head>` тага
`<script src="file.js"></script>`
- чрез използване на `<script>` таг
`<script>`
`<!--`
 `alert('Здравейте');`
`-->`
`</script>`
- чрез директно задаване на JS код на ниво елемент
`<i onclick="alert('Здрасти');">Кликнете тук</i>`


Добрата практика изисква:

1. Функциите да се декларират чрез метод 1 или 2
2. На ниво елемент, да се прави извикване на вече дефинираните функции



Променливи



- Няколко думи за променливите:
 - Променливите са динамични и слаботипизирани
 - Декларират се чрез ключовата дума `var` следвана от името
 - Името на променливата може да съдържа букви, цифри, "_" и "\$" 
 - Името на променливата не може да започва с цифра
 - Името на променливата не може да бъде ключова дума или оператор
 - Възможно е присвояване на стойност, заедно с декларацията
 - По подразбиране всяка променлива има стойност `undefined`
- Ами константите? - JavaScript няма константи!

//Валидни имена на променливи: ninja, apples_and_oranges, var56
//Невалидни имена на променливи: 3, var, "test", if, while, 5wtx1

```
var firstVar; // Деклариране на променлива с име firstVar  
var secondVar = 1 // Деклариране и инициализиране на стойност
```



JavaScript Keywords



| | | | | |
|----------|-----------|------------|-------------------------------------|--------------|
| abstract | arguments | boolean | break | byte |
| case | catch | char | class* | const |
| continue | debugger | default | delete | do |
| double | else | enum* | eval | export* |
| extends* | false | final | finally | float |
| for | function | goto | if | implements |
| import* | in | instanceof | int | interface |
| let | long | native | new | null |
| package | private | protected | public | return |
| short | static | super* | switch | synchronized |
| this | throw | throws | transient | true |
| try | typeof | var | void | volatile |
| while | with | yield | <i><u>Виж още: пълен списък</u></i> | |



Типове данни



Типове данни поддържани от езика:

- числа - например 1, 2, 3, -3.14 и други
- низове - задават се в "кавички" или 'апострофи'
- логически (булеви) - true или false
- масиви
 - задават се чрез []
 - съдържат списък с елементи
 - елементите могат да бъдат от различни типове
 - възможно е да има вложени масиви
 - ключовете на масивите са задължително цели числа, в диапазона от 0..N-1, където N е броя на елементите

Типове данни

11

/ Примери за типове данни */*

var a = 3; *//Инициализиране на числова променлива:*

var b = "JavaScript iz kuwl"; *//Инициализиране на променлива с низ*

var c = **true**; *//Инициализиране на булева променлива*

//създаване на нов масив:

var empty = []; *//празен масив*

var my_data = [1, 2, 3, "hi", "bye", -2.11];

var array_of_arrays = [[1, 2, 3], [4, 5, 6], "anything else?"]; *//масив от масиви*

//Извеждане на типа на данните в променливите

alert(**typeof** a);

alert(**typeof** b);

alert(**typeof** c);



Camel Case



Historically, programmers have used three ways of joining multiple words into one variable name:

Hyphens:

first-name, last-name, master-card, inter-city.

Underscore:

first_name, last_name, master_card, inter_city.

Camel Case:

FirstName, LastName, MasterCard, InterCity.



In programming languages, especially in JavaScript, camel case often starts with a lowercase letter:

firstName, lastName, masterCard, interCity.



Въвеждане на данни



В JavaScript може да въвеждаме данни по няколко начина:

- чрез диалогов прозорец - `prompt(text, default_text);`
- чрез диалог за потвърждение - `confirm(text);`
- чрез взимане стойността на HTML елемент

```
<html>
<body>
  <input type="text" name="user_name" id="user_name" />
  <script>
    var confirm_value = confirm('Are you sure?');
    var prompt_value = prompt("Please, enter text", "Default Text...");
    var user_name_value = document.getElementById("user_name").value;
  </script>
</body>
</html>
```

- Виж още: [JavaScript HTML Input Examples](#)



Извеждане на данни



JavaScript може да извежда данни по няколко начина:

- чрез системно съобщение - `window.alert()`;
- вътре в HTML документ - `document.write()`;
- вътре в HTML елемент - `innerHTML`
- в конзолата на браузъра - `console.log()`

```
<html>
<body>
  <script>
    window.alert(1 + 2);
    document.write(3 + 4);
    document.getElementById("demo").innerHTML = 5 + 6;
  </script>
  <button onclick="document.write(7 + 8)">Try it</button>
</body>
</html>
```



Оператори



JavaScript работи със следните оператори:

- оператор за присвояване =
- празен оператор ;
- аритметични оператори

- събиране +
- изваждане -
- умножение *
- деление /
- деление по модул %
- увеличаване с 1 ++
- намаляване с 1 --

- комбинирани оператори: += , -= , *=, /=
- логически оператори: <, >, <=, >=, ==, !=, ===, !==, &&, ||

Примери:

```
var x = 5;      // assign the value 5 to x
var y = 2;      // assign the value 2 to y
var z = x + y;  // assign the value 7 to z
console.log(x % y);
```

```
txt1 = "John";
txt2 = "Doe";
txt3 = txt1 + " " + txt2;
```

```
x = 5 + 5;
y = "5" + 5;
z = "Hello" + 5;
```



Проверка на условие



В JavaScript има 2 основни начина за проверка на условие:

- if оператор

- синтаксис: `if (условие) { оператори; }else{ оператори; }`
- ако условието е истина, се изпълняват операторите
- ако условието е лъжа, се изпълняват `else операторите`
- скобите `{` и `}` са важни!

- троен оператор

- синтаксис: `(условие) ? оператори; : оператори;`
- ако условието е истина, се изпълняват операторите
- ако условието е лъжа, се изпълняват операторите

Проверка на условие



```
<html>
<body>
<p>Въведете години:</p><input id="age" value="18" />
<button onclick="checkMe()">Try it</button><p id="result"></p>
<script>
function checkMe() {
    var age, voteable;
    age = Number(document.getElementById("age").value);
    if (isNaN(age)) {
        voteable = "Грешни входни данни!";
    } else {
        voteable = (age < 18) ? "Много сте млад!" : "Добре влизай!";
    }
    document.getElementById("result").innerHTML = voteable;
}
</script>
</body>
</html>
```

Избор на вариант



switch конструкция

- Позволява избор на вариант измежду няколко
- Проверява за стойност и тип!!!
- Не забравяйте **break; !!!**
- Синтаксис:

```
switch(израз) {  
    case n:  
        code block  
        break;  
    case n:  
        code block  
        break;  
    default:  
        default code block  
}
```

Пример:

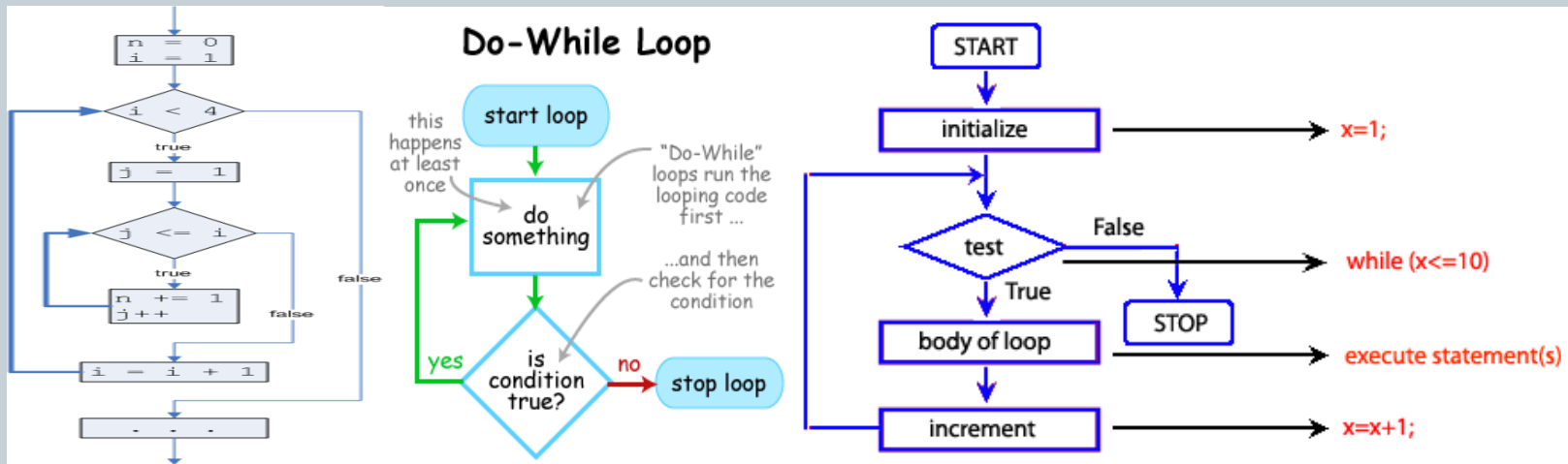
```
switch (new Date().getDay()) {  
    case 0:  
        day = "Неделя";  
        break;  
    case 1:  
        day = "Понеделник";  
        break;  
    case 2:  
        day = "Вторник";  
        break;  
    default:  
        day = "Непознат ден";  
}
```



Цикли



- оператор for - изпълнява се определен брой пъти
- оператор while - изпълнява се 0 или повече пъти
- оператор do..while - изпълнява се най-малко 1 път




Още по темата:


- http://www.w3schools.com/js/js_loop_for.asp
- http://www.w3schools.com/js/js_loop_while.asp

Често Срещани Грешки

20

-  Случайно използване на оператор за присвояване (=), вместо оператор за сравнение (==) може да доведе до неочаквани резултати!


```
var x = 0;  
if (x == 10){} // Коректно използване на оператор за сравнение  
if (x = 10){} // Връща true, защото присвоява на X стойност 10, а 10 > 0  
if (x = 0){} // Връща false, защото присвоява на X, стойност 0, а 0 = false
```

-  Случайно използване на оператор за сравнение (==), вместо оператор за идентичност (===)

```
var x = 10;  
var y = "10";  
if (x == y){} // Връща true, защото сравнява само стойността  
if (x === y){} // Връща false, защото сравнява стойността и типа
```

Често Срещани Грешки

21

 Грешно предположение, че конструкцията switch...case проверя за равни стойности, вместо за идентичност

// Запомнете: switch конструкцията проверява за идентичност, т.е.

```
var x = 10;  
switch(x) {  
  case 10: alert("Число"); break;  
  case "10": alert("Низ"); break;  
}
```

// Ще изведе съобщение за Число, защото проверява и стойност и тип

 Искате събиране, а се получава слепване (конкатенация)

// Оценяването на аргументите става от ляво надясно!

```
var x = 10 + 5;           // резултата ще бъде 15  
var x = 10 + "5";         // резултата ще бъде "105"  
var x = 10 + 6 + "29";    // резултата ще бъде "1629"
```

Често Срещани Грешки

22

- Неразбиране на това, как работят float числата

// Запомнете: бъдете внимателни при работа с числа с плаваща запетая

```
var x = 0.1;
```

```
var y = 0.2;
```

```
var z = x + y      // очаквате, че стойността на Z е 0.3, но грешите!!!
```

```
if (z == 0.3)      // връща false, защото стойността на Z е 0.3000000004
```

// За да решите проблема, използвайте следния малък трик :-)

```
var z = (x * 10 + y * 10) / 10;    // z will be 0.3
```

Грешно прекъснат низов литерал

```
var x =              // Работи коректно
```

```
"Hello World!";
```

```
var x = "Hello      // НЕ работи, ще изведе синтактична грешка!  
World!";
```


```
var x = "Hello \     // Работи коректно.
```

```
World!";            // Бъдете внимателни с този синтаксис! :)
```



Често Срещани Грешки

23

 Грешно предположение, че масивите са асоциативни. В много програмни езици масивите са асоциативни, но тук НЕ са, т.е. техните ключове винаги са цели числа

```
var person = [];  
person[0] = "John";  
person[1] = "Doe";  
person[2] = 46;  
var x = person.length; // връща 3  
var y = person[0];      // "John"  
// Коректен код
```

```
var person = [];  
person["firstName"] = "John";  
person["lastName"] = "Doe";  
person["age"] = 46;  
var x = person.length; // връща 0  
var y = person[0];      // undefined  
// НЕкоректен код
```

 Завършване на дефиницията на масив със запетая (,)

```
var points = [40, 100, 1, 5, 25, 10,]; // НЕкоректен код. НЕ пишете така!  
var points = [40, 100, 1, 5, 25, 10]; // Коректен код. Пишете го така!
```



Често Срещани Грешки

24

 Други трудно откриваеми грешки:

- Грешно поставен оператор ;
- Изпуснат оператор ;
- Символ изписан на кирилица, вместо на латиница
- Липсваща или неправилно поставена { скоба }
- Неразбиране на разликата между масиви и обекти
- Неразбиране на разликата между `undefined` и `null`.
`null` е за обекти, `undefined` е за променливи и методи.
- Семантични грешки, се откриват изключително трудно, след дълбоко и подробно тестване.
- Повече информация: [тук](#)

Функции



- Функциите представляват самостоятелни парчета програмен код, които решават определена задача.
- Всяка функция се състои от име, списък с формални (входни) параметри, тяло и изходни параметри.
- Параметрите се разделят помежду си със запетая ","
- Функцията се дефинира с формални параметри, и се извиква (инвокира) с реални аргументи
- Функция, която не връща стойност се нарича още void функция, а в някои езици се нарича също така процедура

- Синтаксис:

```
function toCelsius(fahrenheit) {  
    return (5/9) * (fahrenheit-32);  
}  
document.getElementById("d").innerHTML = toCelsius(77);
```



Функции



Защо да използваме функции:

- кодът е преизползваем
- пишете веднъж използвате многократно
- различни входни параметри дават различен резултат
- ако е нужна промяна, променяте само на едно място

Задачи за самоподготовка върху функции:

- извикайте функцията
- поправете функцията
- допишете тялото на функцията
- извикайте функцията с параметри
- дефинирайте функцията myFunction, която извежда текст

Повече информация: [тук](#)




```
var jsObject = {  
  name: "Caitlyn",  
  age: 24,  
  favorite_foods: ['pizza', 'subway sandwiches', 'soup'],  
  quirk: "I hate bunched up seeds. They look like larvae."  
};
```

Обекти (Въведение)



- Обектите представляват съвкупност от свойства и методи
- Свойствата представляват променливи за обекта
- Методите представляват функции за обекта
- JS обектите са асоциативни (за разлика от масивите)

| Object | Properties | Methods |
|--|--|--|
|  | <p>car.name = Fiat</p> <p>car.model = 500</p> <p>car.weight = 850kg</p> <p>car.color = white</p> | <p>car.start()</p> <p>car.drive()</p> <p>car.brake()</p> <p>car.stop()</p> |

Пример за обект кола (car)

```
var jsObject = {  
  name: "Caitlyn",  
  age: 24,  
  favorite_foods: ['pizza', 'subway sandwiches', 'soup'],  
  quirk: "I hate bunched up seeds. They look like larvae."  
};
```

Обекти (Синтаксис)



- Как се дефинира обект в JavaScript:

- името на свойството / метода НЕ се загражда в кавички / апострофи
- отделните свойства / методи се разделят помежду си със , (запетая)
- между името на свойството / метода и неговата стойност се слага :
- цялата дефиниция на обекта се загражда във { } (фигурни скоби)
- след затварящата } (фигурна скоба) се слага ; (точка и запетая)
- можете да се обръщате към свойствата на обекта по 2 начина:
objectName.propertyName или *objectName["propertyName"]*
- **НЕ** създавайте инстанции на обектите String, Number, Boolean, тъй като това ще се отрази негативно върху бързодействието

// Това може да бъде потенциален проблем. НЕ пишете така!

```
var x = "John";  
var y = new String("John");
```

// (x === y) е false, x и y имат различни типове (string и object)

```
var jsObject = {  
  name: "Caitlyn",  
  age: 24,  
  favorite_foods: ['pizza', 'subway sandwiches', 'soup'],  
  quirk: "I hate bunched up seeds. They look like larvae."  
};
```

Обекти (Примери)



<p id="demo"></p>

<script>

var d = new Date(); // Date е вграден в езика обект. Очаквайте повече скоро

var person = {

firstName: "Yordan",

lastName: "Enev",

age : 29,

favmovies: ['The Lord of the Rings', 'The Hobbit', 'The Notebook'],

fullName: function() {

return this.firstName + " " + this.lastName;

}

};

document.getElementById("demo").innerHTML = person.fullName();

alert('Рождена година: ' + (d.getFullYear() - person.age));

</script>



Методи за работа с данни



JavaScript има много и различни методи за работа с данни. Най-общо те могат да бъдат групирани в няколко обекта:

- Math обект - за работа с математически операции
- Date обект - за работа с дати
- Array обект - за работа с масиви
- String обект - за работа с низове
- RegExp обект - за работа с регулярни изрази

Внимание: Не създавайте прекалено много обекти, защото това се отразява зле на бързодействието и може да доведе до неочаквани последици...

Math обект



Math обектът ви позволява да работите с аритметични операции и методи. Той няма конструктор, и всички обръщания стават през него, без да се създава явно негова инстанция.

| Property | Description | Константи на Math обекта |
|----------------|---|--------------------------|
| <u>E</u> | Returns Euler's number (approx. 2.718) | |
| <u>LN2</u> | Returns the natural logarithm of 2 (approx. 0.693) | |
| <u>LN10</u> | Returns the natural logarithm of 10 (approx. 2.302) | |
| <u>LOG2E</u> | Returns the base-2 logarithm of E (approx. 1.442) | |
| <u>LOG10E</u> | Returns the base-10 logarithm of E (approx. 0.434) | |
| <u>PI</u> | Returns PI (approx. 3.14) | |
| <u>SQRT1_2</u> | Returns the square root of 1/2 (approx. 0.707) | |
| <u>SQRT2</u> | Returns the square root of 2 (approx. 1.414) | |



Методи на Math обекта

| Method | Description |
|-------------------------------|---|
| <code>abs(x)</code> | Returns the absolute value of x |
| <code>acos(x)</code> | Returns the arccosine of x, in radians |
| <code>asin(x)</code> | Returns the arcsine of x, in radians |
| <code>atan(x)</code> | Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians |
| <code>atan2(y,x)</code> | Returns the arctangent of the quotient of its arguments |
| <code>ceil(x)</code> | Returns the value of x rounded up to its nearest integer |
| <code>cos(x)</code> | Returns the cosine of x (x is in radians) |
| <code>exp(x)</code> | Returns the value of E^x |
| <code>floor(x)</code> | Returns the value of x rounded down to its nearest integer |
| <code>log(x)</code> | Returns the natural logarithm (base E) of x |
| <code>max(x,y,z,...,n)</code> | Returns the number with the highest value |
| <code>min(x,y,z,...,n)</code> | Returns the number with the lowest value |
| <code>pow(x,y)</code> | Returns the value of x to the power of y |
| <code>random()</code> | Returns a random number between 0 and 1 |
| <code>round(x)</code> | Returns the value of x rounded to its nearest integer |
| <code>sin(x)</code> | Returns the sine of x (x is in radians) |
| <code>sqrt(x)</code> | Returns the square root of x |
| <code>tan(x)</code> | Returns the tangent of an angle |



Примери за работа с Math

33

Опитайте следните примери:

| | |
|------------------------------------|--|
| Math.PI; | // връща стойността на константата Pi |
| Math.round(4.7); | // връща 5 |
| Math.round(4.4); | // връща 4 |
| Math.pow(8,2); | // връща 64 |
| Math.sqrt(64); | // връща 8 |
| Math.abs(-4.7); | // връща 4.7 |
| Math.ceil(4.4); | // връща 5 закръгля нагоре до цяло число |
| Math.floor(4.7); | // връща 4 закръгля надолу до цяло число |
| Math.sin(90 * Math.PI / 180); | // връща 1 (синус на 90 градуса) |
| Math.cos(0 * Math.PI / 180); | // връща 1 (косинус на 0 градуса) |
| Math.min(0, 150, 30, 20, 8, -200); | // връща -200 |
| Math.max(0, 150, 30, 20, 8, -200); | // връща 150 |
| Math.random(); | // връща случайна стойност между [0,1) !!! |

Още по темата: [тук](#)



Случайни числа с Math

34

Опитайте следните примери:

```
Math.random();           // връща случайна стойност между [0,1)
Math.floor(Math.random() * 10); // връща случайно число между 0 и 9
Math.floor(Math.random() * 11); // връща случайно число между 0 и 10
Math.floor(Math.random() * 10) + 1; // връща случайно число между 1 и 10
```

```
// Функцията връща случайно число в интервала [min, max)
function getRndInteger(min, max) {
    return Math.floor(Math.random() * (max - min) ) + min;
}
```

```
// Функцията връща случайно число в интервала [min, max]
function getRndInteger(min, max) {
    return Math.floor(Math.random() * (max - min + 1) ) + min;
}
```

Още по темата: [тук](#)



Date обект



Date обектът ви позволява да работите с дати.

Има 4 начина за инициализация на Date обекта:

- new Date()
- new Date(milliseconds)
- new Date(dateString)
- new Date(year, month, day, hours, minutes, seconds, milliseconds)

Датата може да се задава като низ, в следните 4 формата:

| Type | Example |
|------------|---|
| ISO Date | "2015-03-25" (The International Standard) |
| Short Date | "03/25/2015" or "2015/03/25" |
| Long Date | "Mar 25 2015" or "25 Mar 2015" |
| Full Date | "Wednesday March 25 2015" |



Методи на Date обекта



Често използвани методи за взимане на дата:

| Method | Description |
|-------------------|---|
| getDate() | Get the day as a number (1-31) |
| getDay() | Get the weekday as a number (0-6) |
| getFullYear() | Get the four digit year (yyyy) |
| getHours() | Get the hour (0-23) |
| getMilliseconds() | Get the milliseconds (0-999) |
| getMinutes() | Get the minutes (0-59) |
| getMonth() | Get the month (0-11) |
| getSeconds() | Get the seconds (0-59) |
| getTime() | Get the time (milliseconds since January 1, 1970) |



Методи на Date обекта



Често използвани методи за задаване на дата:

| Method | Description |
|-------------------|---|
| setDate() | Set the day as a number (1-31) |
| setFullYear() | Set the year (optionally month and day) |
| setHours() | Set the hour (0-23) |
| setMilliseconds() | Set the milliseconds (0-999) |
| setMinutes() | Set the minutes (0-59) |
| setMonth() | Set the month (0-11) |
| setSeconds() | Set the seconds (0-59) |
| setTime() | Set the time (milliseconds since January 1, 1970) |

Вижте още: [пълен списък с методи на Date обекта](#)



RegExp обект



RegExp обектът служи за работа с регулярни изрази. Регулярният израз е поредица от символи, която образува т. нар. модел за търсене. Този модел за търсене може да се използва в операции за търсене и заместване на текст.

```
var str = "Visit W3Schools!";  
var res = str.replace(/w3schools/i, "WebAcademy");  
var n    = str.search(/w3s/i); // n = 6; Броенето започва от нула :-)
```

Пример за лесен RegExp

- Синтаксис: */регулярен израз (pattern)/модификатори*
- Методи за работа с RegExp:

| Method | Description |
|----------------------------|--|
| exec() | Tests for a match in a string. Returns the first match |
| test() | Tests for a match in a string. Returns true or false |
| toString() | Returns the string value of the regular expression |



RegExr обект



- **Модификатори:**

- Служат за задаване на различни опции
- Списък с често използвани модификатори

| Modifier | Description |
|----------|--|
| i | Perform case-insensitive matching |
| g | Perform a global match (find all matches rather than stopping after the first match) |
| m | Perform multiline matching |

- **Набор от символи**

| Expression | Description |
|------------|---|
| [abc] | Find any of the characters between the brackets |
| [0-9] | Find any of the digits between the brackets |
| (x y) | Find any of the alternatives separated with |

- символа ^ задава отрицание



RegExr обект



- **Метасимволи:**

- Представяват символи със специално значение
- Списък с често използвани метасимволи (вижте всички)

| Metacharacter | Description |
|---------------|--|
| . | Find a single character, except newline or line terminator |
| \w | Find a word character |
| \d | Find a digit |
| \s | Find a whitespace character |
| \b | Find a match at the beginning/end of a word |
| \0 | Find a NUL character |
| \n | Find a new line character |
| \f | Find a form feed character |
| \r | Find a carriage return character |
| \t | Find a tab character |

- Главните букви указват отрицание. Например \D - не цифра



RegExp обект



- Модифициране на количество
 - Задава се чрез така наречените quantifiers
 - Списък с всички модификатори на количество

| Quantifier | Description |
|------------------------------|--|
| <u>n^+</u> | Matches any string that contains at least one n |
| <u>n^*</u> | Matches any string that contains zero or more occurrences of n |
| <u>$n?$</u> | Matches any string that contains zero or one occurrences of n |
| <u>$n\{X\}$</u> | Matches any string that contains a sequence of X n 's |
| <u>$n\{X,Y\}$</u> | Matches any string that contains a sequence of X to Y n 's |
| <u>$n\{X, \}$</u> | Matches any string that contains a sequence of at least X n 's |
| <u>$n\\$</u> | Matches any string with n at the end of it |
| <u>n</u> | Matches any string with n at the beginning of it |
| <u>$?=n$</u> | Matches any string that is followed by a specific string n |
| <u>$?!n$</u> | Matches any string that is not followed by a specific string n |



Методи на Array обекта

| Method | Description |
|--------------------------------------|--|
| <u>concat()</u> | Joins two or more arrays, and returns a copy of the joined arrays |
| <u>copyWithin()</u> | Copies array elements within the array, to and from specified positions |
| <u>every()</u> | Checks if every element in an array pass a test |
| <u>fill()</u> | Fill the elements in an array with a static value |
| <u>filter()</u> | Creates a new array with every element in an array that pass a test |
| <u>find()</u> | Returns the value of the first element in an array that pass a test |
| <u>findIndex()</u> | Returns the index of the first element in an array that pass a test |
| <u>forEach()</u> | Calls a function for each array element |
| <u>indexOf()</u> | Search the array for an element and returns its position |
| <u>isArray()</u> | Checks whether an object is an array |
| <u>join()</u> | Joins all elements of an array into a string |
| <u>lastIndexOf()</u> | Search the array for an element, starting at the end, and returns its position |
| <u>map()</u> | Creates a new array with the result of calling a function for each array element |



Методи на Array обекта

| Method | Description |
|----------------------|--|
| <u>pop()</u> | Removes the last element of an array, and returns that element |
| <u>push()</u> | Adds new elements to the end of an array, and returns the new length |
| <u>reduce()</u> | Reduce the values of an array to a single value (going left-to-right) |
| <u>reduceRight()</u> | Reduce the values of an array to a single value (going right-to-left) |
| <u>reverse()</u> | Reverses the order of the elements in an array |
| <u>shift()</u> | Removes the first element of an array, and returns that element |
| <u>slice()</u> | Selects a part of an array, and returns the new array |
| <u>some()</u> | Checks if any of the elements in an array pass a test |
| <u>sort()</u> | Sorts the elements of an array |
| <u>splice()</u> | Adds/Removes elements from an array |
| <u>toString()</u> | Converts an array to a string, and returns the result |
| <u>unshift()</u> | Adds new elements to the beginning of an array, and returns the new length |
| <u>valueOf()</u> | Returns the primitive value of an array |



Методи на Array обекта



```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];

var citrus = fruits.slice(1,3); // Връща елементите с индекс в интервал [1,3)
var citrus = fruits.slice(2);   // Връща елементите с индекс >= 2
fruits.splice(0, 2);           // Премахва първите 2 елемента на масива
fruits.splice(2, 0, "Kiwi");    // Добавя елемента "Kiwi" на втора позиция
fruits[fruits.length] = "owl"; // Добавя елемента "owl" на последна позиция
fruits.unshift("Kiwi");        // Добавя елемента "Kiwi" като първи елемент
fruits.shift();                // Премахва и връща първия елемент на масива
fruits.push("Kiwi");           // Добавя елемента "Kiwi" на последна позиция
var x = fruits.pop();          // Премахва и връща последния елемент
fruits.join(" * ");            // Слеща елементите на масива с указания низ

var child1 = ["Чанита", "Попова"];
var child2 = ["Кремена", "Желева"];
var WA_Children = fruits.concat(child1, child2);
```



Методи на String обекта

| Method | Description |
|--|--|
| <u>charAt()</u> | Returns the character at the specified index (position) |
| <u>charCodeAt()</u> | Returns the Unicode of the character at the specified index |
| <u>concat()</u> | Joins two or more strings, and returns a new joined strings |
| <u>endsWith()</u> | Checks whether a string ends with specified string/characters |
| <u>fromCharCode()</u> | Converts Unicode values to characters |
| <u>includes()</u> | Checks whether a string contains the specified string/characters |
| <u>indexOf()</u> | Returns the position of the first found occurrence of a specified value in a string |
| <u>lastIndexOf()</u> | Returns the position of the last found occurrence of a specified value in a string |
| <u>localeCompare()</u> | Compares two strings in the current locale |
| <u>match()</u> | Searches a string for a match against a regular expression, and returns the matches |
| <u>repeat()</u> | Returns a new string with a specified number of copies of an existing string |
| <u>replace()</u> | Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced |
| <u>search()</u> | Searches a string for a specified value, or regular expression, and returns the position of the match |



Методи на String обекта

| Method | Description |
|--|---|
| <u>search()</u> | Searches a string for a specified value, or regular expression, and returns the position of the match |
| <u>slice()</u> | Extracts a part of a string and returns a new string |
| <u>split()</u> | Splits a string into an array of substrings |
| <u>startsWith()</u> | Checks whether a string begins with specified characters |
| <u>substr()</u> | Extracts the characters from a string, beginning at a specified start position, and through the specified number of character |
| <u>substring()</u> | Extracts the characters from a string, between two specified indices |
| <u>toLocaleLowerCase()</u> | Converts a string to lowercase letters, according to the host's locale |
| <u>toLocaleUpperCase()</u> | Converts a string to uppercase letters, according to the host's locale |
| <u>toLowerCase()</u> | Converts a string to lowercase letters |
| <u>toString()</u> | Returns the value of a String object |
| <u>toUpperCase()</u> | Converts a string to uppercase letters |
| <u>trim()</u> | Removes whitespace from both ends of a string |
| <u>valueOf()</u> | Returns the primitive value of a String object |



Събития



- Събитията (events) възникват, когато нещо се случи, а JavaScript от своя страна реагира на тези събития
- Свързват се с конкретен елемент от страницата

```
<button onclick="this.innerHTML=Date()">The time is?</button>
```

- Най-често използвани събития:

| Event | Description |
|-------------|--|
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |





Какво ни е нужно?

- Намиране на HTML елементи
- Промяна на HTML елементи
- Добавяне и премахване на елементи

Вижте още:

- JavaScript HTML DOM Document
- JavaScript HTML DOM Elements
- CSS-JS References



Намиране на HTML елементи

- по id атрибут
- по име на таг
- по име на клас
- по CSS селектор
- по колекция от обекти

```
var myElement = document.getElementById("main");  
var x = document.getElementsByTagName("p");  
var y = document.getElementsByClassName("intro");  
var z = document.querySelectorAll("p.intro");
```

```
var x = document.forms["frm1"];  
var text = "";  
var i;  
for (i = 0; i < x.length; i++) {  
    text += x.elements[i].value + "<br>";  
}  
document.getElementById("demo").innerHTML = text;
```

First name:

Last name:

Donald
Duck
Submit



Промяна на HTML елементи е възможна чрез:

- HTML съдържание
- име на атрибут
- метод `setAttribute`
- свойство `style`

```
element.innerHTML = new html content;
```

```
element.attribute = new value;
```

```
element.setAttribute(attribute, value);
```

```
element.style.property = new style; // Псевдо код
```

```
<div id="main"></div>
```

```
<script>
```

```
var element = document.getElementById('main');
```

```
element.innerHTML = "new html content";
```

```
element.title = "Change Title Attribute";
```

```
element.setAttribute("title", "New Title Again");
```

```
element.style.border = "1px solid blue";
```

```
</script>
```

// Примерен код



Методи за добавяне и премахване на елементи:

- `document.createElement(element)` - създава елемент
- `document.removeChild(element)` - премахва елемент
- `document.appendChild(element)` - добавя елемент
- `document.replaceChild(element)` - замества елемент

```
<input type="button" onclick="addNew();" value="+"><ol id="my_list"></ol>
<script>
var i = 1;
function addNew(){
    var element = document.createElement('li');
    element.id= "item_"+i;
    element.innerHTML = 'New Item '+i;
    document.getElementById('my_list').appendChild(element); i++;
}</script> <!-- Динамично добавяне на нов елемент към списък -->
```

CSS - JS References



| CSS Property | JavaScript Reference |
|-----------------------|----------------------|
| background | background |
| background-attachment | backgroundAttachment |
| background-color | backgroundColor |
| background-image | backgroundImage |
| background-position | backgroundPosition |
| background-repeat | backgroundRepeat |
| border | border |
| border-bottom | borderBottom |
| border-bottom-color | borderBottomColor |
| border-bottom-style | borderBottomStyle |
| border-bottom-width | borderBottomWidth |
| border-color | borderColor |
| border-left | borderLeft |
| border-left-color | borderLeftColor |
| border-left-style | borderLeftStyle |
| border-left-width | borderLeftWidth |
| border-right | borderRight |
| border-right-color | borderRightColor |
| border-right-style | borderRightStyle |

| CSS Property | JavaScript Reference |
|--------------------|----------------------|
| border-right-width | borderRightWidth |
| border-style | borderStyle |
| border-top | borderTop |
| border-top-color | borderTopColor |
| border-top-style | borderTopStyle |
| border-top-width | borderTopWidth |
| border-width | borderWidth |
| clear | clear |
| clip | clip |
| color | color |
| cursor | cursor |
| display | display |
| filter | filter |
| font | font |
| font-family | fontFamily |
| font-size | fontSize |
| font-variant | fontVariant |
| font-weight | fontWeight |
| height | height |



CSS - JS References



| CSS Property | JavaScript Reference |
|---------------------|----------------------|
| left | left |
| letter-spacing | letterSpacing |
| line-height | lineHeight |
| list-style | listStyle |
| list-style-image | listStyleImage |
| list-style-position | listStylePosition |
| list-style-type | listStyleType |
| margin | margin |
| margin-bottom | marginBottom |
| margin-left | marginLeft |
| margin-right | marginRight |
| margin-top | marginTop |
| overflow | overflow |
| padding | padding |
| padding-bottom | paddingBottom |
| padding-left | paddingLeft |
| padding-right | paddingRight |
| padding-top | paddingTop |
| page-break-after | pageBreakAfter |

| CSS Property | JavaScript Reference |
|-------------------------------|---------------------------|
| page-break-before | pageBreakBefore |
| position | position |
| float | cssFloat |
| text-align | textAlign |
| text-decoration | textDecoration |
| text-decoration: blink | textDecorationBlink |
| text-decoration: line-through | textDecorationLineThrough |
| text-decoration: none | textDecorationNone |
| text-decoration: overline | textDecorationOverline |
| text-decoration: underline | textDecorationUnderline |
| text-indent | textIndent |
| text-transform | textTransform |
| top | top |
| vertical-align | verticalAlign |
| visibility | visibility |
| width | width |
| z-index | zIndex |
| | |
| | |



Задача 1



- Чрез JavaScript код генерирайте шахматна дъска.

Жокер:

- За визуално представяне на дъската можете да ползвате както дивове, така и таблици

От какво се нуждаете?

- 1-2 цикъла For
- доза логическо мислене
- щипка старание
- увереност при делението по модул :-)
- 5-10 минути върху клавиатурата



Задача 2



- Направете "команден пулт" за управление на "совалка"

Жокер:

- Нека "совалката" да е изображение.
- Направете 4 бутона с които да движите "совалката"
- Совалката се движи чрез смяна на позицията си

От какво се нуждаете?

- 1 изображение със совалка
- 4-5 бутона
- доза опит с координатни системи
- щипка старание + свидетелство за управление на НЛО :-)
- 15-20 минути върху клавиатурата



Задача 3



- Направете форма за добавяне на готварска рецепта.

Жокер:

- Продуктите трябва да могат да се добавят автоматично
- За всеки продукт трябва да имаме име и количество

От какво се нуждаете?

- 1 текстово поле за кратък текст (заглавие)
- 1 текстово поле за дълъг текст (описание)
- 1 таблица с необходимите продукти
- опит в готварството и смелост да пишете `append`
- 50-60 минути печене на бавен огън върху клавиатурата



ВЪПРОСИ

58



Исползвана литература

59

- http://www.w3schools.com/js/js_best_practices.asp
- http://www.w3schools.com/js/js_comparisons.asp
- http://www.w3schools.com/js/js_if_else.asp
- http://www.w3schools.com/js/js_loop_for.asp
- http://www.w3schools.com/js/js_loop_while.asp
- http://www.w3schools.com/js/js_mistakes.asp
- http://www.w3schools.com/js/js_output.asp
- http://www.w3schools.com/js/js_switch.asp
- http://www.w3schools.com/js/js_syntax.asp
- http://www.w3schools.com/js/js_variables.asp
- http://www.w3schools.com/jsref/jsref_operators.asp
- http://www.w3schools.com/jsref/prop_text_value.asp
- http://www.w3schools.com/js/js_timing.asp



Използвана литература

60

- http://www.w3schools.com/js/js_string_methods.asp
- http://www.w3schools.com/js/js_number_methods.asp
- http://www.w3schools.com/js/js_math.asp
- http://www.w3schools.com/js/js_dates.asp
- http://www.w3schools.com/js/js_date_formats.asp
- http://www.w3schools.com/js/js_date_methods.asp
- http://www.w3schools.com/js/js_array_methods.asp
- http://www.w3schools.com/js/js_arrays.asp
- http://www.w3schools.com/js/js_strict.asp
- http://www.w3schools.com/js/js_json.asp
- <http://www.w3schools.com/json/default.asp>
- http://www.w3schools.com/js/js_window_screen.asp
- http://www.w3schools.com/js/js_window_history.asp



Поздравления!

61



Този модул приключи!

Желаем Ви късмет с финалния изпит!

Вече знаете, какво е HTML & CSS & JS!

HTML = How To Meet Ladies

CSS = Countless Sex Styles

JS = Just Sex