Vista | Página de documentación técnica: Construye una página de documentación técnica | freeCodeCamp.org - Google Chrome

about:blank

JS Documentation

- Introduction
- What you should already know
- JavaScript and Java
- Hello World
- Variables
- Declaring variables
- Variable scope
- Global variables
- Constants
- Data types
- if...else statement
- While statement
- Function declarations
- Reference

Introduction

JavaScript is a cross-platform, object-oriented scripting language. It is a small and lightweight language. Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them. JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects; for example:

- Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
- Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

What you should already know

This guide assumes you have the following basic background:

- A general understanding of the Internet and the World Wide Web (WWW).
- Good working knowledge of HyperText Markup Language (HTML).
- Some programming experience. If you are new to programming, try one of the tutorials linked on the main page about JavaScript.

JavaScript and Java

JavaScript and Java are similar in some ways but fundamentally different in some others. The JavaScript language resembles Java but does not have Java's static typing and strong type checking. JavaScript follows most Java expression syntax, naming conventions and basic control-flow constructs which was the reason why it was renamed from LiveScript to JavaScript. In contrast to Java's compile-time system of classes built by declarations, JavaScript supports a runtime system based on a small number of data types representing numeric, Boolean, and string values. JavaScript has a prototype-based object model instead of the more common class-based object model. The prototype-based model provides dynamic inheritance; that is, what is inherited can vary for individual objects. JavaScript also supports functions without any special declarative requirements. Functions can be properties of objects, executing as loosely typed methods. JavaScript is a very free-form language compared to Java. You do not have to declare all variables, classes, and methods. You do not have to be concerned with whether methods are public, private, or protected, and you do not have to implement interfaces. Variables, parameters, and function return types are not explicitly typed.

-

Hello World

To get started with writing JavaScript, open the Scratchpad and write your first "Hello world" JavaScript code: function greetMe(yourName) { alert("Hello " + yourName); } greetMe("World"); Select the code in the pad and hit Ctrl+R to watch it unfold in your browser!

Variables

You use variables as symbolic names for values in your application. The names of variables, called identifiers, conform to certain rules. A JavaScript identifier must start with a letter, underscore (_), or dollar sign ($); subsequent characters can also be digits (0-9). Because JavaScript is case sensitive, letters include the characters "A" through "Z" (uppercase) and the characters "a" through "z" (lowercase). You can use ISO 8859-1 or Unicode letters such as å and ü in identifiers. You can also use the Unicode escape sequences as characters in identifiers. Some examples of legal names are Number_hits, temp99, and _name.

Declaring variables

INDEX.HTML (Página de Documentación 11)

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <main id="main-doc">
    <nav id="navbar">
    <header>JS Documentation</header>
      <a class="nav-link" href="#Introduction">Introduction</a>
      <a class="nav-link" href="#What_you_should_already_know">What you should already know</a>
      <a class="nav-link" href="#JavaScript_and_Java">JavaScript and Java</a>
      <a class="nav-link" href="#Hello_World">Hello World</a>
      <a class="nav-link" href="#Variables">Variables</a>
      <a class="nav-link" href="#Declaring_variables">Declaring variables</a>
      <a class="nav-link" href="#Variable_scope">Variable scope</a>
      <a class="nav-link" href="#Global_variables">Global variables</a>
      <a class="nav-link" href="#Constants">Constants</a>
      <a class="nav-link" href="#Data_types">Data types</a>
      <a class="nav-link" href="#if...else_statement">if...else statement</a>
      <a class="nav-link" href="#While_statement">While statement</a>
      <a class="nav-link" href="#Function_declarations">Function declarations</a>
      <a class="nav-link" href="#Reference">Reference</a>
      </nav>
    <section class="main-section" id="Introduction">
      <header>Introduction</header>
      <p>JavaScript is a cross-platform, object-
        oriented scripting language. It is a small and lightweight language. Inside a host environment (for example, a w
        eb browser), JavaScript can be connected to the objects of its environment to provide programmatic control over
        them.
```

JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language ele
ments such as operators, control structures, and statements. Core JavaScript can be extended for a variety of pu
rposes by supplementing it with additional objects; for example:

```
<ul>
    <li>Client-
    side JavaScript extends the core language by supplying objects to control a browser and its Document Object Mode
    l (DOM). For example, client-
    side extensions allow an application to place elements on an HTML form and respond to user events such as mouse
    clicks, form input, and page navigation.</li>
</ul>


<ul>
    <li>Server-
    side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For e
    xample, server-
    side extensions allow an application to communicate with a database, provide continuity of information from one
    invocation to another of the application, or perform file manipulations on a server.</li>
</ul>
</p>
<code></code>
</section>

<section class="main-section"id="What_you_should_already_know">
<header> What you should already know</header>
<p>This guide assumes you have the following basic background:


<ul>
    <li>A general understanding of the Internet and the World Wide Web (WWW).</li>
</ul>


<ul>
    <li>Good working knowledge of HyperText Markup Language (HTML).</li>
</ul>
```

```html
    <ul>
        <li>Some programming experience. If you are new to programming, try one of the tutorials linked on the main pa
        ge about JavaScript.</li>
    </ul></p>
</section>

<section class="main-section"id="JavaScript_and_Java">
    <header>JavaScript and Java</header>
    <p>JavaScript and Java are similar in some ways but fundamentally different in some others. The JavaScript languag
      e resembles Java but does not have Java's static typing and strong type checking. JavaScript follows most Java e
      xpression syntax, naming conventions and basic control-
      flow constructs which was the reason why it was renamed from LiveScript to JavaScript.

      In contrast to Java's compile-
      time system of classes built by declarations, JavaScript supports a runtime system based on a small number of da
      ta types representing numeric, Boolean, and string values. JavaScript has a prototype-
      based object model instead of the more common class-based object model. The prototype-
      based model provides dynamic inheritance; that is, what is inherited can vary for individual objects. JavaScript
       also supports functions without any special declarative requirements. Functions can be properties of objects, e
      xecuting as loosely typed methods.

      JavaScript is a very free-
      form language compared to Java. You do not have to declare all variables, classes, and methods. You do not have
      to be concerned with whether methods are public, private, or protected, and you do not have to implement interfa
      ces. Variables, parameters, and function return types are not explicitly typed.

    </p>
    <code></code>
    <li></li>
</section>

<section class="main-section" id="Hello_World">
    <header>Hello World</header>
    <p>To get started with writing JavaScript, open the Scratchpad and write your first "Hello world" JavaScript code:
      function greetMe(yourName) { alert("Hello " + yourName); }
      greetMe("World");
```

```html
          Select the code in the pad and hit Ctrl+R to watch it unfold in your browser!

    </p>
    <code></code>

</section>

<section class="main-section" id="Variables">
  <header>Variables</header>
  <p>You use variables as symbolic names for values in your application. The names of variables, called identifiers,
      conform to certain rules.

    A JavaScript identifier must start with a letter, underscore (_), or dollar sign ($); subsequent characters can
    also be digits (0-
    9). Because JavaScript is case sensitive, letters include the characters "A" through "Z" (uppercase) and the cha
    racters "a" through "z" (lowercase).


    You can use ISO 8859-
    1 or Unicode letters such as å and ü in identifiers. You can also use the Unicode escape sequences as characters
     in identifiers. Some examples of legal names are Number_hits, temp99, and _name.

  </p>
  <code></code>
</section>

<section class="main-section"id="Declaring_variables">
  <header>Declaring variables</header>
  <p>You can declare a variable in three ways:
    With the keyword var. For example,
    var x = 42.
    This syntax can be used to declare both local and global variables.


    By simply assigning it a value. For example,
    x = 42.
    This always declares a global variable. It generates a strict JavaScript warning. You shouldn't use this variant.
```

```
      With the keyword let. For example,
      let y = 13.
      This syntax can be used to declare a block scope local variable. See Variable scope below.

  </p>
  <code></code>
</section>

<section class="main-section"id="Variable_scope">
  <header>Variable scope</header>
  <p>When you declare a variable outside of any function, it is called a global variable, because it is available to
     any other code in the current document. When you declare a variable within a function, it is called a local var
     iable, because it is available only within that function.

     JavaScript before ECMAScript 2015 does not have block statement scope; rather, a variable declared within a bloc
     k is local to the function (or global scope) that the block resides within. For example the following code will
     log 5, because the scope of x is the function (or global context) within which x is declared, not the block, whi
     ch in this case is an if statement.

     if (true) { var x = 5; } console.log(x); // 5
     This behavior changes, when using the let declaration introduced in ECMAScript 2015.

     if (true) { let y = 5; } console.log(y); // ReferenceError: y is
     not defined

  </p>
  <code></code>
</section>

<section class="main-section" id="Global_variables">
  <header>Global variables</header>
  <p>Global variables are in fact properties of the global object. In web pages the global object is window, so you
     can set and access global variables using the window.variable syntax.
```

```html
        Consequently, you can access global variables declared in one window or frame from another window or frame by sp
        ecifying the window or frame name. For example, if a variable called phoneNumber is declared in a document, you
        can refer to this variable from an iframe as parent.phoneNumber.

      </p>
      <code></code>
  </section>

  <section class="main-section" id="Constants">
    <header>Constants</header>
    <p>You can create a read-
      only, named constant with the const keyword. The syntax of a constant identifier is the same as for a variable i
      dentifier: it must start with a letter, underscore or dollar sign and can contain alphabetic, numeric, or unders
      core characters.

      const PI = 3.14;
      A constant cannot change value through assignment or be re-
      declared while the script is running. It has to be initialized to a value.

      The scope rules for constants are the same as those for let block scope variables. If the const keyword is omitt
      ed, the identifier is assumed to represent a variable.

      You cannot declare a constant with the same name as a function or variable in the same scope. For example:

      // THIS WILL CAUSE AN ERROR function f() {}; const f = 5; // THIS
      WILL CAUSE AN ERROR ALSO function f() { const g = 5; var g;
      //statements }
      However, object attributes are not protected, so the following statement is executed without problems.
      const MY_OBJECT = {"key": "value"}; MY_OBJECT.key =
      "otherValue";

    </p>
    <code></code>
  </section>

  <section class="main-section" id="Data_types">
```

```html
<header>Data types</header>
<p>The latest ECMAScript standard defines seven data types:


<ul>
    <li>Six data types that are primitives:</li>
</ul>


  Boolean. true and false.
  null. A special keyword denoting a null value. Because JavaScript is case-
  sensitive, null is not the same as Null, NULL, or any other variant.
  undefined. A top-level property whose value is undefined.
  Number. 42 or 3.14159.
  String. "Howdy"
  Symbol (new in ECMAScript 2015). A data type whose instances are unique and immutable.
<ul>
    <li>and Object</li>
</ul>
  Although these data types are a relatively small amount, they enable you to perform useful functions with your a
  pplications. Objects and functions are the other fundamental elements in the language. You can think of objects
  as named containers for values, and functions as procedures that your application can perform.


</p>
<code></code>
</section>

<section class="main-section" id="if...else_statement">
  <header>if...else statement</header>
  <p>Use the if statement to execute a statement if a logical condition is true. Use the optional else clause to exe
    cute a statement if the condition is false. An if statement looks as follows:
    if (condition) { statement_1; } else { statement_2; }
    condition can be any expression that evaluates to true or false. See Boolean for an explanation of what evaluate
    s to true and false. If condition evaluates to true, statement_1 is executed; otherwise, statement_2 is executed.
     statement_1 and statement_2 can be any statement, including further nested if statements.
    You may also compound the statements using else if to have multiple conditions tested in sequence, as follows:


    if (condition_1) { statement_1; } else if (condition_2) {
```

statement_2; } else if (condition_n) { statement_n; } else {
            statement_last; }
            In the case of multiple conditions only the first logical condition which evaluates to true will be executed. To
             execute multiple statements, group them within a block statement ({ ... }) . In general, it's good practice to
            always use block statements, especially when nesting if statements:
            if (condition) { statement_1_runs_if_condition_is_true;
            statement_2_runs_if_condition_is_true; } else {
            statement_3_runs_if_condition_is_false;
            statement_4_runs_if_condition_is_false; }
            It is advisable to not use simple assignments in a conditional expression, because the assignment can be confuse
            d with equality when glancing over the code. For example, do not use the following code:
            if (x = y) { /* statements here */ }
            If you need to use an assignment in a conditional expression, a common practice is to put additional parentheses
             around the assignment. For example:
            if ((x = y)) { /* statements here */ }

    </p>
    <code></code>
</section>

<section class="main-section" id="While_statement">
    <header>While statement</header>
    <p>A while statement executes its statements as long as a specified condition evaluates to true. A while statement
        looks as follows:
        while (condition) statement
        If the condition becomes false, statement within the loop stops executing and control passes to the statement fo
        llowing the loop.
        The condition test occurs before statement in the loop is executed. If the condition returns true, statement is
        executed and the condition is tested again. If the condition returns false, execution stops and control is passe
        d to the statement following while.

        To execute multiple statements, use a block statement ({ ... }) to group those statements.

        Example:
        The following while loop iterates as long as n is less than three:

```
        var n = 0; var x = 0; while (n < 3) { n++; x += n; }
        With each iteration, the loop increments n and adds that value to x. Therefore, x and n take on the following va
        lues:


    <ul>
        <li>After the first pass: n = 1 and x = 1</li>
    </ul>


    <ul>
        <li>After the second pass: n = 2 and x = 3</li>
    </ul>


    <ul>
        <li>After the third pass: n = 3 and x = 6</li>
    </ul>


      After completing the third pass, the condition n < 3 is no longer true, so the loop terminates.


    </p>
    <code></code>
</section>

<section class="main-section" id="Function_declarations">
    <header>Function declarations</header>
    <p>A function definition (also called a function declaration, or function statement) consists of the function keyw
      ord, followed by:
    <ul>
        <li>The name of function</li>
    </ul>


    <ul>
        <li>A list of arguments to the function, enclosed in parentheses and separated by commas.</li>
    </ul>


    <ul>
```

```html
            <li>The JavaScript statements that define the function, enclosed in curly brackets, { }.</li>
        </ul>

        For example, the following code defines a simple function named square:

        function square(number) { return number * number; }
        The function square takes one argument, called number. The function consists of one statement that says to retur
        n the argument of the function (that is, number) multiplied by itself. The return statement specifies the value
        returned by the function.

        return number * number;
        Primitive parameters (such as a number) are passed to functions by value; the value is passed to the function, b
        ut if the function changes the value of the parameter, this change is not reflected globally or in the calling f
        unction.
    </p>
    <code></code>
    </section>

    <section class="main-section" id="Reference">
      <header>Reference</header>
        <ul>
            <li>All the documentation in this page is taken from <a href=""https://developer.mozilla.org/en-
            US/docs/Web/JavaScript/Guide">MDN</a></li>
        </ul>
    </p>
    <code></code>
    </section>
  </main>
</body>
</html>
```

STYLES.CSS (Página de Documentación 11)

```css
body {
    font-family: Verdana, Tahoma, sans-serif;
}

main {
    top: 20;
    margin-left: 315px;
}

#navbar {
    background-color: #D5F5E3;
    position: absolute;
    padding: 0;
    margin: 0;
    border: none;
    height: 100%;
    min-width: 300px;
    float: left;
    top: 0px;
    left: 0px;
}

header {
    color: black;
    background-color: #FAD7A0;;
    margin: 5px;
    text-align: left;
    font-size: 1.5em;
}

#main-doc {
    text-align: left;
}
```

```css
#navbar ul {
  height: 75%;
  padding: 0;
  overflow: auto;
  border: 1px solid;
}


#navbar li {
  color: #4d4e53;
  border-top: 1px solid;
  border-bottom: 1px solid;
  list-style: none;
  position: relative;
  width: 100%;
}


#navbar a {
  display: block;
  padding: 10px 10px; (Esto se escribe cuando quiero separar vaias palabras para que no esté, pegadas, unas conm otras)
  color: #4d4e53;
  text-decoration: none;
  cursor: pointer;
  border-top: 1px solid;
  border-bottom: 1px solid;
}


@ media (max-width:815px){
  #navbar {
    width: 30%;
    font-size: 14px;
    }
  }
/* For mobile phone:*/
```

(Esta es otra opción de query): @media (max-width: 768px) { #main-section { width: 358px;