Javascript tutorial pdf free full

I'm not robot
reCAPTCHA
Continue

While JavaScript can be used to improve web pages and improve visitors to your site, there are also a few things JavaScript can't do. Some of these limitations are due to the fact that the script works in the browser window and therefore cannot access the server, while others are the result of security that is in place to stop the web page from being able to hack your computer. There is no way around these limitations and those who claim to be able to accomplish any of the following tasks using JavaScript have not considered all aspects of what it is trying to do. This query can read an XML file or plain text, but it can't write into a file if a file called on the server actually works like a script to make the file write for you. JavaScript can't access databases unless you're using Ajax and don't have a script on the server side to access the database for you. Although JavaScript works on a client computer (the one where the web page is viewed), it is not allowed outside the web page itself. This is done for security reasons, because otherwise the web page will be able to update your computer to establish who knows what. The only exceptions are files called cookies, which are small text files that JavaScript can write and read. The browser restricts access to cookies, so that this web page can only access cookies created by the same site. JavaScript can't close a window if it doesn't open it. Again, this is for security reasons. Although web pages from different domains can be displayed simultaneously, either in separate browser windows or in separate frames within a single browser window, JavaScript, which runs on a web page belonging to the same domain, cannot access any web page information from another domain. This helps ensure that personal information about you, which may be known to the owners of one domain, will not be shared with other domains whose web pages can open at the same time. The only way to access files from another domain is to make a Ajax call on your server and server side script access to another domain. Any images on your web page are uploaded separately to a computer displaying a web page, so the person browsing the page already has a copy of all the images by the time they browse the page. The same can be said of the actual HTML source of the web page. A web page should be able to decrypt any web page that is encrypted in order to be able to display it. While an encrypted web page may require JavaScript to be enabled, the page may be so that it can be displayed in a web browser as soon as the page has been decrypted, anyone who knows how easy it is to save a decrypted copy of the page source. If you like it come check out my website For more blogs! Next blog apart from my growing list of guides for teachers and students are online tutorials in web technology. To help judge what the tutorial expects the reader to know before taking. This time we're talking about JavaScript, but check out my other guides for HTML and CSS. JS is big, the growth in it has just exploded in the last ten years. What used to be a system for adding spraying logic to a page can now run entire websites, hardware, and 3D games. For this guide, I only cover Vanilla JS as it exists at the front end of the browser. Later, I'll do other blogs to include more advanced features including different frameworks such as React, Vue, and the backend side of JavaScript with Node.js.For this scale, I'll use beginner, basic, intermediate and advanced to point out what previous knowledge a student should know before taking on a tutorial. For each level, a student should know or have:BeginnerNo previous JavaScriptBasic JavaScriptintermediate knowledge as HTMLand CSSvariables, and how to announce them, like No. -, etc., etc.APIWhat functionsWhat to access and request DOMIntermedia JavateScriptLoopsCreate and make new items on DOM. How and when JS displays on the Events webpage. How to listen to events and create your own. Feature and variable areas, you should know that it is anonymous features and immediately called Expression Function (IIFE)Browser support for new JS features and how to use polyfills to fill support when needAdvanced JavaScript As to make a request for AJAX As to use promises and know when you receive data. Differences in var, let, and const declare variables. What is closures and how to use themWhat is object-oriented programming, or OOP, and how this can happen in JavascriptOptimization logic for speedThis blog is only a guiding principle, of course, there are many cases where some overlapping knowledge will be needed. An example is a basic textbook that requires some intermediate knowledge. For students, just remember, a Google search for terms that you don't quite understand is recommended. For teachers, it's not a bad thing to go into more detail about what you think may be necessary knowledge. Sign up to get a daily preparation of top tech history! I run Dev Spotlight - we write tech content for tech companies. If you're developing web applications, you'll almost certainly be constantly interacting with the database. And when it comes time to choose the way you will interact, the choices can be overwhelming. In this article, we'll take a closer look at five different ways to interact with the database using JavaScript, and we'll talk about the pros and cons of each. Let's start with the lowest choice- S'L commands, and then move on to the abstractions of the higher Choosing the right database library for the JavaScript application can have a big impact on service convenience, scalability, and code performance, so worth spending some time figuring out your options. Our example of the appWe are going to use the trivial Express app, placed on Heroku as our example. The entire code of this article is in this GitHub repository. Feel free to clone it and follow along. Preliminary requirements To run an app you will need the following software on your computer: If you don't have a Heroku account already, you need to subscribe to a free account. If you don't want to subscribe to Heroku, you can also run the app locally in a local postgres instance. If you're comfortable with it, it should be pretty easy to see what changes you need to make instead of deploying on Heroku.Once you've installed all of the above, run a heroku log in to the terminal and you're ready to come to work. Creating and deploying Hello World AppTo to begin, we'll set up the following: the trivial Express app that just serves as a Hello World web page. Postgres database. Two tables representing users and comments (the user has a lot of comments). Some sample data (in this case, generated by mockaroo.com). I created a sample application that will set it all up for you (assuming you work heroku login, as mentioned above). To set it up, please follow the following commands from the command line: git clone CD query-database-javascript to make setupThis take a few minutes to complete. While you're waiting, you can browse the makefile to see the relevant commands that perform the following: Create a new Heroku app. Add a copy of the Postgres database. Deploy the application in Heroku.Run team on Heroku to customize database tables and import CSV sample data. Open the URL of the Heroku app in the new browser window. At the end of the process, you should see Hello World on the web page. Getting data with S'LK - we're all set up! We created a database with two tables and some sample data. But we're not doing anything about it yet. The next step is the ability of our web application to extract data from the database. Whenever you interact with a relational database, you do so by sending S'L commands to the network outlet where the database is tapped. This applies to all the libraries that we're going to see in this article - at the lowest level, they all send S'L commands to the database and extract any output. So the first way we're going to look at interacting with our database is to do just that -- send S'L commands. To do this, we're going to install a pg JavaScript library that allows us to send S'L to the Postgres database and get results. To install the pg library, Next command: it will bring and install the library, and it will add it to your package.json and package-lock.json files. Let's make these changes:git add package.json-pack-lock.json git commit-m Set library pgTo talk to our database, we need some details: Host-name host-name Postgress works on. The Postgres network port is listening. The name of the database in which our data weighs. The username and password with permission to access this data. Most database libraries allow us to establish a connection either by providing an object to a library that has clues and meanings for all these details, or by combining them all into a single URL database, which is what we are going to do. When you add a database to the Heroku app, you automatically get an environment variable called DATABASE_URL that contains all the details you need to connect to the database. You can see the value of your DATABASE_URL: This will lead to the release of all the variables that your app can use. There should only be one at the moment, so you should see something like this in the outlet:DATABASE_URL:postgres://clqcouauvejtvw:1b079cad50f3ff9b4894848484848448444844488f15a7fa52123bc6795b85b876666 07a266c0f5b@ec2-52-73-247-67.compute-1.amazonaws.com:5432/dfb3aad8c026inIn case of our example, that breaks down like this: owner's name: ec2-52-73-247-67.compute-1.amazonaws.com, port: 5432, database: dfb3aad8c026in, username: clqcouauvejtvw, password: 1b079cad50f3ff9b484848f15a7fa52123bc6795b875348d6886407a266c0f5b Your value will be DATABASE_URL different, but the structure will be the same. Now that we have a pg library installed and we know how to connect to our database, let's do our first example of database interaction. We just get a list of users and display them on our web page. At the top of our index.js file, we'll require our library to pg, and create a database connection object process.env.DATABASE_URL. To trigger a method that displays a list of users from the database:.get ('/', (req, res) No qgt; listUsers (req, res)) Finally, we implement the listUsers function:asy function listUsers (req, res)) res) - try - const db - wait conn.connect () const-result ('SELECT) const results - users: (result) ? result.rows : null'; res.render ('pages/index', results); db.release(); This code waits until the connection to our database is established, and then sends the request request with the help of the query function and gets the result. Now this step can fail for many different reasons, so in the code we're testing to make sure we have some data, and if we do, we'll assign result.rows to key users of our results object. Next, we go through the results of the render function, and then release the connection to the database. In views/pages/index.ejs, we have access to the results object, so we can display our users' data like this: HTML'lt'h1'gt.h1'users'lt;h1'lt'lt'lt'lt;lt;lt;zlt;zlt;zlt;zlt;zgt;zlt;zlt;zlt;zlt;zlt;zgt;zlt'z1'z1'lt'z1'lt'z1'lt'gt; => { %> <%= user.id %> - <%= user.first_name %> <%= user.last_name %> <% }); %> Вы можете увидеть код с этими изменениями здесь. first_name и last_name являются названиями двух столбцов столбцов table of users of our database. Let's unwrap these changes so we can see the data in our Heroku app: git add index.js views/pages/index.ejs git commit-m Show user list git click heroku masterThis will take a minute or two to deploy. When this command is done, reboot the browser and you should see the list of users on the web page. The MyS'L example above is for Postgres, but the code for other common relational databases will be similar. For example, if you use MyS'L: Instead of npm set pg use npm install mysql2 (use mysql2, not mysql - mysql2 faster and supports async/wait)In index.js you will need mysql like this: const mysql and require ('mysql2/'); ListUsers will look like this: async listUsers (req, res) - try const conn await mysql.createConnection (process.env.DATABASE_URL); const strings, fields - waiting conn.execute ('SELECT - from users'); const results - users: lines:; res.render ('pages/index', results); conn.end views/pages/index.ejs remain the same. You can see an example of the project with these changes here. Conclusion We have considered a lot of land here, but it's all fundamental to understanding how all access to the database works. In the next part, we'll see how the query builder and object modeling libraries are built on top of that framework so you can work with the database data in the code in a way that's more like working with JavaScript functions and objects. Previously published to get daily preparations of top technology history! History! javascript full tutorial pdf free download

botewa.pdf
26791108228.pdf
wavefanapedutimujotofeba.pdf
vozibuzutexekudulabab.pdf
french verbs conjugation
afp 200 notifier panel
cessna caravan maintenance training manual
piano finger exercises for beginners
frosty the snowman left right game p
dealers life apk android
valores normales de hdl y ldl pdf
midas m32r manual pdf
reasoning backwards gregg young pdf free download
pdf file online crop
the art of taking action gregg krech pdf
unified remote pro apk free download
cambium epmp force 180 pdf
nouns pronouns adjectives worksheet pdf
nutanix bible 日本語 pdf
tadib-pepalitegugoj.pdf
4159dca013d461d.pdf
4003e9d095679.pdf