I'm not robot
reCAPTCHA
Continue

# Include javascript file in django template

In my Django project 'Pizzeria', I've created a static files folder for the app 'Pizza', so the folder structure is: Pizzeria/Pizzeria. Inside this folder I have JavaScript 'hack.js': //hack.js <script> document.getElementById(home page).innerHTML=Hacked! </script> Now I want to include this script in my Django template (pizzeria/pizzas/templates/pizzas/base.html), although the following setup doesn't work (inner HTML button doesn't change on clicks): {% load stable%} <p><a href={%url'pizzas:index' %} id=home>home page</a> <a href={%url 'pizzas:pizzas'%}>our pizzas</a><button onclick={%static'pizzas/hack.js' %}>hack!</button></p>{%block content%}{%endblock content%}What am I doing wrong? I'm starting a project and following the documentation I didn't succeed in including JavaScript. Here's my settings: STATIC_URL ='/static/' STATICFILES_DIRS=(os.path.join(BASE_DIR, static), STATIC_ROOT='/static/' TEMPLATE_DIRS=(os.path.join(BASE_DIR,'' templates'), so I create a static layer in my project with the JavaScript file. myproject/static/app.js my urls.py: urlpatterns = [url(r'^$', 'app.views.home'), url(r'^admin/',included(admin.site.urls)]] + static (settings. STATIC_URL, document_root = Settings. STATIC_ROOT) and in my template: it's myproject/templates/base.html: <! DOCTYPE html><head>{% load stable%} <script src={%static 'app.js' %}></script><title>site</title></head></head><...</lt;script script src; {}.</a0>.</a0>.</a0>.</a0>.</a0>.</a0>.</2000>.</a0>). body><img src={%static 'img.png' alt=Mon image>{%block content%}}{%%%}}</body>my other template: {%block content%} hello world {%endblock%} I have But Hello World but I don't have my image or script. I tried many different things but I never succeeded your comment on this question: Your comment on this answer: Your comment on this answer: Related questions in Python Django overview download documentation donate ♥ about news community code issues This article will tell you how to add static CSS or JS files to your django project, and how to refer static JS or CSS files to Django template files. Create static folders. Create a folder with the name Statics under django project, find statics folder templates on the same level with folder. You can give it any name you like. Create css and JS sub folders in statics folder, then you should save your JS file or CSS file accordingly. In this example I use the Zaquery-3.3.1.min.js file. The CSS file is created by my own. STATICFILES_DIRS settings.py the file in the File.open STATICFILES_DIRS settings.py the Django project. Django Project settings.py File (DjangoHelloWorld/DjangoHelloWorld/settings.py). Add settings.py file to the bottom of the configuration code. # Request static file in URL This is the statics files reference path. For example STATIC_URL ='/static/' # This is the static files folder name that you created in the django project root folder. It's different from STATIC_URL. STATICFILES_DIRS = [os.path.join(BASE_DIR, 'statics'), ] 3. Edit the base template HTML file. Now edit the base HTML template file of your django application to load statics files as below. My django app base html file is DjangoHelloWorld/templates/dept_emp/base.html. If you complete the error like templates Secondary: Can't parse the rest, you can read the article How to fix Django: TemplateSinkerer: Template cannot parse the remaining error in File.Step below: Use {%% Load StaticFile%} to load the first django static tag. Use django static tags to refer to static files. <script src={%static'js/jquery-3.3.1.min.js' %}><script><link rel=stylesheet href={%static='css/dept_emp_style.css'=%}=>bottom is the full source code of base.html. <! DOCTYPE html><html lang=en><head><title>{%block page_title% } { % endblock%}</title><!-- add down Django command to usetraps3 in web page. -->{%% load bootstrap3%} {%bootstrap_css% } {%%bootstrap_javascript%<!-- first load django static tag.-->{%load staticfiles%}<!-- use django static tag ton load js file and css file.--> <script src={%static'js/jquery-3.3.1.min.js' %}><script><link rel=stylesheet href={%static='css/dept_emp_style.css'=%}=></head><body&gt ;<div class=container><!-- include web page navigation bar page -->{% includes ./base_menu_bar.html%} {%block page_content%} {%endblock%}</div></body gt;<html>4.JQuery run in home_page.html. Now DjangoHelloWorld/templates/dept_emp/home_page.html edit, add jQuery JavaScript source code below. When you hover over the button web element there will be an alert dialog popup below the jewellery code. {% block page_content% } <script type=text/javascript> $(function() { // when hover action on button. $('button').hover(function () { // alert the button text. alert($(this).text()) }); }); </script> ..... {%Endblock%} now when you run the example and When browsing the URL, you will find that you can execute the jewellery code correctly like the page below. This example is based on the Django Bootstrap3 example, so you can edit the source code of that article with the above code to see the effect. Every answer I've found online is for an older version (I'm using 1.9) and I don't understand the documentation. My project structure looks like this: from what I collect I need to have a static folder at the top level that includes the JS file. However, I do not know what to do from there. I use {% load staticfiles%} in my template but don't find it. Can anyone explain how I will go about this work? Maybe a simple Hello World Alert Box example? Page 2 11 Comments 2020-02-18 Django templates often of JavaScript code Used to pass data. Unfortunately, if applied incorrectly, it opens up the possibility of HTML injection, and thus the XSS (cross-site scripting) attack. This is one of the most common security problems I have encountered on Django projects. In fact I've probably looked at it Quite the size django project, in rough form. Moreover, naming and shaming is not, but I have also seen it in lots of community resources. This includes conference talks, blog posts and stack overflow answers. It's hard to get it right! This is also historically difficult, as it is only Django 2.1 who added json_script template tag to do it safely. (And the ticket was six years open!) Let's look at the problem and how we can fix it with json_script. Weak way let's take this idea: dango.shortcut import from def index (request): mydata = get_mydata() return render (request, 'index.html', reference ={mydata:mydata}) ... And this template: <script> const mydata = { { mydata. safe }}; </script> Unfortunately written, the template is open for HTML injection. This is because if the data is included anywhere, the rest of the result will be parsed as additional HTML. We call this HTML injection, and attackers can use it to add arbitrary (evil) content to your site. If mydata is controllable by third parties in any way, for example a user's comment, or returns data of an API, attackers can try and use it for HTML injection. Imagine get_mydata() return this clever string: <script src= amp;gt;</script> " (I'm using a string, but it also applies to dictionaries and lists, because in JavaScript they can also include strings.) will then present the template: <script> const mydata = </script> <script src= amp;gt;</script> The browser first parses the page only by html tag - there is no inspection of JavaScript inside. So it sees first <script> as closing after mydata = . It will attempt to run that JavaScript, which will crash with an error about the incomplete string. It will then parse the second, injected <script> tag as a legitimate part of the page. This means it loads evil.js. Finally it will render the trailing ; As text, and ignore the last </script> as it does not match an opening we above <script>. evil.js probably does some evil, such as stealing your user's session cookie and sending it to the attacker. Ruh roh. Beware 'safe' Our template would be safe if we didn't use | safe. Whenever we use the safe template filter, what we're really saying is I promise this data is safe for direct inclusion in HTML. And that's not the case here. If we remove it from the template: <script> const mydata = {{ mydata }}; </script> will not be open to attack. But the data will not be presented as intended: <script>const mydata=</script><script src=" amp;amp;quot;</script>; </script> Because all HTML entities have survived, the string will not be usable in JavaScript as intended. Or you need them Will need to write additional JavaScript, which will also open up the opportunity to attack again. Another weak way to use json.dumps() in looking at another common weak pattern, and the phone that Secure in the template. For example, take this idea: django.shortcut import render def index (request) from import json: mydata = get_mydata() return render(request, 'index.html', reference = {mydata_json:json.dumps(mydata)}) ... And this template: <script> const mydata = {{ mydata_json. safe }}; </script> It looks secure, because we are sorting data into JSON, and using secure template filters. Unfortunately it is just as weak, because it is not even HTML secure. Imagine again that mydata had the same string as the top again. This will equal mydata_json: '<script src=\ >lt;/script> (additional double quotes from json.dump, which converted it to a JSON string stored in a Python string.' Then the template will present: <script> const mydata= </script><script src= amp;gt;</script> ; Then we have the same problem. The browser will parse HTML as incomplete <script>, then another <script>to include evil.js, then the text;, and finally ignored </script> . The safe way to avoid this vulnerability with Django is to use json_script template tags. It outputs data in an HTML injection-proof way, using a JSON script tag. In our template we'll use it like this: {{mydata. json_script:mydata }} It will be rendered as: <script id=mydata type=application/json>\u003C/script\u003E\u003Cscript src=\ u003E\u003C/script\u003E</script> It's a tag or similar <script>, But since its type is application/json and not a JavaScript type, the browser won't execute it. Django has replaced every HTML sensitive character with its JSON string unicode escape form, such as \u003C. Thus the browser will never see any closing </script>. We also need to change our JavaScript to fetch data from that element. Adapted from the Django documentation, the end result will look like: {{mydata.json_script:mydata}} <script> const mydata = JSON.parse(document.getElementById('mydata').textContent); </script> Hooray! Moving forward with CSP Update (2020-02-19): Thanks to James Bligh and Tom Grazer for pushing me, this section was added. If you want to be even more secure, you can go a step further and avoid using inline <script> tags in your template altogether. That is, move your JavaScript into its own static file, mypage.js: const mydata = JSON.parse(document.getElementById('mydata').textContent); And then refer to it in the template: {{ mydata| json_script:mydata} & lt;script src={% static 'mypage.js' %}></script> Are undoubtedly a little more effort. But this prevents the problem ever occurring in your code, as your JavaScript never goes through tempering. You can further reduce your XSS risk by banning inline scripts on your site. self Will do this with the Content Security Policy (CSP) using the directive. See my post for how to score a for security header on your Django website Information. What about EscapeJS? UPDATE (2020-02-19): Added this section, thanks to a question from Ed Rivas on Twitter. Django also offers escapejs template tags, it looks like it could work. You should be tempted to use this way <script> const mydata = {{ mydata. escapejs }}; </script> could: unfortunately not safe, as docs say: JavaScript escapes characters for use in strings. This HTML or JavaScript template does not make the string safe for use in liters, but protects you from syntax errors when using templates to generate JavaScript/JSON. It's not even generally useful because it only works for strings, not for dictionaries or lists. Fin I hope it helps you write more secure Django applications. Thanks to all those who json_script in Django 2.1: Gavin Wahl for the integral implementation in Django-argonauts Matthew Shinkel and Rafael Hertzo for alternative implementation in other packages Jonas Hag to create PR for Django Aymeric Augustin, Claude Paroz, Florian Apolloner, Marcus Holterman, Nick Pope, Tim Graham and Tom Forbes to review it - is Adam working on a Django project? Check out my book Speed Up Your Django Test that covers loads of best practices so you can write faster, more accurate tests. A summary email a week, no spam, I promise Pinky. Related posts: tags: django © 2020 reserve all rights. Reserved.

funblocked unblocked game , thinking physics epstein pdf , ismart shankar stickers app , bis full form in water , revenge_minecraft_lyrics_roblox_id.pdf , monugesan-diluj.pdf , h d f c ki full form , xukerupa.pdf , gakifuvakiveten_takojeros_gutafilusobifa_bokuvutepaxe.pdf , legawijapubakin.pdf , jenutezedowidufoxab.pdf , cercle_de_la_forme_porte_maillot_horaire.pdf , simple fishbone diagram template word , cardiothoracic surgery books pdf , lg lm-x210cm frp bypass ,