


I'm not robot  reCAPTCHA

Continue

Html form checkbox required

Inbox object Find out if a check box should be checked before submitting a form: var x = document.getElementById (myCheck).required; The result of x will be: true Try it Yourself » Definition and Use The required property sets or returns if a check box must be checked before submitting a form. This property reflects the hml attribute required. Browser support property required Yes 10.0 Yes Yes Syntax Return required property: Set the required property: Check boxObject.required=true/false Property Values Description of true property value/false Specifies whether a check box should be checked before submitting a true form - The check box must be checked before submitting a false form - Default. The check box is not a necessary part of submitting the Return value of technical details form: a boolean, returns true if the check box must be checked before submitting a form, otherwise false returns More Examples Set a check box to be a required part of the form submission: document.getElementById(myCheck).required = true; Try it yourself » HTML reference related pages: HTML <input>; Required attribute input check box object <input>; type check box elements are rendered by default as boxes that are checked (checked) when activated, as you can see on an official government paper form. The exact appearance depends on the configuration of the operating system under which the browser is running. Usually this is a square, but may have rounded corners. A check box allows you to select unique values for submission to a form (or not). Note: Radio buttons are similar to check boxes, but with an important distinction— radio buttons are grouped into a set in which only one radio button can be selected at a time, while check boxes allow you to turn on and off unique values. When there are multiple controls, radio buttons allow one to be selected from all of them, while check boxes allow multiple values to be selected. Value A DOMString representing the value of the check box. This is never seen on the client side, but on the server this is the value given to the data sent with the name of the check box. Here's the following example: <form><div><input type=checkboxbox id=subscribeNews name=subscribe value=newsletter><label for=subscribeNews>Subscribe to newsletter?</label ></div><div><button type=submit>Sign up</button></div></form>; In this example, we have an inscription name, and a newsletter value. the form is submitted, the data name/value pair will be subscribed=newsletter. If the value attribute was omitted, the default value of the check box is turned on, so the data sent in this case would be subscribed=on. Note: If a check box is not checked when your form is submitted, there is no value sent to the server to represent its unverified state (for example, value=unverified); the value is not submitted to the server. If you want to send a one value for the check box when it is not checked, you can include a <input type=hidden> inside the form with the same name and value, generated by JavaScript perhaps. Additional attributes In addition to the common attributes shared by all <input> elements, check box entries support the following attributes: Description of the Boolean verified attribute; if present, the check box is switched by default A Boolean that, if present, indicates that the value of the check box is indeterminate instead of the true or false value The sequence to be used as the value of the check box when submitting the form, if the check box is currently switching into a checked boolean attribute indicating whether or not this check box is checked by default (when the page is loaded). It does not indicate whether this check box is currently checked: If the state of the check box changes, this content attribute does not reflect the change. (Only the verified IDL attribute of the HTMLInputElement is updated.) Note: Unlike other input controls, a check box value is only included in the sent data if the check box is currently checked. If it is, then the value of the value attribute of the check box is reported as the value of the entry. Unlike other browsers, Firefox by default persists the verified dynamic state of a <input> through page loads. Use the autocomplete attribute to control this feature. indeterminate If the indeterminate attribute is present in the <input> element by setting a check box, the value of the check box is neither true nor false, but is undetermined, which means that its state cannot be determined or declared in pure binary terms. This can happen, for example, if the state of the check box depends on several other check boxes, and these check boxes have different values. Essentially, then, the indeterminate attribute adds a third possible state to the check box: I don't know. In this state, the browser can draw the check box in gray or with a different tag within the check box. For example, browsers on macOS can draw the check box with a hyphen inside to indicate an unexpected state. Value The value attribute is one that everyone <input>s share; however, it serves a special purpose for type check box entries: when a form is submitted, only check boxes that are currently checked are submitted to the server, and the reported value is the value of the value attribute. If the value is not specified otherwise, it is the sequence by default. This is demonstrated in the Value section above. Using check box entries We have already covered the most basic use of the above check boxes. Let's go now for the other common features and techniques related to the check box you'll need. Handling multiple check boxes The example we saw above contained only one check box; in real-world situations, you are likely to find multiple check boxes. If they're not completely related, then you can just deal with all of them they as shown above. However, if they are all related, things are not so simple. For example, in the following demo, we've included several check boxes to allow the user to select their interests (see the full version in the Examples section). <fieldset><legend>Choose your interests</legend><div><input type=checkboxbox id=coding name=interest value=coding><label for=coding>Coding</label></div><div><input type=checkboxbox id=music name=interest value=music><label for=music>Music</label></div></fieldset>In this example, you'll see that we gave each check box the same name. If both check boxes are checked and then the form is submitted, you will receive a sequence of pairs of names/values submitted as follows: interest=coding&interest=music. When this string reaches the server, you need to analyze it beyond as an associative array, so that all values, not just the last value, of interest are captured. For a technique used with PHP, see Handle multiple check boxes with a single server variable, for example. Checking boxes by default To make a check box checked by default, just give it the verified attribute. See example below.<fieldset><legend>Choose your interests</legend><div><input type=checkboxbox id=coding name=interest value=checked=></label for=coding>Coding</label></div><div><input type=checkboxbox id=music name=interest value=music><label for=music>Music</label></div></fieldset>Providing a larger area of success for your check boxes In the examples above, you may have noticed that you can switch a check box by clicking on your <label> associated element, as well as in the check box itself. This is a really useful feature of HTML form labels that makes it easy to click on the option you want, especially on small screen devices like smartphones. In addition to accessibility, this is another good reason to properly configure <label> elements in your forms. Indeterminate state check boxes In addition to the verified and unverified states, there is a third state in which a check box may be: undetermined. This is a state in which it is impossible to tell if the item is toggled in or out. This is defined using the indeterminate property of the HTMLInputElement object via JavaScript (cannot be defined using an HTML attribute): inputInstance.indeterminate = true; A check box in the indeterminate state has a horizontal line in the box (it looks a bit like a hyphen sign or less) rather than a check/tick in most browsers. There are not many use cases for this property. most common is when a check box is available that has a number of sub-options (which are also check boxes). If all sub-options are checked, the owning check box is also checked, and if all are not checked, the own check box is not checked. If any or more of the sub-options have a different state than the others, the check box itself is in the indeterminate <label> </label> This can be seen in the example below (thanks to the CSS Tricks for inspiration). In this example, we track the ingredients we're collecting for a recipe. When you check or uncheck an ingredient's check box, a JavaScript function checks the total number of verified ingredients: If none is checked, the recipe name check box is set to cleared. If one or two are checked, the recipe name check box is set to undetermined. If all three are checked, the recipe name check box is set to verified. So in this case, the indeterminate state is used to state that the collection of the ingredients has already begun, but the recipe is not yet complete. var global = document.querySelector ("input[id=EnchTb]); var ingredients = document.querySelectorAll ("ul input"); overall.addEventListener('click', function(e) { e.preventDefault(); }); para(var i = 0; i < ingredients.length; i++)= {= ingredients[i].addEventListener('click', ' updatedisplay);= }= function= updatedisplay()= {= var= checkedcount=0; for(var= i=0; i=>< ingredients.length; i++)= {= if(ingredients[i].checked)= {= checkedcount++;= }= if(checkedcount=== 0)= {= overall.checked=false; = else= if(checkedcount=== ingredients.length)= {= overall.checked=true; overall.indeterminate=false; } = else= {= overall.checked=false; overall.indeterminate=true; } = == == if= if= you= submit= a= form= with= an= indeterminate= checkbox.= the= same= thing= happens= as= if= the= checkbox= were= unchecked= == no= data= is= submitted= to= represent= the= checkbox.= validation= checkboxes= do= = validation = validation (offered= to= all=><input>s). However, most States of Validity will always be false. If the check box has the required attribute but is not checked, then validityState.valueMissing is true. ExampleS The following example is an extended version of the example of several check boxes that we saw above—it has more standard options, plus another check box that, when checked, causes a text field to appear to enter a value for the other option. This is achieved with a simple block of JavaScript. The example also includes some CSS to improve the style. Html <form><fieldset><legend>Choose your interests</legend><div><input type=checkboxbox id=coding name=interest value=coding ><label for=coding>Coding</label></div><div><input type=checkboxbox id=music name=interest value=music> <label for=music>Music</label></div><div><input type=checkboxbox id=art name=interest value=art><label for=art>Art</label></div><div><input type=checkboxbox id=sports name=interest value=sports><label type=checkboxbox id=cooking name=interest value=cooking><label for=cooking>Cozinhando</label></div></div><div><input type=checkboxbox id=other name=interest value=other><label for=other>Outros</label></div><input type=text id=otherValue name=other></div></div></fieldset></form> name=other></div><div><div></div></fieldset></form> type=submit>Submit form CSS html { font-family: sans-serif; } form { width: 600px; margin: 0 auto; } div { bottom-margin: 10px; } fieldset { background: cyan; edge: 5px solid blue; } legend { padding: 10px; background: blue; color: cyan; } JavaScript var otherCheckbox = document.querySelector ("input[value=other]"); var otherText = document.querySelector ("input[id=otherValue]); otherText.style.visibility = 'hidden'; otherCheckbox.addEventListener('change', () => { if(otherCheckbox.checked) { otherText.style.visibility = 'visible'; otherText.value = ""; } else { otherText.style.visibility = 'hidden'; } }) Browser compatibility specifications Compatibility on GitHubDesktopChromeEdgeFirefoxInternet ExplorerExplorerSafariAndroid webviewChrome for AndroidFirefox for AndroidOpera for AndroidSafari on iOSSamsung Internettype=CheckboxChrome Support support for YesEdge Full support 1 Full SimIE Support Full SimIE Full Support SimOpera Full Support SimSafari Full Support SimWebView Full Android Support SimChrome Full Android Support SimFirefox Full Support 4Opera Full Android Support SimSafari iOS Full Support SimSamsung Internet Full Support Yes Full Support