Continue

The built-in browser component lets you render different kinds of form inputs. To display an input, render the built-in browser component. See more examples below.Props supports all common element props.formAction: A string or function. Overrides the parent for type="submit" and type="image". When a URL is passed to action the form will behave like a standard HTML form. When a function is passed to formAction the function will handle the form submission. See .You can make an input controlled by passing one of these props:checked: A boolean. For a checkbox input or a radio button, controls whether it is selected.value: A string. For a text input, controls its text. (For a radio button, specifies its form data.)When you pass either of them, you must also pass an onChange handler that updates the passed value.These props are only relevant for uncontrolled inputs: These props are relevant both for uncontrolled and controlled inputs:accept: A string. Specifies which filetypes are accepted by a type="file" input.alt: A string. Specifies the alternative image text for a type="image" input.capture: A string. Specifies the media (microphone, video, or camera) captured by a type="file" input.autoComplete: A string. Specifies one of the possible autocomplete behaviors.autoFocus: A boolean. If true, React will focus the element on mount.dirname: A string. Specifies the form field name for the elements directionality.disabled: A boolean. If true, the input will not be interactive and will appear dimmed.children: does not accept children.form: A string. Specifies the id of the this input belongs to. If omitted, its the closest parent form.formAction: A string. Overrides the parent for type="submit" and type="image".formEnctype: A string. Overrides the parent for type="submit" and type="image".formMethod: A string. Overrides the parent for type="submit" and type="image".formNoValidate: A string. Overrides the parent for type="submit" and type="image".formTarget: A string. Overrides the parent for type="submit" and type="image".height: A string. Specifies the image height for type="image".list: A string. Specifies the id of the with the autocomplete options.max: A number. Specifies the maximum value of numerical and datetime inputs.maxLength: A number. Specifies the maximum length of text and other inputs.min: A number. Specifies the minimum value of numerical and datetime inputs.minLength: A number. Specifies the minimum length of text and other inputs.multiple: A boolean. Specifies whether multiple values are allowed for ;}A controlled input makes sense if you needed state anywayfor example, to re-render your UI on every edit:function Form() { const [firstName, setFirstName] = useState(''); return (First name: setFirstName(e.target.value)} /> {firstName !== '' && Your name is {firstName}.} ...Its also useful if you want to offer multiple ways to adjust the input state (for example, by clicking a button):function Form() { const [age, setAge] = useState(''); const ageAsNumber = Number(age); return (Age: setAge(e.target.value)} type="number" /> setAge(ageAsNumber + 10)}> Add 10 years The value you pass to controlled components should not be undefined or null. If you need the initial value to be empty (such as with the firstName field below), initialize your state variable to an empty string ('').import { useState } from 'react'; export default function Form() { const [firstName, setFirstName] = useState(''); const [age, setAge] = useState('20'); const ageAsNumber = Number(age); return (First name: setFirstName(e.target.value)} /> Age: setAge(e.target.value)} type="number" /> setAge(ageAsNumber + 10)}> Add 10 years {firstName !== '' && Your name is {firstName}. } {ageAsNumber > 0 && Your age is {ageAsNumber}. });}If you pass value without onChange, it will be impossible to type into the input. When you control an input by passing some value to it, you force it to always have the value you passed. So if you pass a state variable as a value but forget to update that state variable synchronously during the onChange event handler, React will revert the input after every keystroke back to the value that you specified. When you use a controlled input, you set the state on every keystroke. If the component containing your state re-renders a large tree, this can get slow. Theres a few ways you can optimize re-rendering performance.For example, suppose you start with a form that re-renders all page content on every keystroke:function App() { const [firstName, setFirstName] = useState(''); return (setFirstName(e.target.value)} />);}Since doesnt rely on the input state, you can move the input state into its own component:function App() { return ( );}function SignupForm() { const [firstName, setFirstName] = useState(''); return (setFirstName(e.target.value)} />);}This significantly improves performance because now only SignupForm re-renders on every keystroke.If there is no way to avoid re-rendering (for example, if PageContent depends on the search inputs value), useDeferredValue lets you keep the controlled input responsive even in the middle of a large re-render. Troubleshooting My text input doesnt update when I type into it If you render an input with value but no onChange, you will see an error in the console:You provided a value prop to a form field without an onChange handler. This will render a read-only field. If the field should be mutable use defaultValue. Otherwise, set either onChange or readOnly.As the error message suggests, if you only wanted to specify the initial value, pass defaultValue instead:If you want to control this input with a state variable, specify an onChange handler: setSomething(e.target.value)} />If the value is intentionally read-only, add a readOnly prop to suppress the error: If you render a checkbox with checked but no onChange, you will see an error in the console:You provided a checked prop to a form field without an onChange handler. This will render a read-only field. If the field should be mutable use defaultChecked. Otherwise, set either onChange or readOnly.As the error message suggests, if you only wanted to specify the initial value, pass defaultChecked instead:If you want to control this checkbox with a state variable, specify an onChange handler: setSomething(e.target.checked)} />You need to read e.target.checked rather than e.target.value for checkboxes.If the checkbox is intentionally read-only, add a readOnly prop to suppress the error: If you control an input, you must update its state variable to the inputs value from the DOM during onChange.You cant update it to something other than e.target.value (or e.target.checked for checkboxes):function handleChange(e) { setFirstName(e.target.value.toUpperCase());}You also cant update it asynchronously:function handleChange(e) { setTimeout(() => { setFirstName(e.target.value); }, 100);}To fix your code, update it synchronously to e.target.value:function handleChange(e) { setFirstName(e.target.value);}If this doesnt fix the problem, its possible that the input gets removed and re-added from the DOM on every keystroke. This can happen if youre accidentally resetting state on every re-render, for example if the input or one of its parents always receives a different key attribute, or if you nest component function definitions (which is not supported and causes the inner component to always be considered a different tree). Im getting an error: A component is changing an uncontrolled input to be controlled If you provide a value to the component, it must remain a string throughout its lifetime.You cannot pass value={undefined} first and later pass value="some string" because React wont know whether you want the component to be uncontrolled or controlled. A controlled component should always receive a string value, not null or undefined.If your value is coming from an API or a state variable, it might be initialized to null or undefined. In that case, either set it to an empty string ('') initially, or pass value={someValue ?? ''} to ensure value is a string.Similarly, if you pass checked to a checkbox, ensure its always a boolean.Input types in HTML forms are the backbone of interactive web applications. They allow users to send information to web servers for various purposes like searching, logging in, or providing feedback. In this blog, we'll explore common HTML input types: text, password, radio, and checkbox.Text InputThe text input type is the most basic form of input and is widely used for collecting simple text data.In the above example, the placeholder attribute provides a hint to the user about what to enter.Password InputThe password input type is similar to the text type but hides the characters entered by the user for security reasons.Radio ButtonsRadio buttons are used when you want the user to select only one option from a set of choices.MaleFemaleCheckboxCheckboxes allow the user to select multiple options from a set.Subscribe to newsletterMore Input TypesHere is a comprehensive list of input types you can use in HTML:Input TypeDescriptiontextAllows the user to type a single line of text.passwordAllows the user to type a password.submitRepresents a button that, when pressed, submits the form.resetRepresents a button that, when pressed, resets all the form controls to their initial values.radioRepresents an option in a set of options that are mutually exclusive with each other.checkboxRepresents an option in a set that may be selected independently of other options.buttonRepresents a clickable button.colorAllows the user to select a color.dateAllows the user to select a date.datetime-localAllows the user to select a date and time with no time zone.emailAllows the user to enter an email address.fileAllows the user to select one or more files from their device storage.hiddenRepresents a value that is not displayed but is submitted to the server.imageDefines an image that acts as a submit button.monthAllows the user to select a month and year.numberAllows the user to enter a number.rangeAllows the user to select a number from a range.searchAllows the user to enter a search query string.telAllows the user to enter a telephone number.timeAllows the user to select a time.urlAllows the user to enter a URL.weekAllows the user to select a week.ConclusionUnderstanding the different types of HTML input is crucial for creating interactive and user-friendly forms. Each input type serves a specific purpose, making it easier to collect, validate, and process user data. The HTML tag defines the field where the user can enter data. The type attribute determines what type of user input is taken. Browser Output Here, type - determines the type of input the tag takes name - specifies the name of the input which is what the data is named as when submitting the data to a server. Different Input Types The various types of input tags available in HTML5 are: text - creates a single-line text fields (default) button - creates a button with no default functionality checkbox - creates a checkbox color - creates a color picker date - creates a date picker datetime-local - creates a date and time picker email - creates an input field that allows the user to input a valid email address file - creates an input field that lets the user upload a file or multiple files hidden - creates an invisible input field image - creates a button using an image month - creates an input field that lets the user enter month and year password - creates an input field that lets the user enter information securely radio - creates a radio button range - creates a range picker from which the user can select the value reset - creates the button which clears all the form values to their default value search - allows user to enter their search queries in the text fields submit - allows user to submit form to the server tel - defines the field to enter a telephone number time - creates an input field that accepts time value url - lets the user enter and edit a URL week - lets the user pick a week and a year from a calendar 1. Input Type text The input type text is used to create single-line text fields. It is the default input type. Search: Browser Output The input type text can also contain minlength, maxlength, and size attributes. For example, Name Browser Output In the above example, values are only allowed between the length of 4 to 8 characters. Note: If the type attribute is not provided, the tag becomes the type of text. 2. Input Type button The input type button is used to create a button with no default functionality. For example, Browser Output The text inside the value attribute is displayed in the button. Note: Generally, javascript is used to add functionality to such buttons. 3. Input Type checkbox The input type checkbox is used to create a checkbox input. For example, Subscribe to newsletter! Browser Output (checkbox unselected) The checkbox can be toggled between selected and not selected. Browser Output (checkbox selected) The value of the checkbox is included in the form data only if the checkbox is selected and is omitted if the checkbox is not selected. 4. Input Type color The input type color is used to create an input field that lets the user pick a color. For example, Background Color Browser Output (before expanding) Browser Output (after expanding) The color picker is inbuilt into the browser. Users can also manually enter the hex code of the color instead. The UI for the color picker differs from browser to browser. The input type date is used to create an input field that lets the user input a date using the date picker interface from the browser. For example, Birthday: Browser Output (before expanding) Browser Output (after expanding) The input type datetime-local is used to create an input field that lets the user pick a date and time using a date-time-picker from the browser. The time selected from the input does not include information about the timezone. For example, Choose a time for your appointment: Browser Output (before expanding) Browser Output (after expanding) 7. Input Type email The input type email is used to create an input field that allows the user to input a valid email address. Enter your email: Browser Output This input field throws an error if the value provided is not a valid email. For example, Browser Output 8. Input Type file The input type file is used to create an input field that lets the user upload a file or multiple files from their device. For example, Browser Output The input type hidden is used to create an invisible input field. This is used to supply additional value to the server from the form that cannot be seen or modified by the user. For example, 10. Input Type image The input type image is used to create a button using an image. Browser Output Let's see an example of how we can use it in a form. First name: Last name: Browser Output 11. Input Type month The input type month is used to create an input field that lets the user enter month and year using a visual interface from the browser. For example, Select Month: Browser Output (before expanding) Browser Output (after expanding) 12. Input Type password The input type password is used to create an input field that lets the user enter information securely. For example, Password: Browser Output The browser displays all the characters the user types using an asterisk (*). 13. Input Type radio The input type radio is used to define a radio button. Radio buttons are defined in a group. Radio buttons let users pick one option out of a group of options. cat dog Browser Output From the above example, we can see that all the radio buttons share the same name attribute. It allows the user to select exactly one option from the group of radio buttons. When submitting the form data, the key for the input will be the name attribute, and the value will be the radio button selected. Note: The name attribute is used as the key for the data when submitting the form. 14. Input Type range The input type range is used to create a range picker from which the user can select the value. User can select a value from the range given. It has a default range from 0 to 100. For example, Select value: Browser Output 15. Input Type reset The input type reset defines the button which clears all the form values to their default value. For example, Name: Browser Output Browser Output (after reset) 16. Input Type search The input type search allows user to enter their search queries in the text fields. It is similar to input type text. For example, Search: Browser Output Note: The search input does not work as a search unless we use some JavaScript to do the search calculation. The input type submit is used when the user submits the form to the server. It is rendered as a button. For example, Browser Output Here, The text provided in the value attribute of the input is shown in the button. 18. Input Type tel The input type tel is used to define the field to enter a telephone number. The telephone number is not automatically validated to a particular format as telephone formats differ across the world. It has an attribute named pattern used to validate the entered value. For example, Phone Number: Browser Output The pattern in the above example allows numbers in the format XXX-XX-XXX where X is a number between 0 and 9. 19. Input Type time The input type time attribute creates an input field that accepts time value. It allows users to add time in hours, minutes, and seconds easily. For example, Select Time: Browser Output (before expanding) Browser Output (after expanding) 20. Input Type url The input type url is used to let the user enter and edit a URL. For example, URL: Browser Output Here, placeholder is a hint that specifies the expected value of an input field, and the pattern defines what type of value is accepted. The above pattern means that only text beginning with https:// will be valid. 21. Input Type week The input type week lets the user pick a week and a year from a calendar. For example, Week Browser Output The INPUT element represents a field whose contents may beedited by the user. It has the following attributes. NAME Symbolic name used when transferring the form's contents.This attribute is always needed and should uniquely identifythis field. TYPE Defines the type of data the field accepts.Defaults to free text. SIZE Specifies the size or precision of the field according toits type. MAXLENGTH The maximum number of characters that will be accepted asinput. This can be greater that specified by SIZE ,in which case the field will scroll appropriately.The default is unlimited. VALUE The initial value for the field, or the value when checkedfor checkboxes and radio buttons. This attribute is requiredfor radio buttons. SRC A URL or URN specifying an image - for use only with TYPE=IMAGEMAP. ALIGN Vertical alignment of the image - for use only with TYPE=IMAGEMAP. Propsed CHECKED When present indicates that a checkbox or radio button isselected. DISABLED When present indicates that this field is temporarily disabled.Browsers should show this by "greying it" out in some manner. ERROR When present indicates that the field's initial value isin error in some way, e.g. because it is inconsistent with thevalues of other fields. Servers should include an explanatoryerror message with the form's text. Types The following types of fields can be defined with the TYPEattribute : TEXT Single line text entry fields. Use the SIZE attribute tospecify the visible width in characters,e.g. SIZE="24" for a 24 character field.The MAX attribute can be used to specify an upper limit to thenumber of characters that can be entered into a text field,e.g. MAX=72 . Use the TEXTAREA element for text fields whichcan accept multiple lines (see below). HIDDEN No field is presented to the user, but the content of thefield is sent with the submitted form. This value may be usedto transmit state information about client/server interaction. CHECKBOX Used for simple Boolean attributes, or for attributes whichcan take multiple values at the same time.The latter is represented by a number of checkbox fields eachof which has the same NAME . RADIO For attributes which can take a single value from a set ofalternatives. Each radio button field in the group should begiven the same NAME . SUBMIT This is a button that when pressed submits the form.It offers authors control over the location of this button.You can use an image as a submit button by specifying a URL withthe SRC attribute. RESET This is a button that when pressed resets the form's fieldsto their initial values as specified by the VALUE attribute.You can use an image as a reeset button by specifying a URL withthe SRC attribute. Proposed types RANGE This allows you to specify an integer range with the MINand MAX attributes, e.g. MIN=1 MAX=100 .Users can select any value in this range. INT For entering integer numbers, the maximum number of digitscan be specified with the SIZE attribute (excluding the signcharacter), e.g. size=3 for a three digit number. FLOAT For fields which can accept floating point numbers. SCRIBBLE A field upon which you can write with a pen or mouse.The size of the field in millimeters is given as SIZE= width, height. The units are absolute as they relate to the dimensionsof the human hand, rather than pixels of varying resolution.The scribble may involve time and pressure data in addition tothe basic ink data. You can use scribble for signatures or sketches.The field can be initialised by setting the SRC attribute toa URL which contains the ink *2 . The VALUE attribute is ignored. AUDIO This provides a way of entering spoken messages into a form.Browsers might show an icon which when clicked pops-up a setof tape controls that you can use to record and replay messages.The initial message can be set by specifying a URL with the SRCattribute. The VALUE attribute is ignored. Obsolete types DATE Fields which can accept a recognized date format. URL For fields which expect document references as URLs or URNs. IMAGE This allows you to specify an image field upon which youcan click with a pointing device. The SRC and ALIGN attributesare exactly the same as for the IMG and IMAGE elements.The symbolic names for the x and y coordinates of the click eventare specified with .x and .y for the given with the NAME attribute.The VALUE attribute is ignored. When you need to let users enter more than one line of text,you should use the TEXTAREA element. The tag is used within the element and defines fields for user input. The type of the field (text, checkbox, radio button, password field, etc.) is determined by the value of the type attribute. The tag doesnt have a text content, it contains only attributes. It belongs to a tag group called form elements. To associate text with a specific element, we use the tag which sets a text label for it. The tag is empty, which means that the closing tag isnt required. But in XHTML, the () tag must be closed (). Example of the HTML tag: Title of the document Example of the HTML tag with the HTML tag: Title of the document Your Name: Your Surname: Enter Your E-Mail: The main attribute that determines the type of input field is type. If the attribute is missing, the default value is "text". The tag also supports the Global attributes and the Event Attributes. How to style tag? Common properties to alter the visual weight/emphasis/size of text in tag: CSS font-style property sets the style of the font. normal | italic | oblique | initial | inherit. CSS font-family property specifies a prioritized list of one or more font family names and/or generic family names for the selected element. CSS font-size property sets the size of the font. CSS font-weight property defines whether the font should be bold or thick. CSS text-transform property controls text case and capitalization. CSS text-decoration property specifies the decoration added to text, and is a shorthand property for text-decoration-line, text-decoration-color, text-decoration-style. Coloring text in tag: CSS color property describes the color of the text content and text decorations. CSS background-color property sets the background color of an element. Text layout styles for tag: CSS text-indent property specifies the indentation of the first line in a text block. CSS text-overflow property specifies how overflowed content that is not displayed should be signalled to the user. CSS white-space property specifies how white-space inside an element is handled. CSS word-break property specifies where the lines should be broken. Other properties worth looking at for tag: Baseline Widely available *The HTML element is used to create interactive controls for web-based forms in order to accept data from the user; a wide variety of types of input data and control widgets are available, depending on the device and user agent. The element is one of the most powerful and complex in all of HTML due to the sheer number of combinations of input types and attributes.Name (4 to 8 characters):label { display: block; font: 1rem "Fira Sans", sans-serif;}input,label { margin: 0.4rem 0;}How an works varies considerably depending on the value of its type attribute, hence the different types are covered in their own separate reference pages. If this attribute is not specified, the default type adopted is text.The available types are as follows: Type Description Basic Examples button A push button with no default behavior displaying the value of the value attribute, empty by default. checkbox A check box allowing single values to be selected/deselected. color A control for specifying a color; opening a color picker when active in supporting browsers. date A control for entering a date (year, month, and day, with no time). Opens a date picker or numeric wheels for year, month, day when active in supporting browsers. datetime-local A control for entering a date and time, with no time zone. Opens a date picker or numeric wheels for date- and time-components when active in supporting browsers. email A field for editing an email address. Looks like a text input, but has validation parameters and relevant keyboard in supporting browsers and devices with dynamic keyboards. file A control that lets the user select a file. Use the accept attribute to define the types of files that the control can select. hidden A control that is not displayed but whose value is submitted to the server. There is an example in the next column, but it's hidden! image A graphical submit button. Displays an image defined by the src attribute. The alt attribute displays if the image src is missing. month A control for entering a month and year, with no time zone. number A control for entering a number. Displays a spinner and adds default validation. Displays a numeric keypad in some devices with dynamic keypads. password A single-line text field whose value is obscured. Will alert user if site is not secure. radio A radio button, allowing a single value to be selected out of multiple choices with the same name value. range A control for entering a number whose exact value is not important. Displays as a range widget defaulting to the middle value. Used in conjunction min and max to define the range of acceptable values. reset A button that resets the contents of the form to default values. Not recommended. search A single-line text field for entering search strings. Line-breaks are automatically removed from the input value. May include a delete icon in supporting browsers that can be used to clear the field. Displays a search icon instead of enter key on some devices with dynamic keypads. submit A button that submits the form. tel A control for entering a telephone number. Displays a telephone keypad in some devices with dynamic keypads. text The default value. A single-line text field. Line-breaks are automatically removed from the input value. time A control for entering a time value with no time zone. url A field for entering a URL. Looks like a text input, but has validation parameters and relevant keyboard in supporting browsers and devices with dynamic keyboards. week A control for entering a date consisting of a week-year number and a week number with no time zone. Obsolete values datetime A control for entering a date and time (hour, minute, second, and fraction of a second) based on UTC time zone. The element is so powerful because of its attributes; the type attribute, described with examples above, being the most important. Since every element, regardless of type, is based on the HTMLInputElement interface, they technically share the exact same set of attributes. However, in reality, most attributes have an effect on only a specific subset of input types. In addition, the way some attributes impact an input depends on the input type, impacting different input types in different ways.This section provides a table listing all the attributes with a brief description. This table is followed by a list describing each attribute in greater detail, along with which input types they are associated with. Those that are common to most or all input types are defined in greater detail below. Attributes that are unique to particular input typesor attributes which are common to all input types but have special behaviors when used on a given input typeare instead documented on those types' pages.Attributes for the element include the global HTML attributes and additionally:AttributeType(s)DescriptionacceptfileHint for expected file type in file upload controlsaltimagealt attribute for the image type. Required for accessibilityautocapitalizeall except url, email, and passwordControls automatic capitalization in inputted text.autocompleteall except checkbox, radio, and buttonsHint for form autofill featurecapturefileMedia capture input method in file upload controlscheckedcheckbox, radioWhether the command or control is checkeddirnamehidden, text, search, url, tel, emailName of form field to use for sending the element's directionality in form submissiondisabledallWhether the form control is disabledformallAssociates the control with a form elementformactionimage, submitURL to use for form submissionformenctypeimage, submitForm data set encoding type to use for form submissionformmethodimage, submitHTTP method to use for form submissionformnovalidateimage, submitBypass form control validation for form submissionformtargetimage, submitBrowsing context for form submissionheightimageSame as height attribute for ; vertical dimensionlistall except hidden, password, checkbox, radio, and buttonsValue of the id attribute of the of autocomplete optionsmaxdate, month, week, time, datetime-local, number, rangeMaximum valuemaxlengthtext, search, url, tel, email, passwordMaximum length (number of characters) of valuemindate, month, week, time, datetime-local, number, rangeMinimum valueminlengthtext, search, url, tel, email, passwordMinimum length (number of characters) of valuemultipleemail, fileBoolean. Whether to allow multiple valuesnameallName of the form control. Submitted with the form as part of a name/value pairpatterntext, search, url, tel, email, passwordPattern the value must match to be validplaceholdertext, search, url, tel, email, password, numberText that appears in the form control when it has no value setpopovertargetbuttonDesignates an as a control for a popover elementpopovertargetactionbuttonSpecifies the action that a popover control should performreadonlyall except hidden, range, color, checkbox, radio, and buttonsBoolean. The value is not editablerequiredall except hidden, range, color, and buttonsBoolean. A value is required or must be checked for the form to be submittablesizetext, search, url, tel, email, passwordSize of the controlsrcimageSame as src attribute for ; address of image resourcestepdate, month, week, time, datetime-local, number, rangeIncremental values that are validtypeallType of form controlvalueall except imageThe value of the control. When specified in the HTML, corresponds to the initial valuewidthimageSame as width attribute for A few additional non-standard attributes are listed following the descriptions of the standard attributes.acceptValid for the file input type only, the accept attribute defines which file types are selectable in a file upload control. See the file input type.altValid for the image button only, the alt attribute provides alternative text for the image, displaying the value of the attribute if the image src is missing or otherwise fails to load. See the image input type.autocapitalizeControls whether inputted text is automatically capitalized and, if so, in what manner. See the autocapitalize global attribute page for more information.autocomplete(Not a Boolean attribute!) The autocomplete attribute takes as its value a space-separated string that describes what, if any, type of autocomplete functionality the input should provide. A typical implementation of autocomplete recalls previous values entered in the same input field, but more complex forms of autocomplete can exist. For instance, a browser could integrate with a device's contacts list to autocomplete email addresses in an email input field. See autocomplete for permitted values.The autocomplete attribute is valid on hidden, text, search, url, tel, email, date, month, week, time, datetime-local, number, range, color, and password. This attribute has no effect on input types that do not return numeric or text data, being valid for all input types except checkbox, radio, file, or any of the button types.See the autocomplete attribute for additional information, including information on password security and how autocomplete is slightly different for hidden than for other input types.autofocusA Boolean attribute which, if present, indicates that the input should automatically have focus when the page has finished loading (or when the containing the element has been displayed).Note:An element with the autofocus attribute may gain focus before the DOMContentLoaded event is fired.No more than one element in the document may have the autofocus attribute. If put on more than one element, the first one with the attribute receives focus.The autofocus attribute cannot be used on inputs of type hidden, since hidden inputs cannot be focused.Warning:Automatically focusing a form control can confuse visually-impaired people using screen-reading technology and people with cognitive impairments. When autofocus is assigned, screen-readers "teleport" their user to the form control without warning them beforehand.Use careful consideration for accessibility when applying the autofocus attribute. Automatically focusing on a control can cause the page to scroll on load. The focus can also cause dynamic keyboards to display on some touch devices. While a screen reader will announce the label of the form control receiving focus, the screen reader will not announce anything before the label, and the sighted user on a small device will equally miss the context created by the preceding content.captureIntroduced in the HTML Media Capture specification and valid for the file input type only, the capture attribute defines which mediamicrophone, video, or camerashould be used to capture a new file for upload with file upload control in supporting scenarios. See the file input type.checkedValid for both radio and checkbox types, checked is a Boolean attribute. If present on a radio type, it indicates that the radio button is the currently selected one in the group of same-named radio buttons. If present on a checkbox type, it indicates that the checkbox is checked by default (when the page loads). It does not indicate whether this checkbox is currently checked: if the checkbox's state is changed, this content attribute does not reflect the change. (Only the HTMLInputElement's checked IDL attribute is updated.)Note:Unlike other input controls, a checkboxes and radio buttons value are only included in the submitted data if they are currently checked. If they are, the name and the value(s) of the checked controls are submitted.For example, if a checkbox whose name is fruit has a value of cherry, and the checkbox is checked, the form data submitted will include fruit=cherry. If the checkbox isn't active, it isn't listed in the form data at all. The default value for checkboxes and radio buttons is on.dirnameValid for hidden, text, search, url, tel, and email input types, the dirname attribute enables the submission of the directionality of the element. When included, the form control will submit with two name/value pairs: the first being the name and value, and the second being the value of the dirname attribute as the name, with a value of ltr or rtl as set by the browser. Fruit: When the form above is submitted, the input cause both the name / value pair of fruit=cherry and the dirname / direction pair of fruit-dir=ltr to be sent.For more information, see the dirname attribute.disabledA Boolean attribute which, if present, indicates that the user should not be able to interact with the input. Disabled inputs are typically rendered with a dimmer color or using some other form of indication that the field is not available for use.Specifically, disabled inputs do not receive the click event, and disabled inputs are not submitted with the form. formA string specifying the element with which the input is associated (that is, its form owner). This string's value, if present, must match the id of a element in the same document. If this attribute isn't specified, the element is associated with the nearest containing form, if any.The form attribute lets you place an input anywhere in the document but have it included with a form elsewhere in the document.Note:An input can only be associated with one form.formactionValid for the image and submit input types only. See the submit input type for more information.formenctypeValid for the image and submit input types only. See the submit input type for more information.formmethodValid for the image and submit input types only. See the submit input type for more information.formnovalidateValid for the image and submit input types only. See the submit input type for more information.formtargetValid for the image and submit input types only. See the submit input type for more information.heightValid for the image input button only, the height is the height of the image file to display to represent the graphical submit button. See the image input type.idGlobal attribute valid for all elements, including all the input types, it defines a unique identifier (ID) which must be unique in the whole document. Its purpose is to identify the element when linking. The value is used as the value of the 's for attribute to link the label with the form control. See .inputmodeGlobal value valid for all elements, it provides a hint to browsers as to the type of virtual keyboard configuration to use when editing this element or its contents. Values include none, text, tel, url, email, numeric, decimal, and search.listThe value given to the list attribute should be the id of a element located in the same document. The provides a list of predefined values to suggest to the user for this input. Any values in the list that are not compatible with the type are not included in the suggested options. The values provided are suggestions, not requirements: users can select from this predefined list or provide a different value.It is valid on text, search, url, tel, email, date, month, week, time, datetime-local, number, range, and color.Per the specifications, the list attribute is not supported by the hidden, password, checkbox, radio, file, or any of the button types.Depending on the browser, the user may see a custom color palette suggested, tic marks along a range, or even an input that opens like a but allows for non-listed values. Check out the browser compatibility table for the other input types.See the element.maxValid for date, month, week, time, datetime-local, number, and range, it defines the greatest value in the range of permitted values. If the value entered into the element exceeds this, the element fails constraint validation. If the value of the max attribute isn't a number, then the element has no maximum value.There is a special case: if the data type is periodic (such as for dates or times), the value of max may be lower than the value of min, which indicates that the range may wrap around; for example, this allows you to specify a time range from 10 PM to 4 AM.maxlengthValid for text, search, url, tel, email, and password, it defines the maximum string length (measured in UTF-16 code units) that the user can enter into the field. This must be an integer value of 0 or higher. If no maxlength is specified, or an invalid value is specified, the field has no maximum length. This value must also be greater than or equal to the value of minlength.The input will fail constraint validation if the length of the text entered into the field is greater than maxlength UTF-16 code units long. By default, browsers prevent users from entering more characters than allowed by the maxlength attribute. Constraint validation is only applied when the value is changed by the user. See Client-side validation for more information.minValid for date, month, week, time, datetime-local, number, and range, it defines the most negative value in the range of permitted values. If the value entered into the element is less than this, the element fails constraint validation. If the value of the min attribute isn't a number, then the element has no minimum value.This value must be less than or equal to the value of the max attribute. If the min attribute is present but is not specified or is invalid, no min value is applied. If the min attribute is valid and a non-empty value is less than the minimum allowed by the min attribute, constraint validation will prevent form submission. See Client-side validation for more information.There is a special case: if the data type is periodic (such as for dates or times), the value of max may be lower than the value of min, which indicates that the range may wrap around; for example, this allows you to specify a time range from 10 PM to 4 AM.minlengthValid for text, search, url, tel, email, and password, it defines the minimum string length (measured in UTF-16 code units) that the user can enter into the entry field. This must be a non-negative integer value smaller than or equal to the value specified by maxlength. If no minlength is specified, or an invalid value is specified, the input has no minimum length.The input will fail constraint validation if the length of the text entered into the field is fewer than minlength UTF-16 code units long, preventing form submission. Constraint validation is only applied when the value is changed by the user. See Client-side validation for more information.multipleThe Boolean multiple attribute, if set, means the user can enter comma separated email addresses in the email widget or can choose more than one file with the file input. See the email and file input type.nameA string specifying a name for the input control. This name is submitted along with the control's value when the form data is submitted.Consider the name a required attribute (even though it's not). If an input has no name specified, or name is empty, the input's value is not submitted with the form! (Disabled controls, unchecked radio buttons, unchecked checkboxes, and reset buttons are also not sent.)There are two special cases:_charset_ : If used as the name of an element of type hidden, the input's value is automatically set by the user agent to the character encoding being used to submit the form.isindex: For historical reasons, the name isindex is not allowed.The name attribute creates a unique behavior for radio buttons.Only one radio button in a same-named group of radio buttons can be checked at a time. Selecting any radio button in that group automatically deselects any currently-selected radio button in the same group. The value of that one checked radio button is sent along with the name if the form is submitted,When tabbing into a series of same-named group of radio buttons, if one is checked, that one will receive focus. If they aren't grouped together in source order, if one of the group is checked, tabbing into the group starts when the first one in the group is encountered, skipping all those that aren't checked. In other words, if one is checked, tabbing skips the unchecked radio buttons in the group. If none are checked, the radio button group receives focus when the first button in the same name group is reached.Once one of the radio buttons in a group has focus, using the arrow keys will navigate through all the radio buttons of the same name, even if the radio buttons are not grouped together in the source order.When an input element is given a name, that name becomes a property of the owning form element's HTMLFormElement.elements property. If you have an input whose name is set to guest and another whose name is hat-size, the following code can be used:let form = document.querySelector("form");let guestName = form.elements.guest;let hatSize = form.elements["hat-size"];When this code has run, guestName will be the HTMLInputElement for the guest field, and hatSize the object for the hat-size field.Warning:Avoid giving form elements a name that corresponds to a built-in property of the form, since you would then override the predefined property or method with this reference to the corresponding input.patternValid for text, search, url, tel, email, and password, the pattern attribute is used to compile a regular expression that the input's value must match in order for the value to pass constraint validation. It must be a valid JavaScript regular expression, as used by the RegExp type, and as documented in our guide on regular expressions. No forward slashes should be specified around the pattern text. When compiling the regular expression:the pattern will be implicitly wrapped with ^(?: and)$, such that the match is required against the entire input value, i.e., ^(?:)$.the 'v' flag is specified so that the pattern is treated as a sequence of Unicode code points, instead of as ASCII.If the pattern attribute is present but is not specified or is invalid, no regular expression is applied and this attribute is ignored completely. If the pattern attribute is valid and a non-empty value does not match the pattern, constraint validation will prevent form submission. If the multiple is present, the compiled regular expression is matched against each comma separated value.Note:If using the pattern attribute, inform the user about the expected format by including explanatory text nearby. You can also include a title attribute to explain what the requirements are to match the pattern; most browsers will display this title as a tooltip. The visible explanation is required for accessibility. The tooltip is an enhancement.See Client-side validation for more information.placeholderValid for text, search, url, tel, email, password, and number, the placeholder attribute provides a brief hint to the user as to what kind of information is expected in the field. It should be a word or short phrase that provides a hint as to the expected type of data, rather than an explanation or prompt. The text must not include carriage returns or line feeds. So for example if a field is expected to capture a user's first name, and its label is "First Name", a suitable placeholder might be "e.g., Mustafa".Note:The placeholder attribute is not as semantically useful as other ways to explain your form, and can cause unexpected technical issues with your content. See Labels for more information.popovertargetTurns an element into a popover control button; takes the ID of the popover element to control as its value. See the Popover API landing page for more details. Establishing a relationship between a popover and its invoker button using the popovertarget attribute has two additional useful effects:The browser creates an implicit aria-details and aria-expanded relationship between popover and invoker, and places the popover in a logical position in the keyboard focus navigation order when shown. This makes the popover more accessible to keyboard and assistive technology (AT) users (see also Popover accessibility features).The browser creates an implicit anchor reference between the two, making it very convenient to position popovers relative to their controls using CSS anchor positioning. See Popover anchor positioning for more details.popovertargetactionSpecifies the action to be performed on a popover element being controlled by a control . Possible values are:"hide"The button will hide a shown popover. If you try to hide an already hidden popover, no action will be taken."show"The button will show a hidden popover. If you try to show an already showing popover, no action will be taken."toggle"The button will toggle a popover between showing and hidden. If the popover is hidden, it will be shown; if the popover is showing, it will be hidden. If popovertargetaction is omitted, "toggle" is the default action that will be performed by the control button.readonlyA Boolean attribute which, if present, indicates that the user should not be able to edit the value of the input. The readonly attribute is supported by the text, search, url, tel, email, date, month, week, time, datetime-local, number, and password input types.See the HTML attribute: readonly for more information.requiredrequired is a Boolean attribute which, if present, indicates that the user must specify a value for the input before the owning form can be submitted. The required attribute is supported by text, search, url, tel, email, date, month, week, time, datetime-local, number, password, checkbox, radio, and file inputs.See Client-side validation and the HTML attribute: required for more information.sizeValid for email, password, tel, url, and text, the size attribute specifies how much of the input is shown. Basically creates same result as setting CSS width property with a few specialties. The actual unit of the value depends on the input type. For password and text, it is a number of characters (or em units) with a default value of 20, and for others, it is pixels (or px units). CSS width takes precedence over the size attribute.srcValid for the image input button only, the src is string specifying the URL of the image file to display to represent the graphical submit button. See the image input type.stepValid for date, month, week, time, datetime-local, number, and range, the step attribute is a number that specifies the granularity that the value must adhere to.If not explicitly included:step defaults to 1 for number and range.Each date/time input type has a default step value appropriate for the type; see the individual input pages: date, datetime-local, month, time, and week.The value must be a positive numberinteger or floator the special value any, which means no stepping is implied, and any value is allowed (barring other constraints, such as min and max).If any is not explicitly set, valid values for the number, date/time input types, and range input types are equal to the basis for stepping the min value and increments of the step value, up to the max value, if specified.For example, if you have , then any even integer, 10 or greater, is valid. If omitted, , any integer is valid, but floats (like 4.2) are not valid, because step defaults to 1. For 4.2 to be valid, step would have had to be set to any, 0.1, 0.2, or the min value would have had to be a number ending in .2, such as .Note:When the data entered by the user doesn't adhere to the stepping configuration, the value is considered invalid in constraint validation and will match the :invalid pseudoclass.See Client-side validation for more information.tabindexGlobal attribute valid for all elements, including all the input types, an integer attribute indicating if the element can take input focus (is focusable), if it should participate to sequential keyboard navigation. As all input types except for input of type hidden are focusable, this attribute should not be used on form controls, because doing so would require the management of the focus order for all elements within the document with the risk of harming usability and accessibility if done incorrectly.titleGlobal attribute valid for all elements, including all input types, containing a text representing advisory information related to the element it belongs to. Such information can typically, but not necessarily, be presented to the user as a tooltip. The title should NOT be used as the primary explanation of the purpose of the form control. Instead, use the element with a for attribute set to the form control's id attribute. See Labels below.typeA string specifying the type of control to render. For example, to create a checkbox, a value of checkbox is used. If omitted (or an unknown value is specified), the input type text is used, creating a plaintext input field.Permitted values are listed in Input types above.valueThe input control's value. When specified in the HTML, this is the initial value, and from then on it can be altered or retrieved at any time using JavaScript to access the respective HTMLInputElement object's value property. The value attribute is always optional, though should be considered mandatory for checkbox, radio, and hidden.widthValid for the image input button only, the width is the width of the image file to display to represent the graphical submit button. See the image input type.The following non-standard attributes are also available on some browsers. As a general rule, you should avoid using them unless it can't be helped. Attribute Description incremental Whether or not to send repeated search events to allow updating live search results while the user is still editing the value of the field. WebKit and Blink only (Safari, Chrome, Opera, etc.). mozactionhint A string indicating the type of action that will be taken when the user presses the Enter or Return key while editing the field; this is used to determine an appropriate label for that key on a virtual keyboard. Since this attribute is deprecated, use enterkeyhint instead. orient Sets the orientation of the range slider. Firefox only. results The maximum number of items that should be displayed in the drop-down list of previous search queries. Safari only. webkitdirectory A Boolean indicating whether to only allow the user to choose a directory (or directories, if multiple is also present) incremental The Boolean attribute incremental is a WebKit and Blink extension (so supported by Safari, Opera, Chrome, etc.) which, if present, tells the user agent to process the input as a live search. As the user edits the value of the field, the user agent sends search events to the HTMLInputElement object representing the search box. This allows your code to update the search results in real time as the user edits the search.If incremental is not specified, the search event is only sent when the user explicitly initiates a search (such as by pressing the Enter or Return key while editing the field).The search event is rate-limited so that it is not sent more frequently than an implementation-defined interval.orient Similar to the -moz-orient non-standard CSS property impacting the and elements, the orient attribute defines the orientation of the range slider. Values include horizontal, meaning the range is rendered horizontally, and vertical, where the range is rendered vertically. See Creating vertical form controls for a modern approach to creating vertical form controls.results The results attributesupported only by Safariis a numeric value that lets you override the maximum number of entries to be displayed in the element's natively-provided drop-down menu of previous search queries. The value must be a non-negative decimal number. If not provided, or an invalid value is given, the browser's default maximum number of entries is used.webkitdirectory The Boolean webkitdirectory attribute, if present, indicates that only directories should be available to be selected by the user in the file picker interface. See HTMLInputElement.webkitdirectory for additional details and examples.Though originally implemented only for WebKit-based browsers, webkitdirectory is also usable in Microsoft Edge as well as Firefox 50 and later. However, even though it has relatively broad support, it is still not standard and should not be used unless you have no alternative.The following methods are provided by the HTMLInputElement interface which represents elements in the DOM. Also available are those methods specified by the parent interfaces, HTMLElement, Element, Node, and EventTarget.checkValidity()Returns true if the element's value passes validity checks; otherwise, returns false and fires an invalid event at the element.reportValidity()Returns true if the element's value passes validity checks; otherwise, returns false, fires an invalid event at the element, and (if the event isn't canceled) reports the problem to the user.select()Selects the entire content of the element, if the element's content is selectable. For elements with no selectable text content (such as a visual color picker or calendar date input), this method does nothing.setCustomValidity()Sets a custom message to display if the input element's value isn't valid.setRangeText()Sets the contents of the specified range of characters in the input element to a given string. A selectMode parameter is available to allow controlling how the existing content is affected.setSelectionRange()Selects the specified range of characters within a textual input element. Does nothing for inputs which aren't presented as text input fields.showPicker()Displays the browser picker for the input element that would normally be displayed when the element is selected, but triggered from a button press or other user interaction.stepDown()Decrements the value of a numeric input by one, by default, or by the specified number of units.stepUp()Increments the value of a numeric input by one or by the specified number of units.Inputs, being replaced elements, have a few features not applicable to non form elements. There are CSS selectors that can specifically target form controls based on their UI features, also known as UI pseudo-classes. The input element can also be targeted by type with attribute selectors. There are some properties that are especially useful as well. Pseudo-classes relevant to the element: Pseudo-class Description :enabled Any currently enabled element that can be activated (selected, clicked on, typed into, etc.) or accept focus and also has a disabled state, in which it can't be activated or accept focus. :disabled Any currently disabled element that has an enabled state, meaning it otherwise could be activated (selected, clicked on, typed into, etc.) or accept focus were it not disabled. :read-only Element not editable by the user :read-write Element that is editable by the user. :placeholder-shown Element that is currently displaying placeholder text, including and elements with the placeholder attribute present that has, as yet, no value. :default Form elements that are the default in a group of related elements. Matches checkbox and radio input types that were checked on page load or render. :checked Matches checkbox and radio input types that are currently checked (and the in a that is currently selected). :indeterminate checkbox elements whose indeterminate property is set to true by JavaScript, radio elements, when all radio buttons with the same name value in the form are unchecked, and elements in an indeterminate state :valid Form controls that can have constraint validation applied and are currently valid. :invalid Form controls that have constraint validation applied and are currently not valid. Matches a form control whose value doesn't match the constraints set on it by its attributes, such as required, pattern, step and max. :in-range A non-empty input whose current value is within the range limits specified by the min and max attributes and the step. :out-of-range A non-empty input whose current value is NOT within the range limits specified by the min and max attributes or does not adhere to the step constraint. :required , , or element that has the required attribute set on it. Only matches elements that can be required. The attribute included on a non-requirable element will not make for a match. :optional , , or element that does NOT have the required attribute set on it. Does not match elements that can't be required. :blank and elements that currently have no value. :user-invalid Similar to :invalid, but is activated on blur. Matches invalid input but only after the user interaction, such as by focusing on the control, leaving the control, or attempting to submit the form containing the invalid control. :open elements that display a picker for the user to choose a value from (for example) but only when the element is in the open state, that is, when the picker is displayed. We can style a checkbox label based on whether the checkbox is checked or not. In this example, we are styling the color and font-weight of the that comes immediately after a checked input. We haven't applied any styles if the input is not checked. input:checked + label { color: red; font-weight: bold;}It is possible to target different types of form controls based on their type using attribute selectors. CSS attribute selectors match elements based on either just the presence of an attribute or the

value of a given attribute./* matches a password input */input[type="password"] {}/* matches a form control whose valid values are limited to a range of values*/input[min][max] {}/* matches a form control with a pattern attribute */input[pattern] {}By default, the appearance of placeholder text is a translucent or light gray. The ::placeholder pseudo-element is the input's placeholder text. It can be styled with a limited subset of CSS properties.::placeholder { color: blue;}Only the subset of CSS properties that apply to the ::first-line pseudo-element can be used in a rule using ::placeholder in its selector.The appearance property enables the displaying of (almost) any element as a platform-native style based on the operating system's theme as well as the removal of any platform-native styling with the none value.You could make a look like a radio button with div {appearance: radio;} or a radio look like a checkbox with [type="radio"] {appearance: checkbox;}, but don't.Setting appearance: none removes platform native borders, but not functionality.A property specific to text entry-related elements is the CSS caret-color property, which lets you set the color used to draw the text input caret:HTMLNote the red caret:CSSinput.custom { caret-color: red; font: 16px "Helvetica", "Arial", "sans-serif";}ResultThe field-sizing property enables you to control the sizing behavior of form inputs (i.e., they are given a default preferred size by default.) This property enables you to override the default behavior, allowing form controls to adjust in size to fit their contents.This property is typically used to create form fields that shrinkwrap their content and grow as more text is entered. This works with input types that accept direct text input (for example, text and url), input type file, and elements.In certain cases (typically involving non-textual inputs and specialized interfaces), the element is a replaced element. When it is, the position and size of the element's size and positioning within its frame can be adjusted using the CSS object-position and object-fit properties.Labels are needed to associate assistive text with an . The element provides explanatory information about a form field that is always appropriate (aside from any layout concerns you have). It's never a bad idea to use a to explain what should be entered into an or .Associated labelsThe semantic pairing of and elements is useful for assistive technologies such as screen readers. By pairing them using the 's for attribute, you bond the label to the input in a way that lets screen readers describe inputs to users more precisely.It does not suffice to have plain text adjacent to the element. Rather, usability and accessibility requires the inclusion of either implicit or explicit :Enter your name: Enter your name: Enter your name: Enter your name: The first example is inaccessible: no relationship exists between the prompt and the element.In addition to an accessible name, the label provides a larger 'hit' area for mouse and touch screen users to click on or touch. By pairing a with an , clicking on either one will focus the . If you use plain text to "label" your input, this won't happen. Having the prompt part of the activation area for the input is helpful for people with motor control conditions.As web developers, it's important that we never assume that people will know all the things that we know. The diversity of people using the weband by extension your websitepractically guarantees that some of your site's visitors will have some variation in thought processes and/or circumstances that leads them to interpret your forms very differently from you without clear and properly-presented labels.Placeholders are not accessibleThe placeholder attribute lets you specify text that appears within the element's content area itself when it is empty. The placeholder should never be required to understand your forms. It is not a label, and should not be used as a substitute, because it isn't. The placeholder is used to provide a hint as to what an inputted value should look like, not an explanation or prompt.Not only is the placeholder not accessible to screen readers, but once the user enters any text into the form control, or if the form control already has a value, the placeholder disappears. Browsers with automatic page translation features may skip over attributes when translating, meaning the placeholder may not get translated.Note:Don't use the placeholder attribute if you can avoid it. If you need to label an element, use the element.Warning:Client-side validation is useful, but it does not guarantee that the server will receive valid data. If the data must be in a specific format, always verify it also on the server-side, and return a 400 HTTP response if the format is invalid.In addition to using CSS to style inputs based on the :valid or :invalid UI states based on the current state of each input, as noted in the UI pseudo-classes section above, the browser provides for client-side validation on (attempted) form submission. On form submission, if there is a form control that fails constraint validation, supporting browsers will display an error message on the first invalid form control; displaying a default message based on the error type, or a message set by you.Some input types and other attributes place limits on what values are valid for a given input. For example, means only the number 2, 4, 6, 8, or 10 are valid. Several errors could occur, including a rangeUnderflow error if the value is less than 2, rangeOverflow if greater than 10, stepMismatch if the value is a number between 2 and 10, but not an even integer (does not match the requirements of the step attribute), or typeMismatch if the value is not a number.For the input types whose domain of possible values is periodic (that is, at the highest possible value, the values wrap back around to the beginning rather than ending), it's possible for the values of the max and min properties to be reversed, which indicates that the range of permitted values starts at min, wraps around to the lowest possible value, then continues on until max is reached. This is particularly useful for dates and times, such as when you want to allow the range to be from 8 PM to 8 AM:Specific attributes and their values can lead to a specific error ValidityState: Validity object errors depend on the attributes and their values: Attribute Relevant property Description max validityState.rangeOverflow Occurs when the value is greater than the maximum value as defined by the max attribute maxlength validityState.tooLong Occurs when the number of characters is greater than the number allowed by the maxlength property min validityState.rangeUnderflow Occurs when the value is less than the minimum value as defined by the min attribute minlength validityState.tooShort Occurs when the number of characters is less than the number required by the minlength property pattern validityState.patternMismatch Occurs when a pattern attribute is included with a valid regular expression and the value does not match it. required validityState.valueMissing Occurs when the required attribute is present but the value is null or radio or checkbox is not checked. step validityState.stepMismatch The value doesn't match the step increment. Increment default is 1, so only integers are valid on type="number" is step is not included. step="any" will never throw this error. type validityState.typeMismatch Occurs when the value is not of the correct type, for example an email does not contain an @ or a url doesn't contain a protocol. If a form control doesn't have the required attribute, no value, or an empty string, is not invalid. Even if the above attributes are present, with the exception of required, an empty string will not lead to an error.We can set limits on what values we accept, and supporting browsers will natively validate these form values and alert the user if there is a mistake when the form is submitted.In addition to the errors described in the table above, the validityState interface contains the badInput, valid, and customError boolean readonly properties. The validity object includes: For each of these Boolean properties, a value of true indicates that the specified reason validation may have failed is true, with the exception of the valid property, which is true if the element's value obeys all constraints.If there is an error, supporting browsers will both alert the user and prevent the form from being submitted. A word of caution: if a custom error is set to a truthy value (anything other than the empty string or null), the form will be prevented from being submitted. If there is no custom error message, and none of the other properties return true, valid will be true, and the form can be submitted.function validate(input) { let validityState_object = input.validity; if (validityState_object.valueMissing) { input.setCustomValidity("A value is required"); } else if (validityState_object.rangeUnderflow) { input.setCustomValidity("Your value is too low"); } else if (validityState_object.rangeOverflow) { input.setCustomValidity("Your value is too high"); } else { input.setCustomValidity(""); }}The last line, setting the custom validity message to the empty string is vital. If the user makes an error, and the validity is set, it will fail to submit, even if all the values are valid, until the message is null.Custom validation error exampleIf you want to present a custom error message when a field fails to validate, you need to use the Constraint Validation API available on (and related) elements. Take the following form: Enter username (upper and lowercase letters): SubmitThe basic HTML form validation features will cause this to produce a default error message if you try to submit the form with either no valid filled in, or a value that does not match the pattern.If you wanted to instead display custom error messages, you could use JavaScript like the following:const nameInput = document.querySelector("input");nameInput.addEventListener("input", () => { nameInput.setCustomValidity(""); nameInput.checkValidity();});nameInput.addEventListener("invalid", () => { if (nameInput.value === "") { nameInput.setCustomValidity("Enter your username!"); } else { nameInput.setCustomValidity("Usernames can only contain upper and lowercase letters. Try again!",); }});The example renders like so: In brief:We check the valid state of the input element every time its value is changed by running the checkValidity() method via the input event handler.If the value is invalid, an invalid event is raised, and the invalid event handler function is run. Inside this function we work out whether the value is invalid because it is empty, or because it doesn't match the pattern, using an if () block, and set a custom validity error message.As a result, if the input value is invalid when the submit button is pressed, one of the custom error messages will be shown.If it is valid, it will submit as you'd expect. For this to happen, the custom validity has to be cancelled, by invoking setCustomValidity() with an empty string value. We therefore do this every time the input event is raised. If you don't do this, and a custom validity was previously set, the input will register as invalid, even if it currently contains a valid value on submission.Note:Always validate input constraints both client side and server side. Constraint validation doesn't remove the need for validation on the server side. Invalid values can still be sent by older browsers or by bad actors.Note:Firefox supported a proprietary error attribute x-moz-errormessage for many versions, which allowed you set custom error messages in a similar way. This has been removed as of version 66 (see Firefox bug 1513890).The allowed inputs for certain types depend on the locale. In some locales, 1,000.00 is a valid number, while in other locales the valid way to enter this number is 1.000,00.Firefox uses the following heuristics to determine the locale to validate the user's input (at least for type="number"):Try the language specified by a lang/xml:lang attribute on the element or any of its parents.Try the language specified by any Content-Language HTTP header. Or,If none specified, use the browser's locale.When including inputs, it is an accessibility requirement to add labels alongside. This is needed so those who use assistive technologies can tell what the input is for. Also, clicking or touching a label gives focus to the label's associated form control. This improves the accessibility and usability for sighted users, increases the area a user can click or touch to activate the form control. This is especially useful (and even needed) for radio buttons and checkboxes, which are tiny. For more information about labels in general see Labels.The following is an example of how to associate the with an element in the above style. You need to give the an id attribute. The then needs a for attribute whose value is the same as the input's id.Do you like peas?Interactive elements such as form input should provide an area large enough that it is easy to activate them. This helps a variety of people, including people with motor control issues and people using non-precise forms of input such as a stylus or fingers. A minimum interactive size of 4444 CSS pixels is recommended. Content categories Flow content, listed, submittable, resettable, form-associated element, phrasing content. If the type is not hidden, then labelable element, palpable content. Permitted content None; it is a void element. Tag omission Must have a start tag and must not have an end tag. Permitted parents Any element that accepts phrasing content. Implicit ARIA role Permitted ARIA roles type=button: checkbox, combobox, link, menuitem, menuitemcheckbox, menuitemradio, option, radio, switch, tab type=checkbox: button when used with aria-pressed, menuitemcheckbox, option, switch type=image: link, menuitem, menuitemcheckbox, menuitemradio, radio, switch type=radio: menuitemradio type=text with no list attribute: combobox, searchbox, spinbutton type=color|date|datetime-local|email|file|hidden| month|number|password|range|reset|search|submit|tel|url|week or text with list attribute: no role permitted DOM interface HTMLInputElement SpecificationHTML # the-input-element An HTML form with three input fields; two text fields and one submit button: First name: Last name: Try it Yourself The tag specifies an input field where the user can enter data. The element is the most important form element. The element can be displayed in several ways, depending on the type attribute.The different input types are as follows: (default value) Look at the type attribute to see examples for each input type! Tips and NotesTip: Always use the tag to define labels for , , , , and . Browser Support Element Yes Yes Yes Yes Yes Attribute Value Description accept file_extension audio/* video/* image/* media_type Specifies a filter for what file types the user can pick from the file input dialog box (only for type="file") alt text Specifies an alternate text for images (only for type="image") autocomplete on off Specifies whether an element should have autocomplete enabled autofocus autofocus Specifies that an element should automatically get focus when the page loads checked checked Specifies that an element should be pre-selected when the page loads (for type="checkbox" or type="radio") dirname inputname.dir Specifies that the text direction will be submitted disabled disabled Specifies that an element should be disabled form form_id Specifies the form the element belongs to formaction URL Specifies the URL of the file that will process the input control when the form is submitted (for type="submit" and type="image") formenctype application/x-www-form-urlencoded multipart/form-data text/plain Specifies how the form-data should be encoded when submitting it to the server (for type="submit" and type="image") formmethod getpost Defines the HTTP method for sending data to the action URL (for type="submit" and type="image") formnovalidate formnovalidate Defines that form elements should not be validated when submitted formtarget _blank_self_parent_top framename Specifies where to display the response that is received after submitting the form (for type="submit" and type="image") height pixels Specifies the height of an element (only for type="image") list datalist_id Refers to a element that contains pre-defined options for an element max number date Specifies the maximum value for an element maxlength number Specifies the maximum number of characters allowed in an element min number date Specifies a minimum value for an element minlength number Specifies the minimum number of characters required in an element multiple multiple Specifies that a user can enter more than one value in an element name text Specifies the name of an element pattern regexp Specifies a regular expression that an element's value is checked against placeholder text Specifies a short hint that describes the expected value of an element popovertarget element_id Specifies which popover element to invoke (only for type="button") popovertargetaction hideshowtoggle Specifies what happens to the popover element when you click the button (only for type="button") readonly readonly Specifies that an input field is read-only required required Specifies that an input field must be filled out before submitting the form size number Specifies the width, in characters, of an element src URL Specifies the URL of the image to use as a submit button (only for type="image") step numberany Specifies the interval between legal numbers in an input field type button checkbox color date datetime-local email file hidden image month number password radio range reset search submit tel text time url week Specifies the type element to display value text Specifies the value of an element width pixels Specifies the width of an element (only for type="image") The tag also supports the Global Attributes in HTML. Event AttributesThe tag also supports the Event Attributes in HTML. Related PagesHTML tutorial: HTML DOM reference: Default CSS SettingsNone.

Grade 6 input output worksheets. Grade 6 math input output machine worksheets.