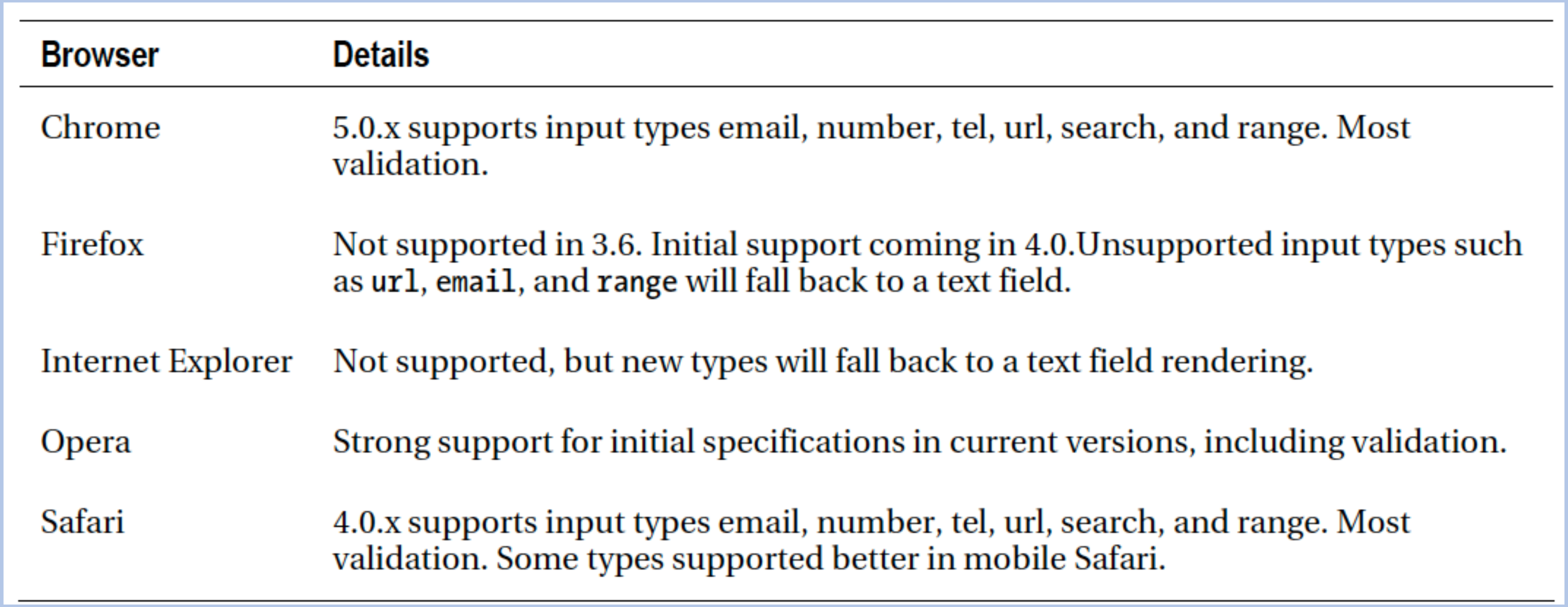
HTML5 Forms

Overview of HTML5 Forms

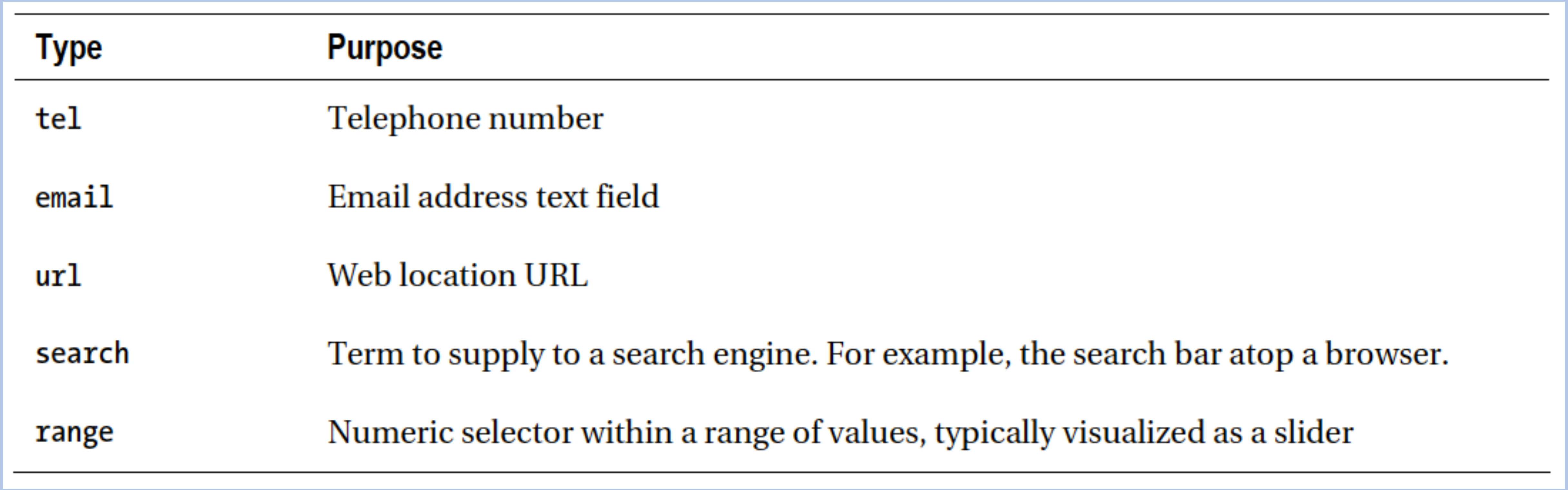
- Forms should still be encapsulated in a <form> element where the basic submission attributes are set.
- Forms still send the values of the controls to the server when the user or the application programmer submits the page.
- All of the familiar form controls—text fields, radio buttons, check boxes, and so on—are still present and working as before (albeit with some new features).
- Form controls are still fully scriptable for those who wish to write their own modifiers and handlers

Browser Support



Browser	Details
Chrome	5.0.x supports input types email, number, tel, url, search, and range. Most validation.
Firefox	Not supported in 3.6. Initial support coming in 4.0.Unsupported input types such as `url`, `email`, and `range` will fall back to a text field.
Internet Explorer	Not supported, but new types will fall back to a text field rendering.
Opera	Strong support for initial specifications in current versions, including validation.
Safari	4.0.x supports input types email, number, tel, url, search, and range. Most validation. Some types supported better in mobile Safari.

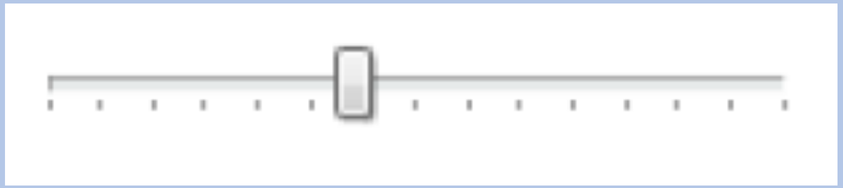
New Input Types



Type	Purpose
`tel`	Telephone number
`email`	Email address text field
`url`	Web location URL
`search`	Term to supply to a search engine. For example, the search bar atop a browser.
`range`	Numeric selector within a range of values, typically visualized as a slider

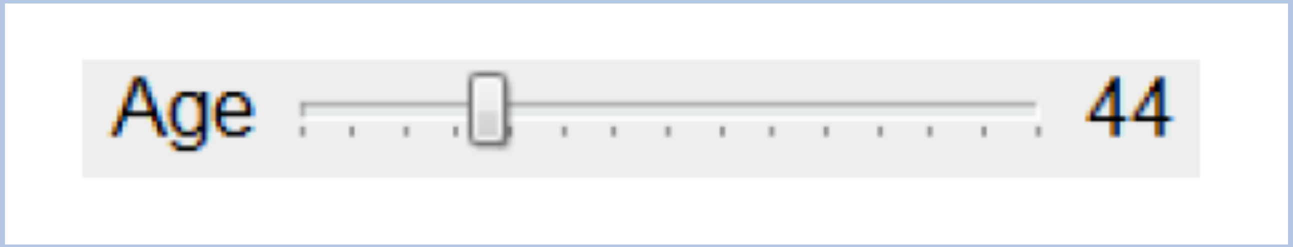
New Input Types

- <input type="email">
- <input type="text">
- <input type="range" min="18" max="120">



onchange handler to update a display field

-
  ```
  <script type="text/javascript">
  function showValue(newVal) {
      document.getElementById("ageDisplay").innerHTML = newVal;
  }
  </script>
  <label for="age">Age</label>
  <input id="age" type="range" min="18" max="120" value="18"
                  onchange="showValue(this.value)”/>
  <span id="ageDisplay">18</span>
  ```



Future HTML5 Types

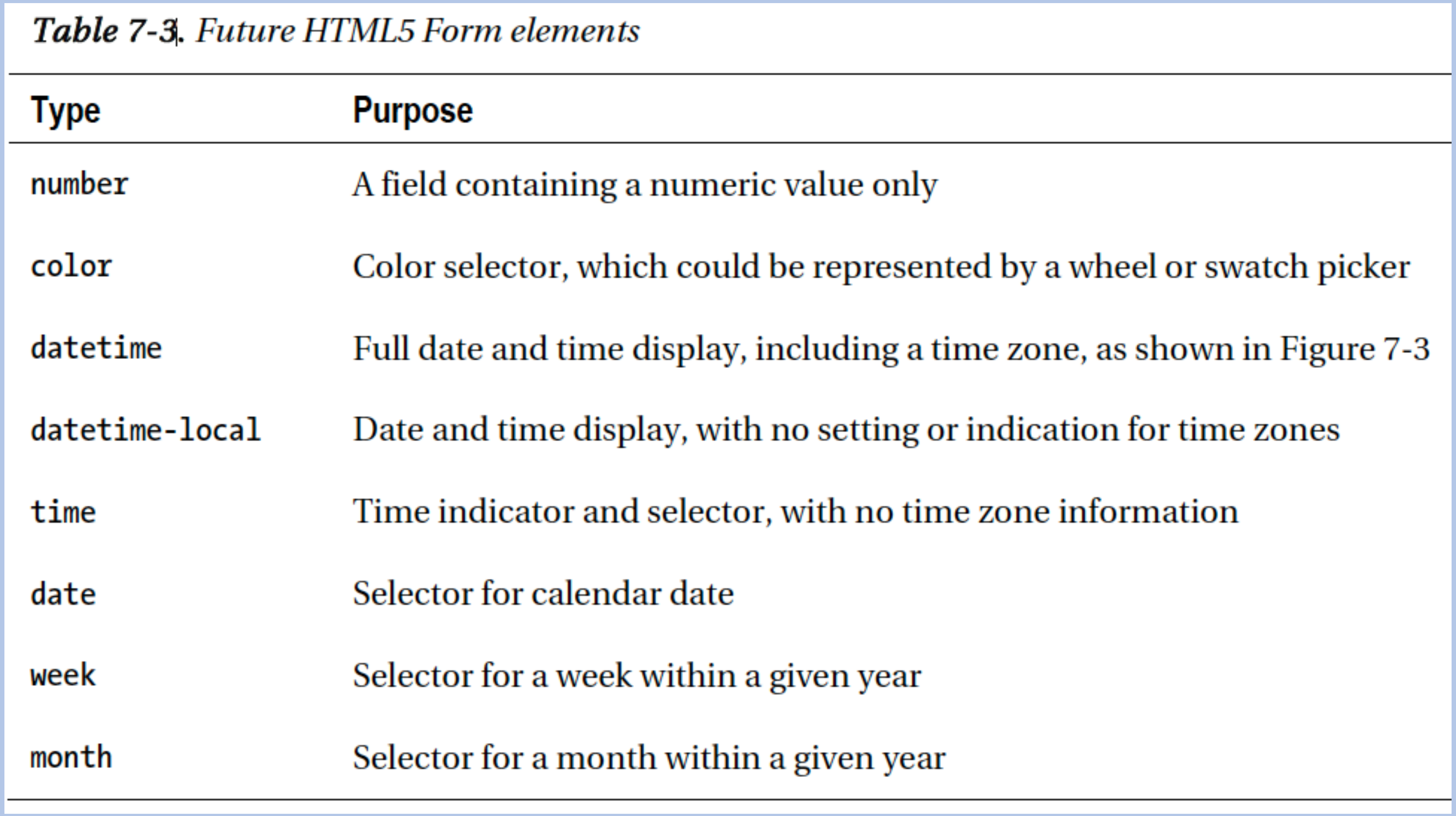
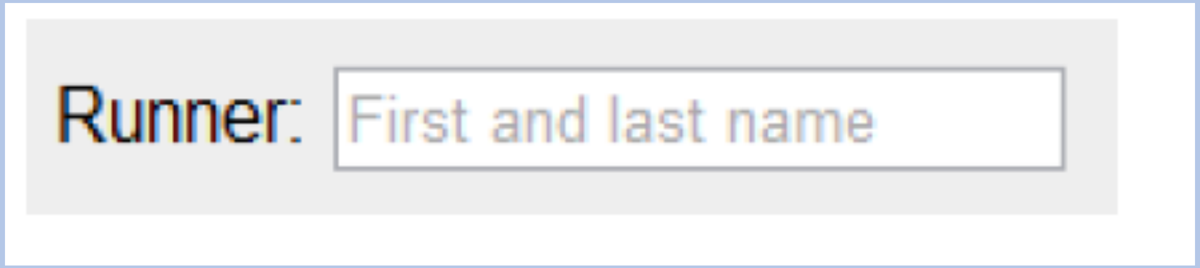


***Table 7-3**. Future HTML5 Form elements*

Type	Purpose
`number`	A field containing a numeric value only
`color`	Color selector, which could be represented by a wheel or swatch picker
`datetime`	Full date and time display, including a time zone, as shown in Figure 7-3
`datetime-local`	Date and time display, with no setting or indication for time zones
`time`	Time indicator and selector, with no time zone information
`date`	Selector for calendar date
`week`	Selector for a week within a given year
`month`	Selector for a month within a given year

The placeholder Attribute

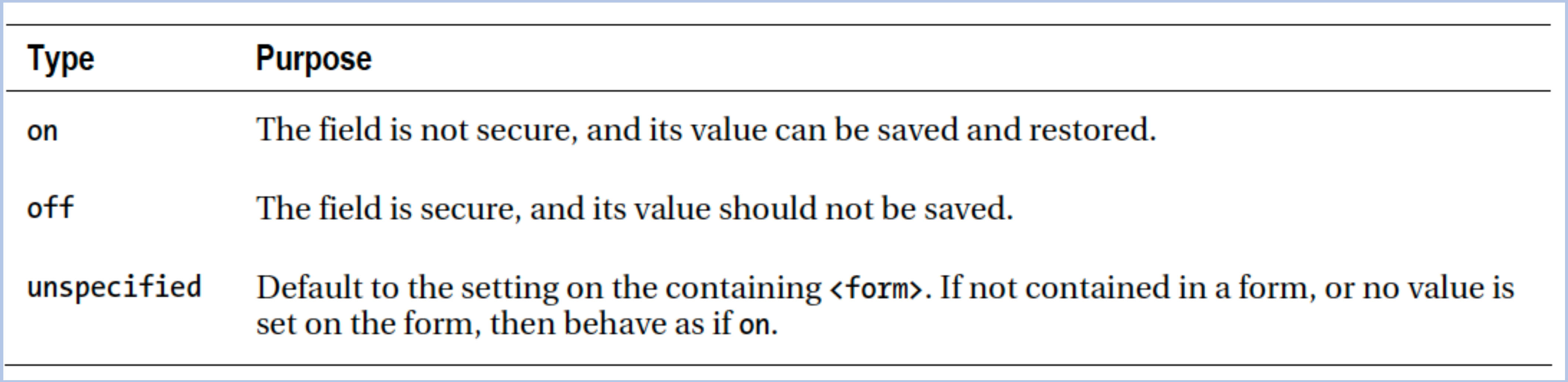
- <label>Runner: <input name="name" placeholder="First and last name" required></label>



The autocomplete Attribute

- <input type="text" name="creditcard" autocomplete="off">

Autocomplete values:



Type	Purpose
`on`	The field is not secure, and its value can be saved and restored.
`off`	The field is secure, and its value should not be saved.
`unspecified`	Default to the setting on the containing `<form>`. If not contained in a form, or no value is set on the form, then behave as if `on`.

The autofocus Attribute

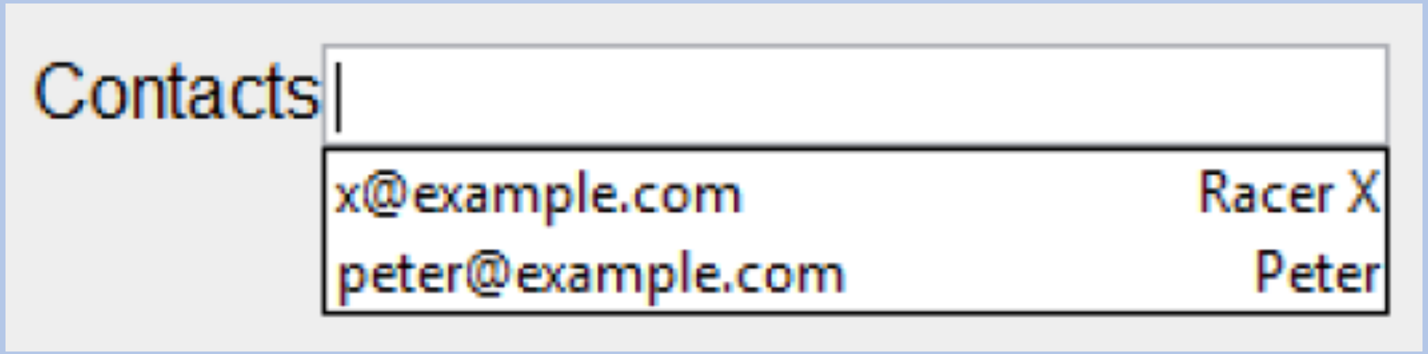
- The autofocus attribute lets a developer specify that a given form element should take input focus immediately when the page loads. Only one attribute per page should specify the autofocus attribute. Behavior is undefined if more than one control is set to autofocus.

  ```
  <input type="search" name="criteria" autofocus>
  ```

The list Attribute and the datalist Element

-
  ```
  <datalist id="contactList">
  <option value="x@example.com" label="Racer X">
  <option value="peter@example.com" label="Peter">
  </datalist>

  <input type="email" id="contacts" list="contactList">
  ```



The min and max Attributes

- `<input id="confidence" name="level" type="range" min="0" max="100" value="0">`

The step Attribute

- <input id="confidence" name="level" type="range" min="0" max="100" step="5" value="0">

The valueAsNumber Function

- document.getElementById("confidence").valueAsNumber(65);

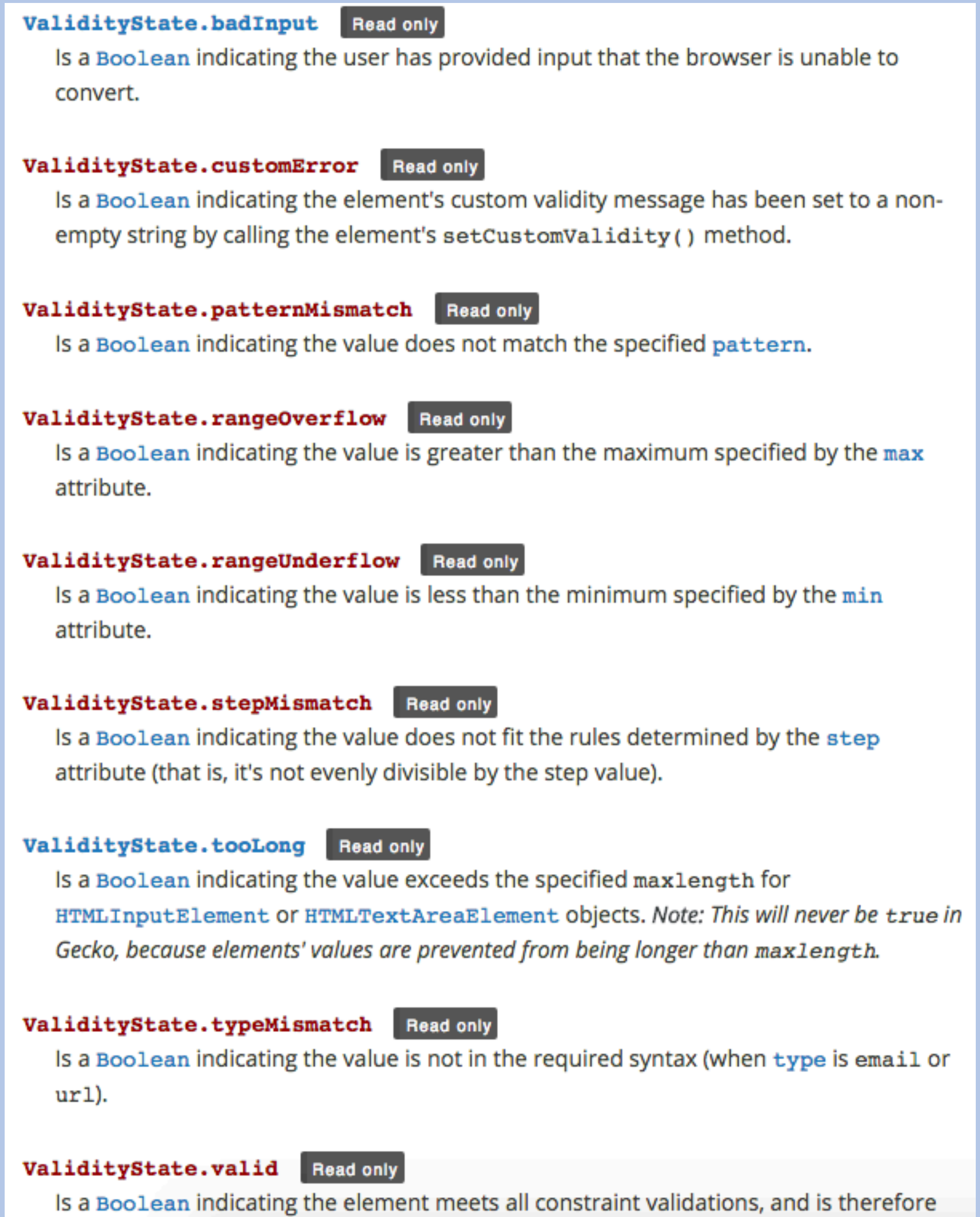
The required Attribute

- <input type="text" id="firstname" name="first" required>

Checking forms with validation

- ValidityState:

  ```
  var valCheck = document.myForm.myInput.validity;
  ```



`ValidityState.badInput` Read only

Is a `Boolean` indicating the user has provided input that the browser is unable to convert.

`ValidityState.customError` Read only

Is a `Boolean` indicating the element's custom validity message has been set to a non-empty string by calling the element's `setCustomValidity()` method.

`ValidityState.patternMismatch` Read only

Is a `Boolean` indicating the value does not match the specified `pattern`.

`ValidityState.rangeOverflow` Read only

Is a `Boolean` indicating the value is greater than the maximum specified by the `max` attribute.

`ValidityState.rangeUnderflow` Read only

Is a `Boolean` indicating the value is less than the minimum specified by the `min` attribute.

`ValidityState.stepMismatch` Read only

Is a `Boolean` indicating the value does not fit the rules determined by the `step` attribute (that is, it's not evenly divisible by the step value).

`ValidityState.tooLong` Read only

Is a `Boolean` indicating the value exceeds the specified `maxlength` for `HTMLInputElement` or `HTMLTextAreaElement` objects. *Note: This will never be `true` in Gecko, because elements' values are prevented from being longer than `maxlength`.*

`ValidityState.typeMismatch` Read only

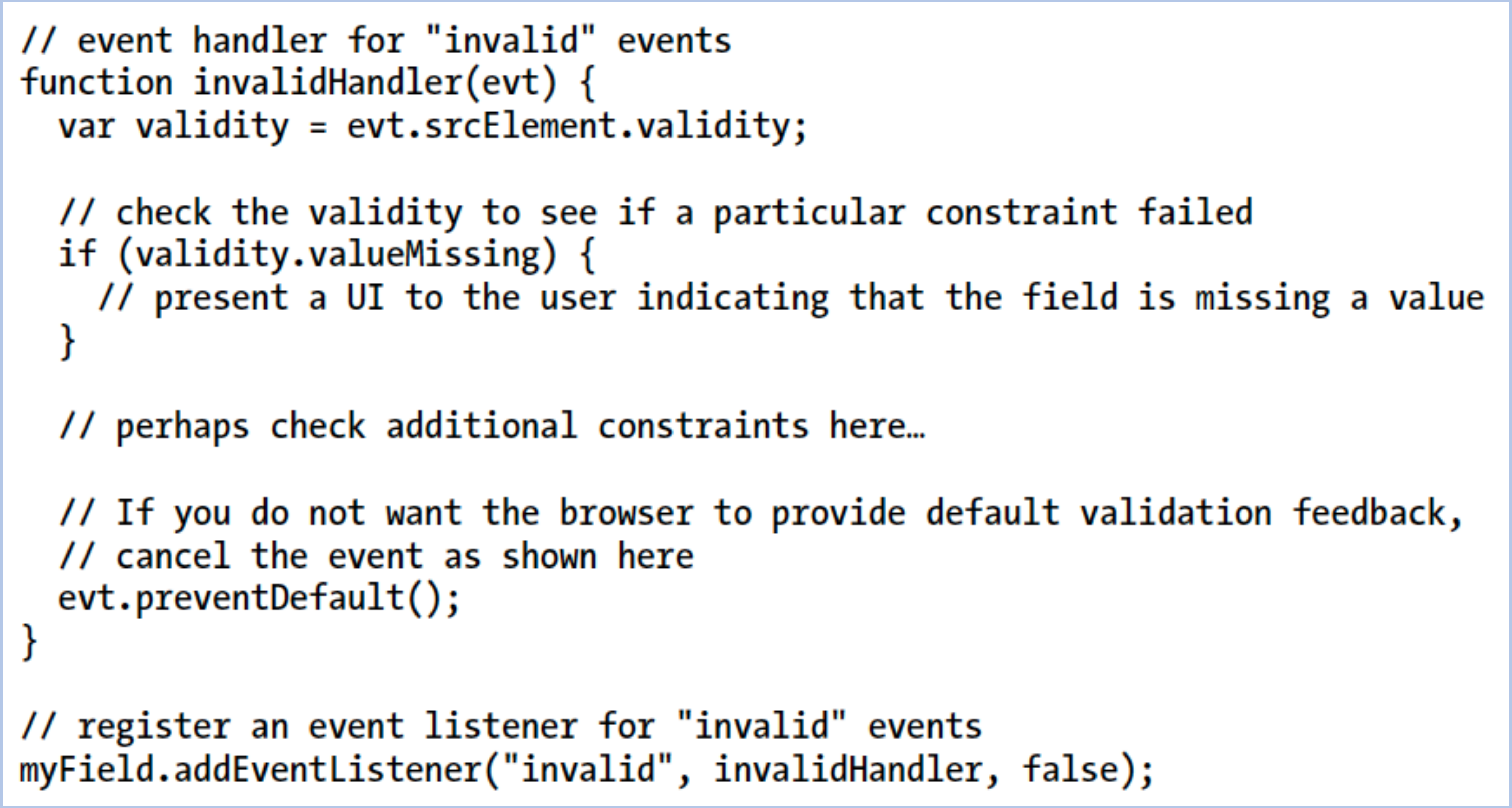
Is a `Boolean` indicating the value is not in the required syntax (when `type` is `email` or `url`).

`ValidityState.valid` Read only

Is a `Boolean` indicating the element meets all constraint validations, and is therefore

- The willValidate Attribute - Indicates whether validation will be checked on this form control at all.
- CheckValidity Function - Allows you to check validation on the form without any explicit user input.
- validationMessage Attribute - This attribute isn’t yet supported by any current browser versions, lets you query programmatically a localized error message that the browser would display based on the current state of validation.

Validation feedback



```
// event handler for "invalid" events
function invalidHandler(evt) {
  var validity = evt.srcElement.validity;

  // check the validity to see if a particular constraint failed
  if (validity.valueMissing) {
    // present a UI to the user indicating that the field is missing a value
  }

  // perhaps check additional constraints here…

  // If you do not want the browser to provide default validation feedback,
  // cancel the event as shown here
  evt.preventDefault();
}

// register an event listener for "invalid" events
myField.addEventListener("invalid", invalidHandler, false);
```

Validation feedback

```
// event handler for "invalid" events
function invalidHandler(evt) {
  var validity = evt.srcElement.validity;

  // check the validity to see if a particular constraint failed
  if (validity.valueMissing) {
    // present a UI to the user indicating that the field is missing a value
  }

  // perhaps check additional constraints here…

  // If you do not want the browser to provide default validation feedback,
  // cancel the event as shown here
  evt.preventDefault();
}

// register an event listener for "invalid" events
myField.addEventListener("invalid", invalidHandler, false);
```