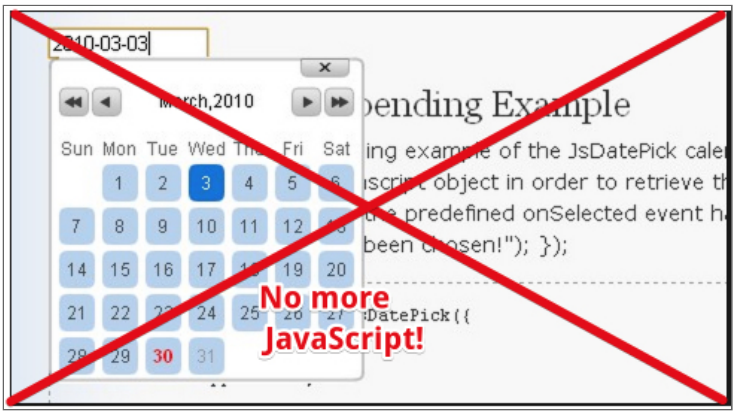
The `<input type="date">` and its variants (`datetime`, `datetime-local`, `time`, `month`, `week`)

INTRODUCTION



For years, date and time pickers in HTML forms made Web developers rely heavily on JavaScript based widgets. The process is simpler in HTML5, which provides a special control to handle this specific kind of data natively. **All mobile browsers support `<input type="time">` and `<input type="date">`,** while support on desktop computers is not yet as good.

Below are a few screenshots of the HTML5 date picker on several mobile devices. Note that the native date pickers of the operating systems are used:

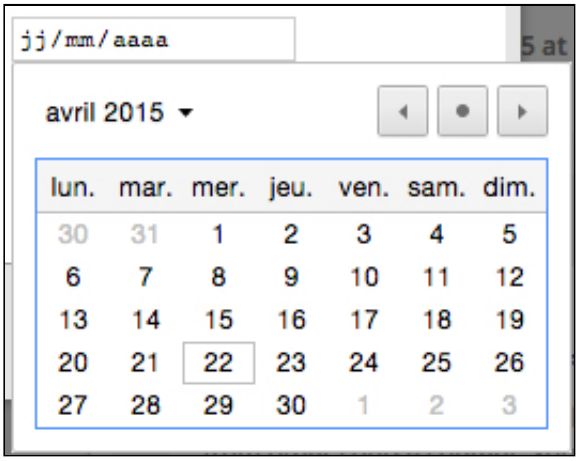
The problem is different on a desktop. While it's great to have native support for a date picker, Web developers would sometimes prefer 100% control over the *look and feel* of the date picker widget. For this purpose, the solution undoubtedly lies with the new Web Components (a way to make custom reusable widgets in HTML/CSS/JS), that will be detailed in the HTML5 Part-2 course.

In this course, we will focus on native implementations. Desktop support is currently not 100% (see the "current support" section below), so it is better to try the following

examples with Opera or Chrome.

> Why don't you try it yourself? Just click on this input field:
> mm/dd/yyyy

With Google Chrome desktop, it shows this date picker widget:



On non-supported browsers, it defaults to an `<input type="text">` input field.

TYPICAL USE OF `<INPUT TYPE="DATE">`

Default use

The default usage is something like:

```
<label for="birthday">Choose birthday party date: </label>
<input type="date" id="birthday">
```

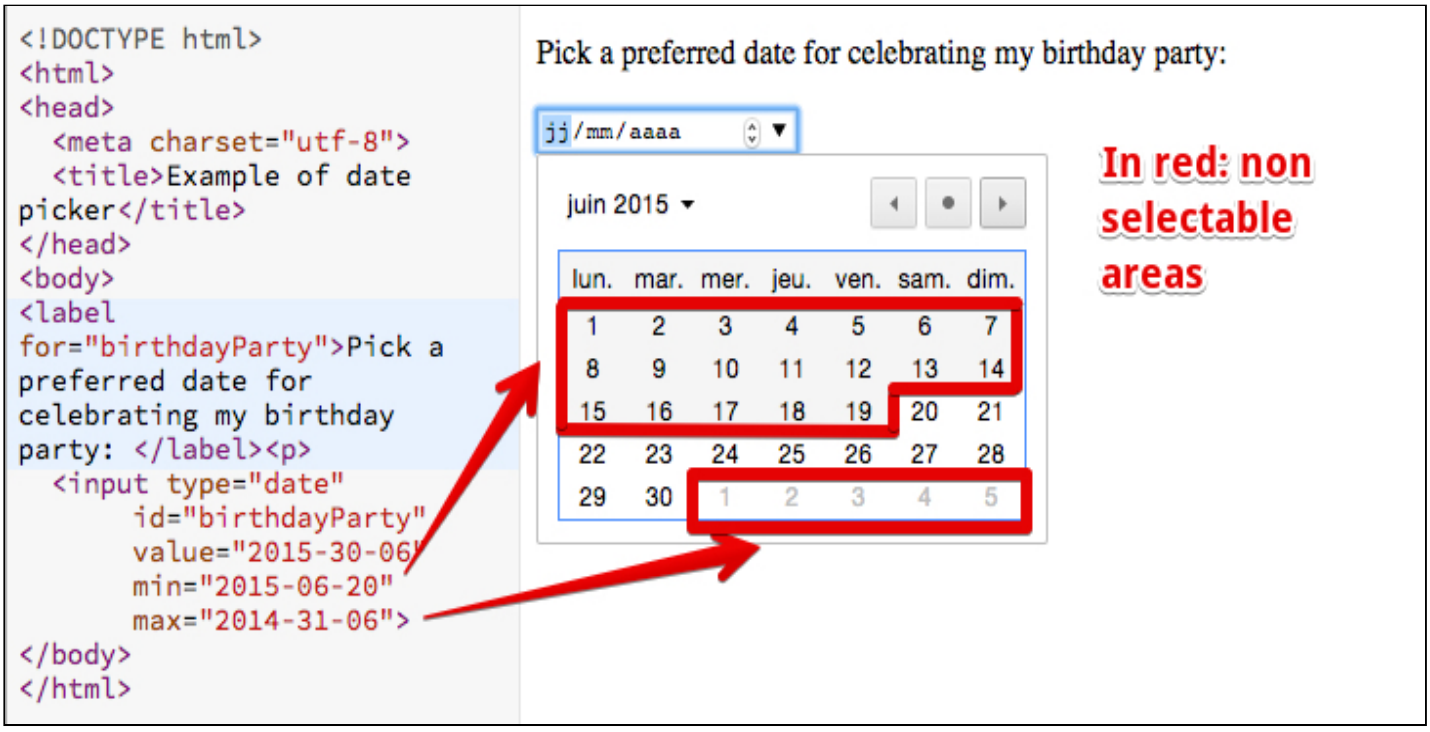
> Result: Choose birthday party date: mm/dd/yyyy

Most of the time you will add other attributes to give some restrictions (choose a date in the past, in the future, only on a Saturday, etc.).

Restrict choice to an interval of dates: attributes `min`, `max` and `value`

The `<input type="date">` comes with several useful attributes. In particular the `value`, `min` and `max` attributes are used to propose a default date, a min and a max date, or for defining an interval of acceptable values.

Try this example: just click the next input field: 06/30/2015, or try it online on JS Bin if you want to tweak the source code:



Errata: the online example is correct, but the above screenshot shows a year equal to "2014" for the `max` attribute, which is incorrect.

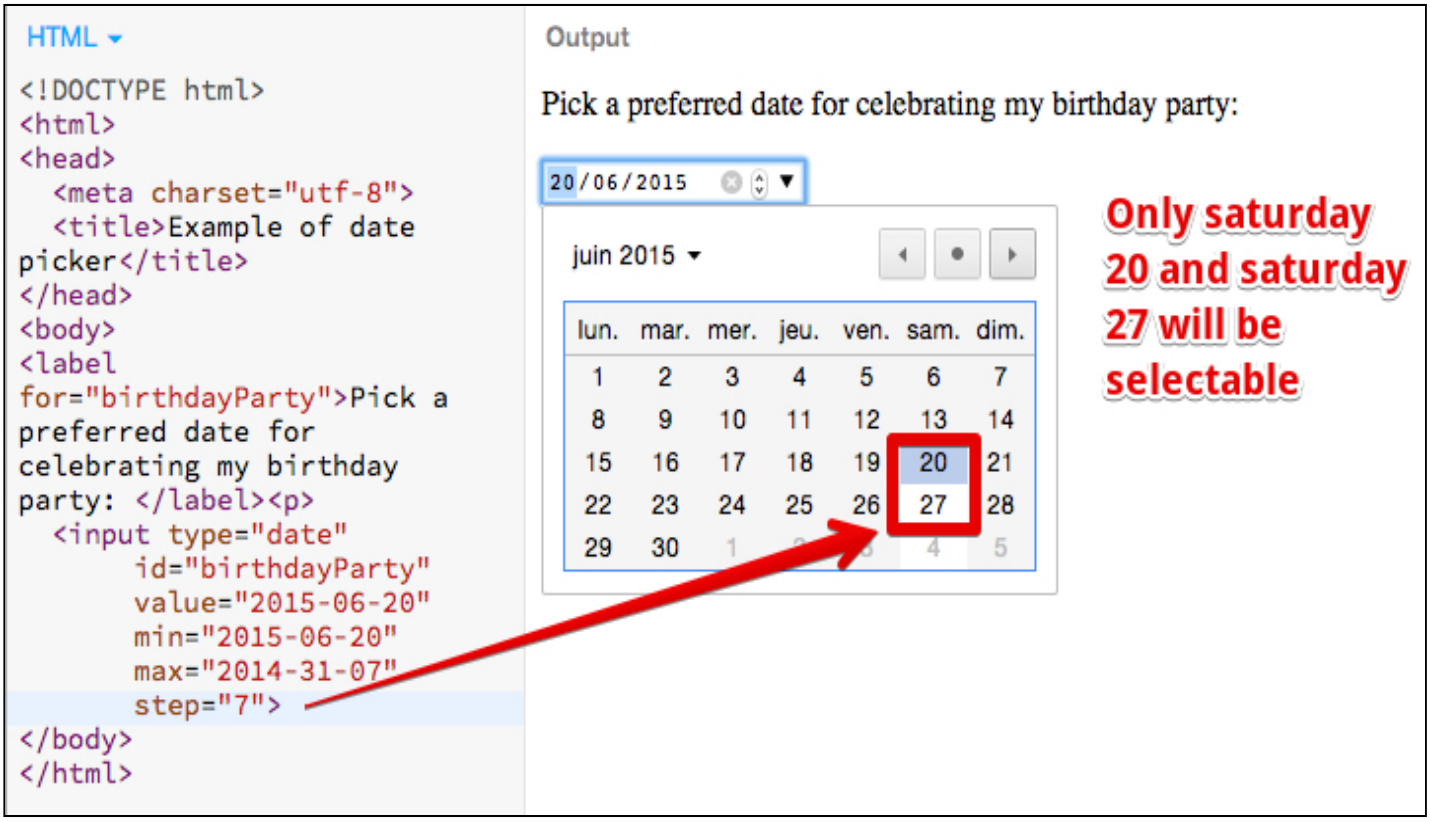
Source code:

```
...
 <input type="date"
3.      id="birthdayParty"
        value="2015-06-20"
        min="2015-06-20"
        max="2015-07-31">
...
```

Choosing one day in a given week, etc. with the `step`: attribute

Using the `value` attribute for setting a date, and using `step=7` for example, will make acceptable only the day of the week that corresponds to the value's day (e.g.: only Mondays). Using `step=2` will make acceptable only every other day, etc.

Example: we want to celebrate birthday parties only on Saturdays, check this on JS Bin!



Errata: the online example is correct, but the above screenshot shows a year equal to "2014" for the `max` attribute, which is incorrect.

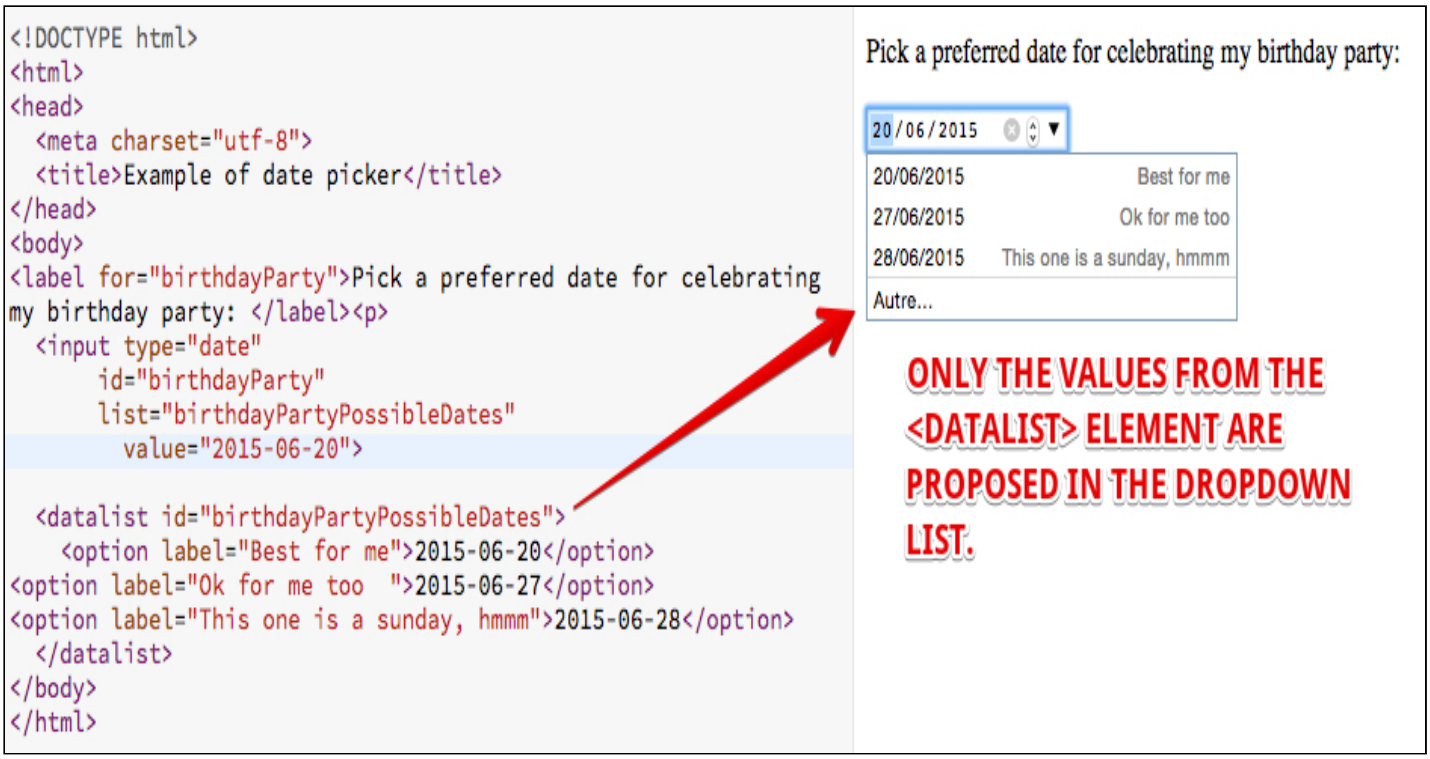
Extract from source code:

```
<input type="date"
    id="birthdayParty"
    value="2015-06-20"
    min="2015-06-20"
    max="2015-07-31"
    step="7">
```

Combining with the `<datalist>` element to restrict the choice of possible values

This feature is not yet (in 2015) supported by all browsers, check this compatibility table.

Online example at JS Bin



Extract from source code:

```
    <input type="date"
        id="birthdayParty"
        list="birthdayPartyPossibleDates"
        value="2015-06-20">
    <datalistid="birthdayPartyPossibleDates">
         <option label="Best for me">2015-06-20</option>
         <option label="Ok for me too ">2015-06-27</option>
         <option label="This one is a sunday, hmmm">2015-06-
28</option>
10.  </datalist>
```

The `list` attribute of the input element must match the `id`attribute of the `datalist` element.

If you use the `min`, `max`, or `step` attributes with a `list`attribute, it may filter the restricted list even more. Checkthis example on JS Bin (tested with Google Chrome), that has a restricted list of three elements, one of which is filtered because it is not in in the min/max range.

RESPONDING TO DATE CHANGES, TRYING DATE/TIME AND OTHER VARIANTS

Listening to the input event

Here is an interactive example at JS Bin where you can change the type of date/time chooser. It also shows how to listen to the input event when a date/time is chosen.

Source code:

```
<!DOCTYPE html>
<html>
<body>
Testing the new date input field. So far works only in Chrome
and Opera browsers.<p>
Choose a date/time : <input type="date"id="date" /><p>
<p>
You picked: <span id="pickedDate"></span>
 </p>
After you have tried the first example, change the value of
the "type" attribute to:
<ul>
<li>datetime</li>
<li>datetime-local</li>
<li>time</li>
<li>week</li>
<li>month</li>
</ul>
And see the result.
<script>
```

```
20.  var field =document.querySelector("#date");
     var result =document.querySelector("#pickedDate");
    field.oninput = function(evt) {
       var date = this.value;
       pickedDate.innerHTML = "<b>"+date+"</b>";
    }
     </script>
    </body>
    </html>
```

Lines 20-26 show how we can detect a date change using JavaScript.

Checking if the chosen date is in the past or in the future using the `valueAsDate` property

The object returned to the input event handler has a useful property named `valueAsDate`. This is a JavaScript date object that can be compared to other JavaScript date objects, in particular to the date of the day we can get with `var date = new Date();`

The following example at JS Bin shows how to ascertain whether a date is in the past or in the future:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Example of date picker</title>
</head>
<body>
<label for="birthDate">Enter your birth date: </label><p>
  <input type="date" id="birthDate" >
  <p>
  You picked: <span id="pickedDate"></span><p>
  <span id="pastFuture"></span>
  </p>
 <script>
  var field = document.querySelector("#birthDate");
  var result = document.querySelector("#pickedDate");
   var pastFuture = document.querySelector("#pastFuture");

  field.oninput = function(evt) {
    var date = this.value;
    pickedDate.innerHTML = "<b>"+date+"</b>";

    if(this.valueAsDate <= new Date()) {
      pastFuture.style.color = 'green';
      pastFuture.innerHTML = "<b>Date in the past, ok!</b>"
    } else {
      pastFuture.style.color = 'red';
      pastFuture.innerHTML = "<b>Date in the future, you're not even born!
</b>"
    }
  }
  </script>
</body>
</html>
```

Enter your birth date:

17/04/2015

You picked: **2015-04-17**

Date in the past, ok!

While if we enter a date in the future:

Enter your birth date:

24/04/2015

You picked: **2015-04-24**

Date in the future, you're not even born!

Extract from source code:

```
<body>
<label for="birthDate">Enter your birth date: </label><p>
 <input type="date" id="birthDate" >
 <p>
You picked: <span id="pickedDate"></span><p>
 <span id="pastFuture"></span>
 </p>
 <script>
 var field =document.querySelector("#birthDate");
 var result =document.querySelector("#pickedDate");
 var pastFuture =document.querySelector("#pastFuture");
field.oninput = function(evt) {
   var date = this.value;
   pickedDate.innerHTML = "<b>"+date+"</b>";
   if(date.valueAsDate <= new Date()) {
       pastFuture.style.color = 'green';
       pastFuture.innerHTML = "<b>Date in the past, ok!</b>"
   } else {
       pastFuture.style.color = 'red';
        pastFuture.innerHTML = "<b>Date in the future, you're
not even born!</b>"
   }
 }
 </script>
</body>
```

Lines 17-23 show how we can compare the date picked in the calendar widget with the current date. Note that we can compare any given dates using JavaScript. To check that the chosen date is before 2000 we would do this:

```
if(this.valueAsDate <= newDate(2000,1,1)) {
...
}
```

HTML 5.1: `<INPUT TYPE="DATETIME">`, "WEEK", "MONTH",

"DATETIME-LOCAL", ETC.

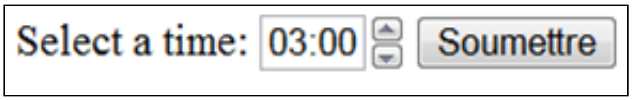
The HTML5 specification indicates that we can use `<input type="date">` and `<input type="time">` while for some years (before the specification became a frozen standard in October 2014), other variants were also present, such as`type=datetime`, `datetime-local`, `month` and `week`.

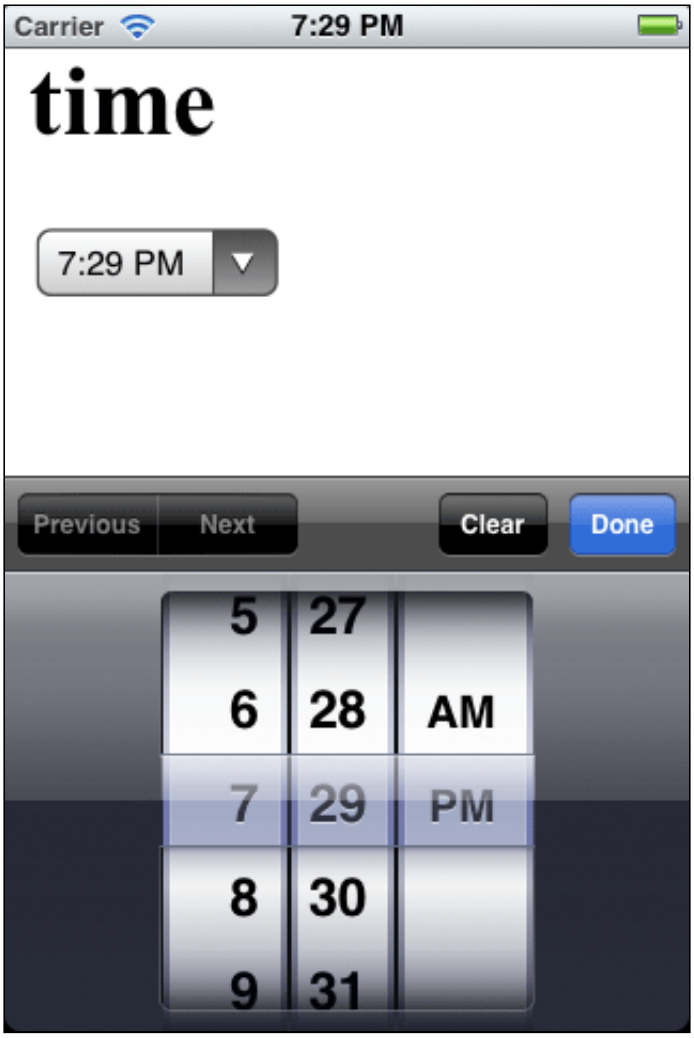
These variants have been moved to the HTML 5.1 specification (still a work in progress) because so far they have only been implemented by Chrome and Opera.

Here is an interactive example at JS Bin where you can change the type of date chooser and try all the different possible values for the type attribute of date pickers.

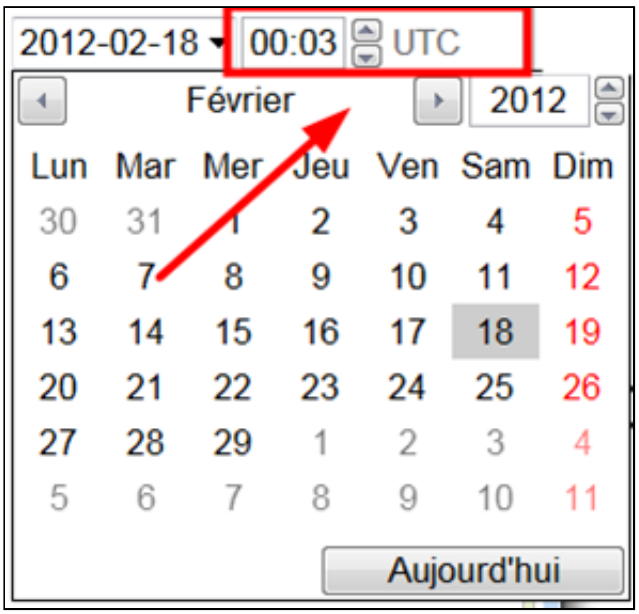
Some screenshots from Opera desktops and Safari IOS:

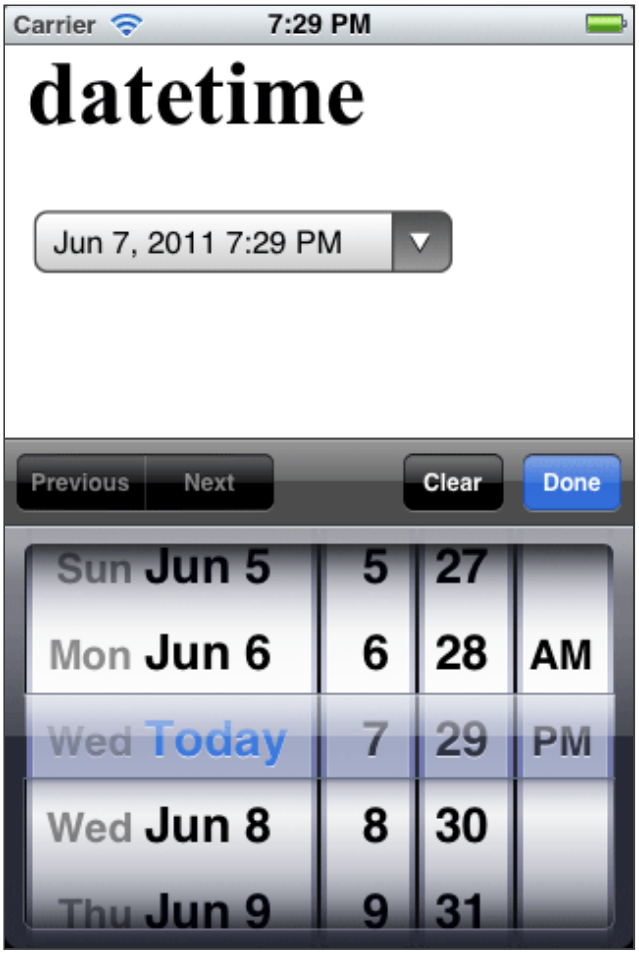
`<input type="time">`:



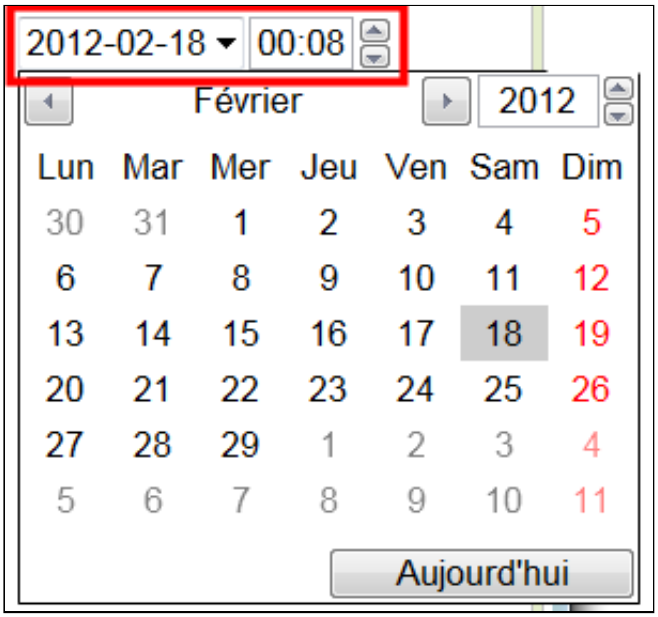



```
<input type="datetime">
```

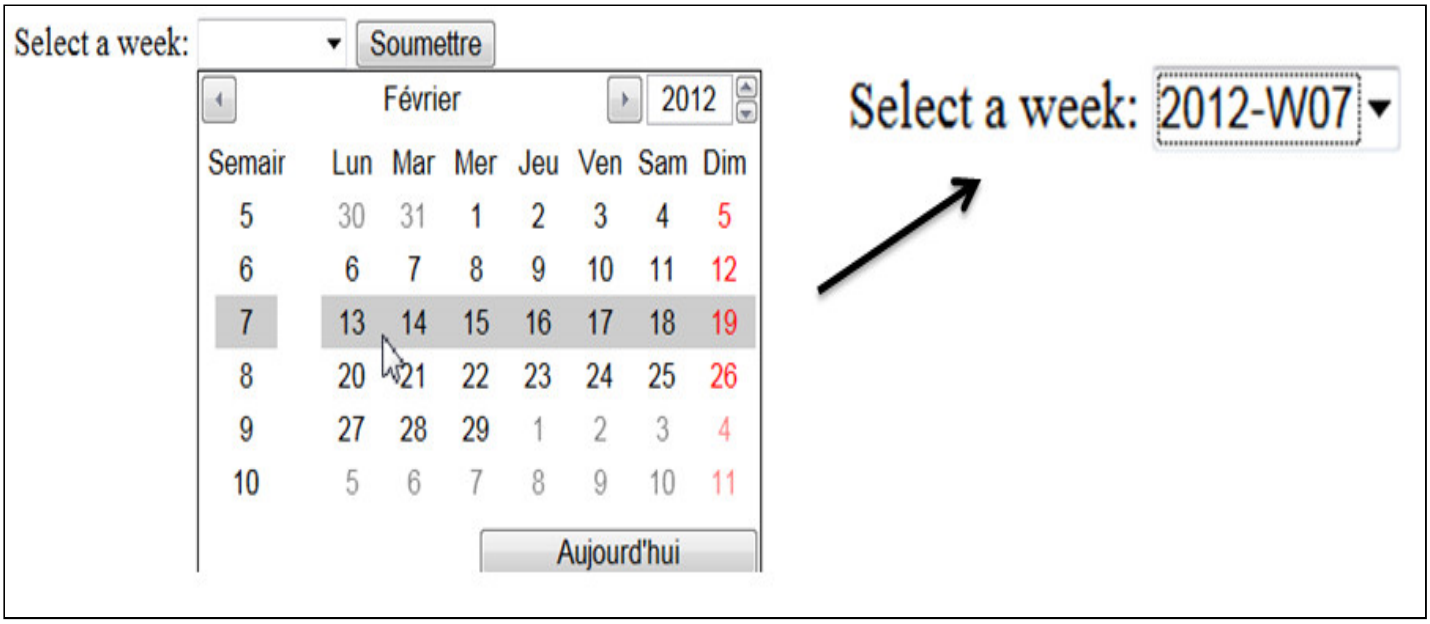




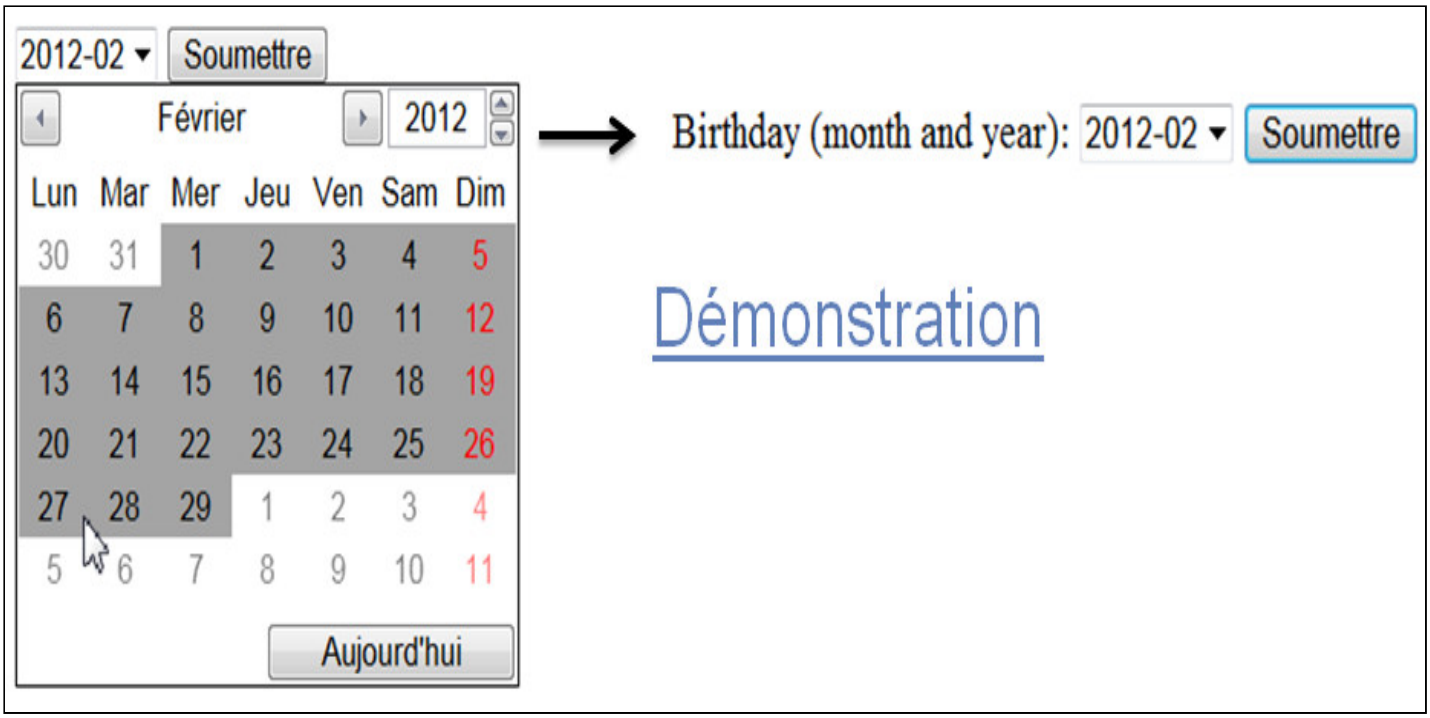

```
<input type="datetime-local">
```




```
<input type="week">:
```



`<input type="month">`:



CURRENT SUPPORT

These new input types are not yet supported by all browsers in their desktop versions (while mobile support is 100% for major browsers).

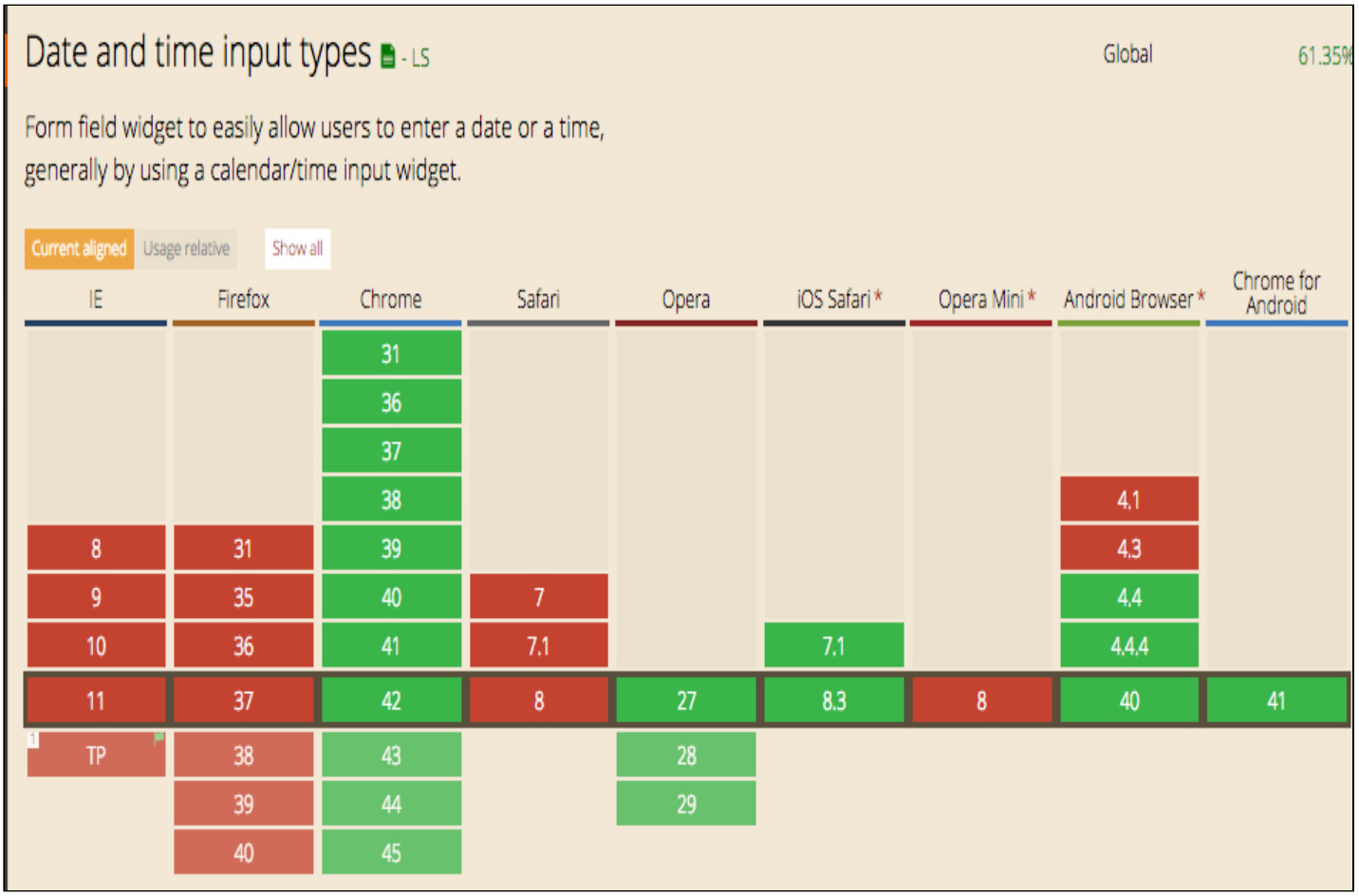
Managing dates in different languages/formats is a difficult task that browser developers have had to address. Furthermore, Web designers do not like the fact that you cannot really control the look and feel of this element (the same criticism as for the `<input type="color">` element, see previous section). These are undoubtedly the reasons why browser implementers did not make the implementation of these widgets high priority.

However, for simple date picking (especially on mobiles), these elements are very practical as is, and many polyfills are available, bringing compatibility to browsers that do not yet support it.

There is also a growing set of Web Components for picking date/time. For example,

see http://customelements.io/ and enter the "date" keyword in the search field.

So far, as at early 2015, Safari, Internet Explorer and FireFox desktop browsers still do not support this input type, as shown in the table below:

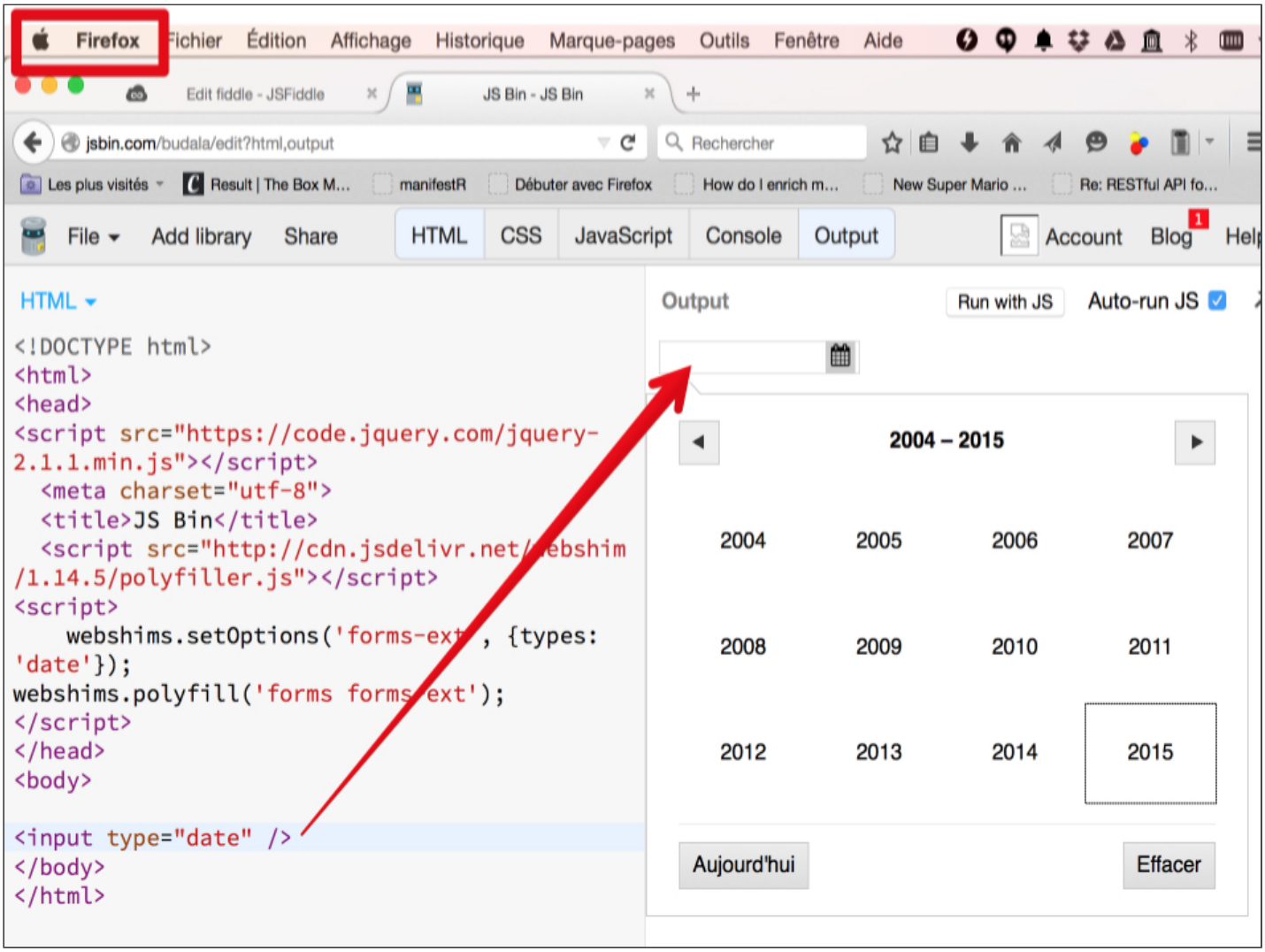


See the up-to-date version of the above table and check for the mobile support.

POLYFILLS / ALTERNATIVE SOLUTIONS

Many polyfills are available. Remember that including a polyfill means that if the target browser supports the native implementation, the polyfill will be ignored and the native implementation will be used instead. Polyfills use alternative JavaScript based widgets only as a fallback.

- Here is a running example on JS Bin, that works with all browsers (including IE and FireFox), easy to reproduce, that uses Webshim and jQuery:



Other popular polyfills are:

- https://css-tricks.com/progressively-enhancing-html5-forms/, a solution that uses Yepnope, Modernizer and the JQuery UI,
- Web Experience Toolkit date polyfill, open sourced by the Canadian government,
- Jon Stipe's polyfill,
- Others are available, just use your favorite search engine :-)

KNOWLEDGE CHECK 5.4.3 (NOT GRADED)

Which attributes are useful to constrain the user to choose a specific day in the week,

such as Monday, Tuesday, etc.?

- [ ] min
- [ ] max
- [ ] step
- [ ] only
- [ ] value

Note: Make sure you select all of the correct options - two answers are correct.