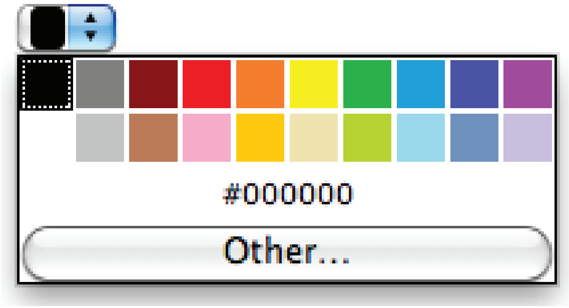
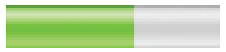
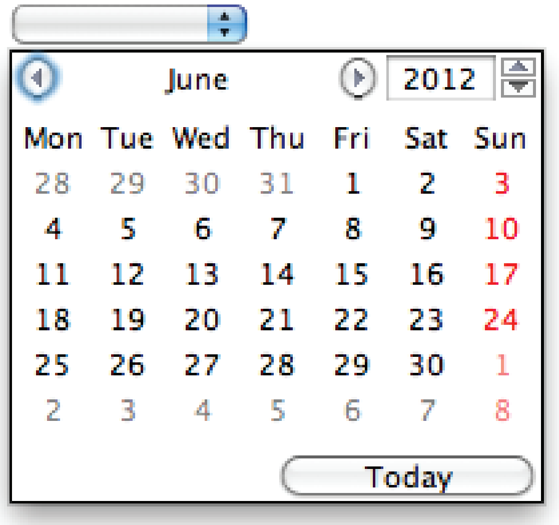
# HTML5 FORMS 2.0







| Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

# HTML5 FORMS 2.0, new tags

## OUTPUT:

Use it to display the result of a calculation:

```
<input type="number" name="one" value="60">+
<input type="number" name="two" value="40">=
<output value="100" for="one two">100</output>
```

## METER:

Dislpays a meter bar, you can set low, high, optimum, min, max values:

```
<meter low="20" optimum="60" high="90"
      min="0" max="100" value="60"></meter>
```

## PROGRESS:

Displays a progress bar to highlight status/progress changes:

```
<progress id="prBar" min="" max="" value="" ></progress>
<script>
   var bar = document.querySelector('#prBar');
   var val = 0;
function bindProgress(){
   bar.max = 1000;
   bar.min = 0;
   var timer = setInterval(function(){updateBar()}, 1000);
}

function updateBar(){
   val += (100);
   bar.value = val;
}
</script>
```

## PLACEHOLDER, I:

Displays some text (indications, suggestions) inside input fields. It disappear on focus and appears back on blur if no change has been made.

```
<input type="password" name="psw" placeholder="insert your password">
CSS ::-webkit-input-placeholder{color:green;}

Modernizr detection:

if ( Modernizr.input.placeholder ) {
    // browser supports placeholder
}
```

## PLACEHOLDER, II:

**jQuery polyfill:**

```
<input type="text" id="date-input" placeholder="12/13/2011">
var dateInput = $('#date-input');
var dateDefault = dateInput.val();

dateInput.on(
    {
      focus: function() {
            var $this = $(this);
            // If this value of the input equals our sample,
            // hide it when the user clicks on it.
            $this.val() === dateDefault && $this.val('');
      },

      blur: function() {
            var $this = $(this);
            // When they click off of the input, if
            // the value is blank, bring back the sample.
            $this.val() === '' && $this.val(dateDefault);
      }
     }
);
```

iv

# HTML5 FORMS 2.0, new attributes

## REQUIRED:

```
<input type="text" required name="username">
```
[it will prevent submission if empty]

## AUTOFOCUS:

```
<textarea autofocus></textarea>
```
[it will set the focus on that field]

## PATTERN:

```
<input type="text" required pattern="\w{6,15}"> (6 to 15 chars)
```
[provide a regExp to check the value, it only works with required]

Find patterns ready to use at http://html5pattern.com/

v

# HTML5 FORMS 2.0, new attributes

## MAXLENGTH:

```
<input type="text" maxlength="15">
```

**[it will set a limit to the length of that value.]**

```
There is no minlength attribute. (use pattern instead)
```

# HTML5 FORMS 2.0, new input types - 1

## NEW INPUT TYPES, I:

```
*email        =>  email validation
*range        =>  native sliders
<input type="range" min=0 max=10 step=1 value=6>

*color        =>  colorpickers [opera only]
*number       =>  only digits, [with arrows to change]
*date         =>  calendar     [opera only]
<input type="date" min="2000-01-10" max="2012-12-31"
[step="7" (for 7 days only)]>
```

# HTML5 FORMS 2.0, new input types - 2

## NEW INPUT TYPES, II:

```
*datetime      =>  yyyy-mm-dd HH:MM
*month         =>  yyyy-mm
*week          =>  yyyy-mmW
*time          =>  HH:MM
```

{polyfill: jQuery ui DATEPICKER}

```
Modernizr.load(
     {
          test: Modernizr.inputtypes && Modernizr.inputtypes.date,
               nope: [
                    'ui/js/jquery-ui-custom.min.js',
                    'ui/css/ui-lightness/jquery-ui-custom.css',
                    'datepicker.js' // your script
               ] // end nope
     } // end loaded obj
);  // end load()
```

```
*url           =>  web addresses
*tel           =>  telephones
*search            =>  search inputs
<input type="search" results="5">
```

# HTML5 FORMS 2.0, constraint validation API

## Validity AND ValidityState:

*read API documentation at:
http://www.w3.org/html/wg/drafts/html/master/forms.html#the-constraint-validation-api

*use ONINVALID and ONINPUT events

*use **e.target.setCustomValidity**(msg) to set message

*the validity attribute (**e.target.validity**) must return a
**validityState** object that represents validity states of the element

```
interface ValidityState {
  readonly attribute boolean valueMissing;
  readonly attribute boolean typeMismatch;
  readonly attribute boolean patternMismatch;
  readonly attribute boolean tooLong;
  readonly attribute boolean rangeUnderflow;
  readonly attribute boolean rangeOverflow;
  readonly attribute boolean stepMismatch;
  readonly attribute boolean badInput;
  readonly attribute boolean customError;
  readonly attribute boolean valid;
};
```

```
input.oninvalid =
function(e){
  e.target.setCustomValidity("");
  if(e.target.validity.valueMissing){
    console.log(e.target.validity);
    e.target.setCustomValidity('missing');
  }//end if

  else if(e.target.validity.typeMismatch){
    e.target.setCustomValidity("");
    e.target.setCustomValidity('wrong type');
  }//end if
}//end oninvalid
```