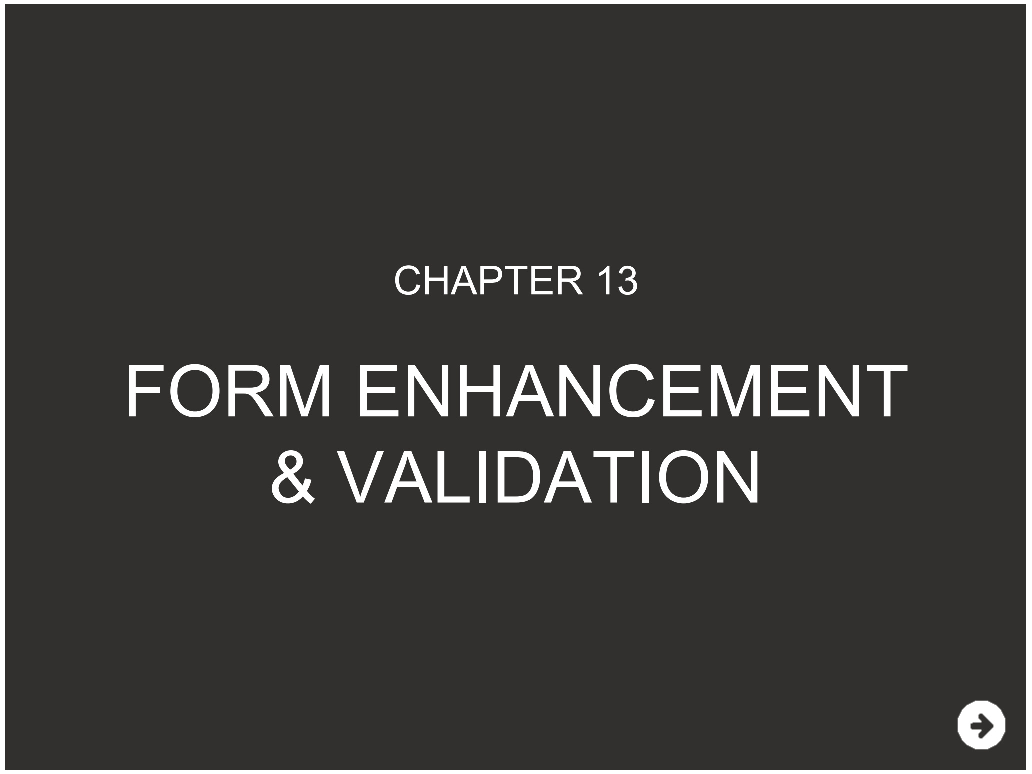
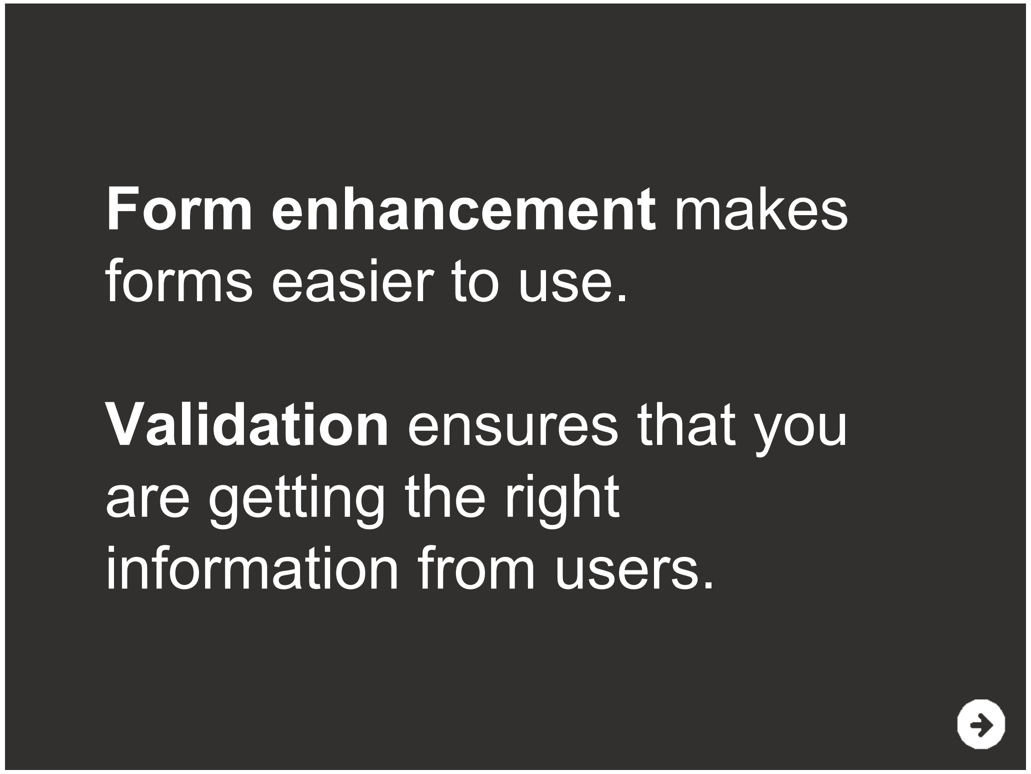
JAVASCRIPT & JQUERY

interactive front-end web development



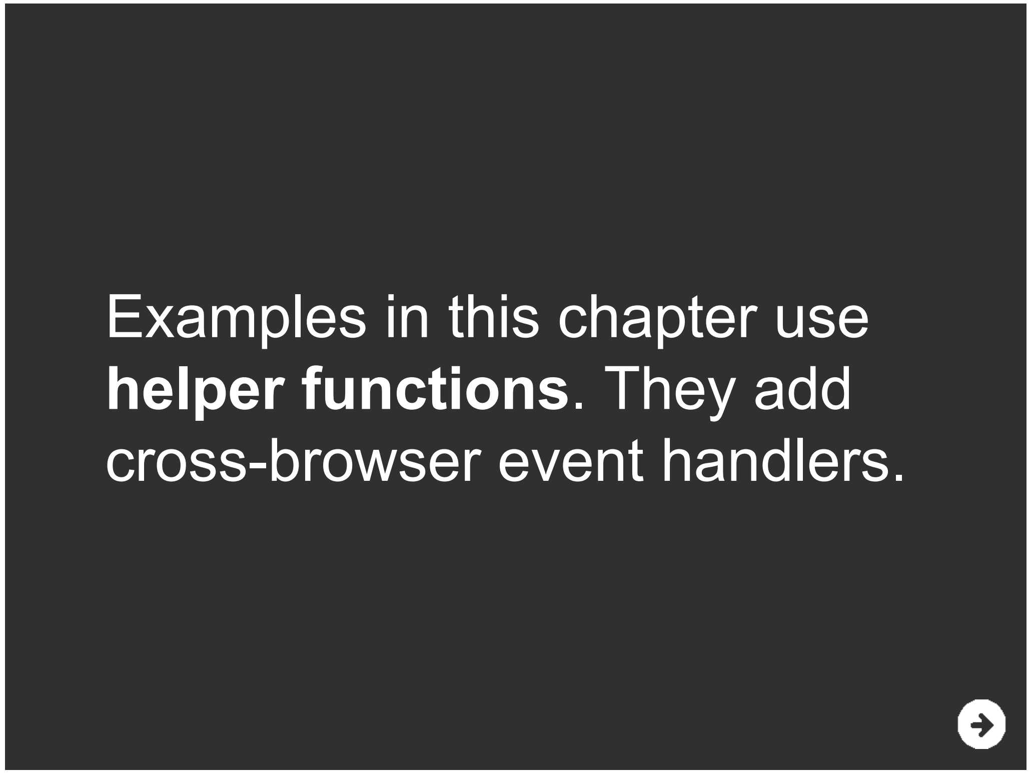
CHAPTER 13

FORM ENHANCEMENT & VALIDATION



Form enhancement makes forms easier to use.

Validation ensures that you are getting the right information from users.



Examples in this chapter use **helper functions**. They add cross-browser event handlers.

Helper function to add events:

```
function addEvent(el, event, callback) {
  // If addEventListener works use it
  if ('addEventListener' in el) {
    el.addEventListener(event, callback, false);
  } else {
    // Otherwise create IE fallback
    el['e' + event + callback] = callback;
    el[event + callback] = function () {
      el['e' + event + callback](window.event);
    };
    el.attachEvent('on' + event, el[event + callback]);
  }
}
```

DOM nodes for form controls have different properties and methods than other elements.

`<form>` ELEMENT

properties	**methods**	**events**
`action`	`submit()`	`submit`
`method`	`reset()`	`reset`
`name`		
`elements`		

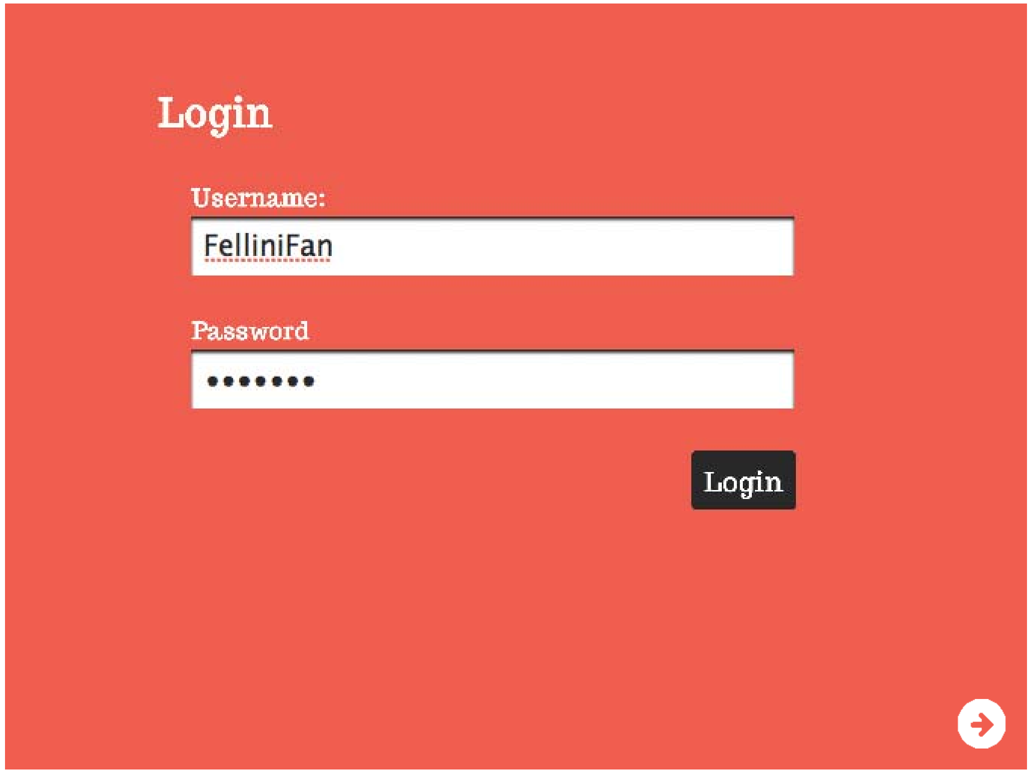
FORM CONTROLS

properties	**methods**	**events**
`value`	`focus()`	`blur`
`type`	`blur()`	`focus`
`name`	`select()`	`click`
`disabled`	`click()`	`change`
`checked`		`input`
`selected`		`keyup`
`form`		`keydown`
`defaultChecked`		`keypress`

WORKING WITH FORMS

SUBMITTING FORMS

To work with a form’s content, use the `preventDefault()` method of the `event` object to stop it from being sent.



Login

Username:
FelliniFan

Password
•••••••

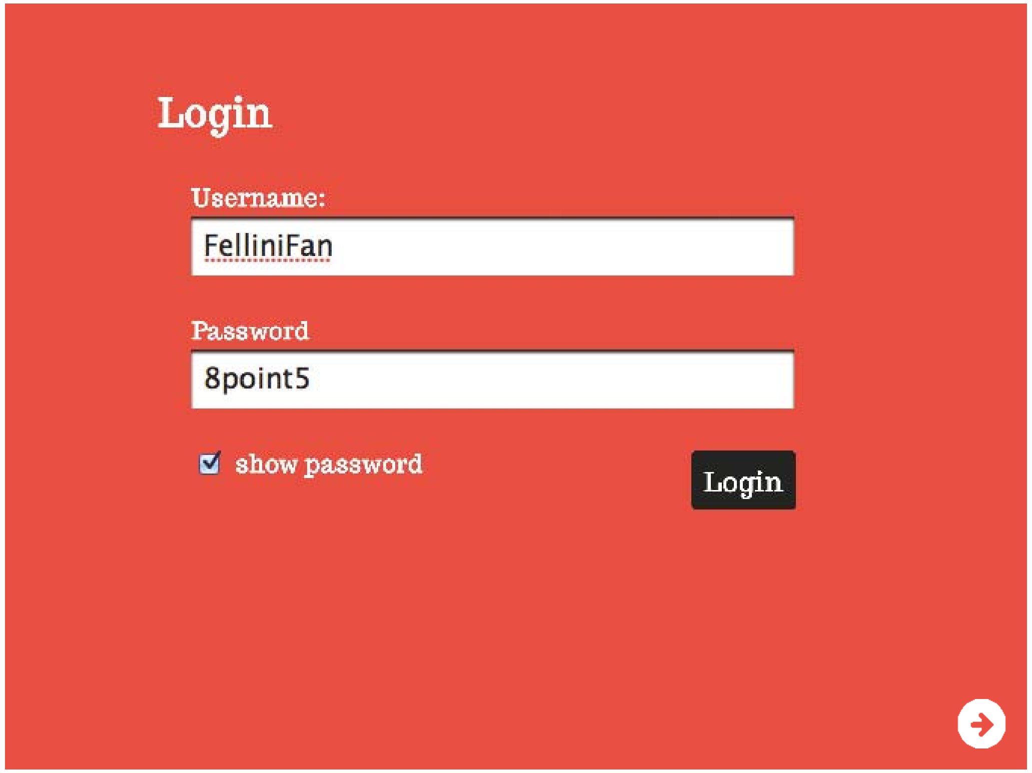
Submitting a form:

```
(function() {
  var form = document.getElementById('login'); // Form element

  addEvent(form, 'submit', function(e) { // On submit
    e.preventDefault();                  // Stop it being sent
    var elements = this.elements;        // Get form elements
    var username = elements.username.value; // Get username
    var msg      = 'Welcome ' + username;   // Welcome message

    // Write welcome message
    document.getElementById('main').textContent = msg;
  });
}());
```

TYPE OF INPUT

The `type` property of an input corresponds with the `type` attribute in HTML.

(It won't work in IE8 or earlier.)



Login

Username:

FelliniFan

Password

8point5

☑ show password

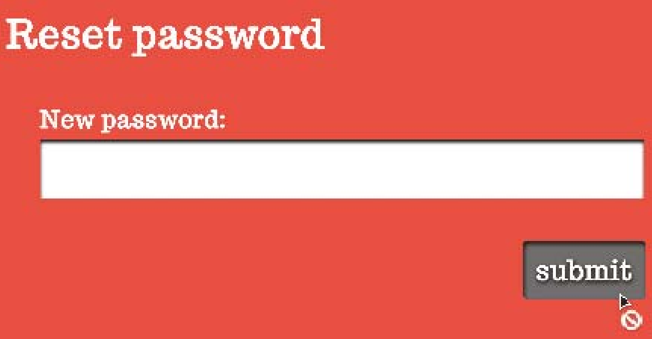
Login

Showing a password:

```
var pwd = document.getElementById('pwd');      // Get pwd input
var chk = document.getElementById('showPwd'); // Get checkbox

addEvent(chk, 'change', function(e) {     // Click on checkbox
  var target = e.target || e.srcElement; // Get that element
  try {                                   // Try following code
    if (target.checked) {                 // If checked set
      pwd.type = 'text';                  // type to text
    } else {                              // Otherwise set
      pwd.type = 'password';              // type to password
    }
  } catch(error) {                        // If an error
    alert('This browser cannot switch type'); // Show warning
  }
});
```

DISABLE INPUTS

The `disabled` property of an input corresponds with the `disabled` attribute in HTML.



Reset password

New password:

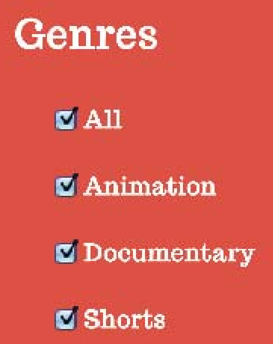
submit

Disabling a submit button:

```
var form      = document.getElementById(‘newPwd'); // Form
var password  = document.getElementById('pwd');     // Password
var submit    = document.getElementById('submit'); // Submit

var submitted = false;              // Has form been submitted?
submit.disabled = true;             // Disable submit button
submit.className = 'disabled';      // Style submit button
```

CHECKBOXES

The `checked` property of an input corresponds with the `checked` attribute in HTML.



Genres

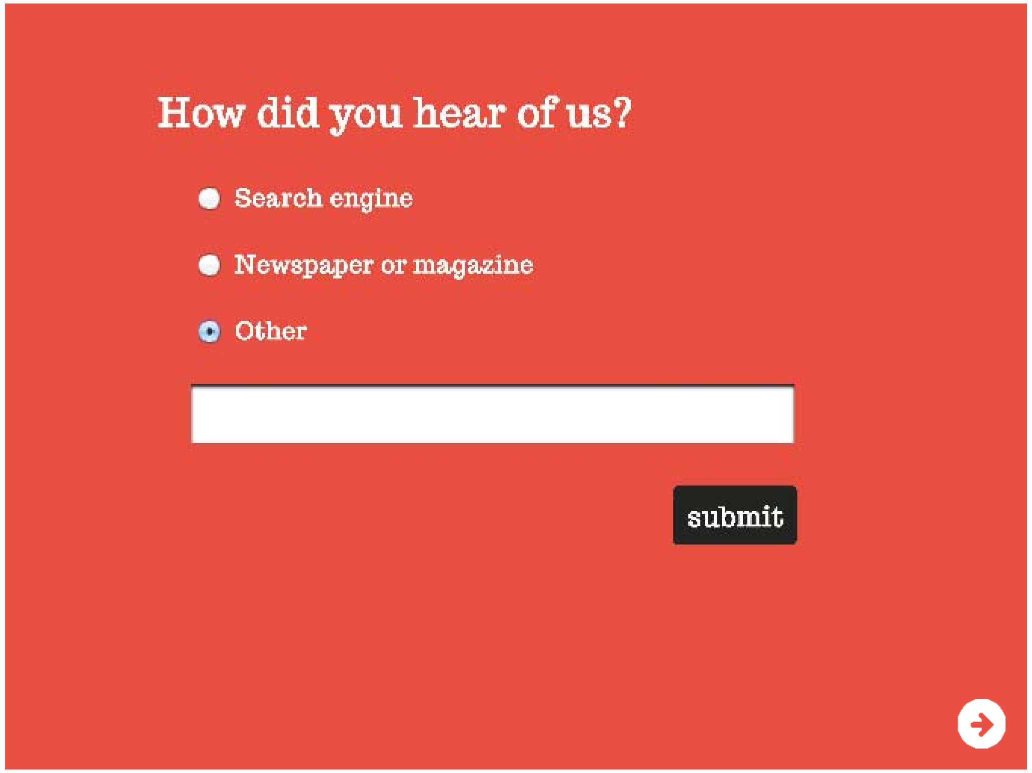
- ☑ All
- ☑ Animation
- ☑ Documentary
- ☑ Shorts

Selecting all checkboxes:

```
var form = document.getElementById('interests'); // Form
var elements = form.elements;                  // Elements in form
var options  = elements.genre;                 // Genre checkboxes
var all = document.getElementById(‘all');     // 'All' checkbox

function updateAll() {
  for (var i = 0; i < options.length; i++) { // Each checkbox
    options[i].checked = all.checked;         // Update property
  }
}
addEvent(all, 'change', updateAll);           // Event listener
```

RADIO BUTTONS

The `checked` property is also commonly used with radio buttons.



How did you hear of us?

- Search engine
- Newspaper or magazine
- [✓ Other]

submit

Showing a text input:

```
options   = form.elements.heard;                // Radio
buttons
other     = document.getElementById('other');   // Other button
otherText = document.getElementById('other-text'); // Other
text otherText.className = 'hide';                   // Hide
other

for (var i = [0]; i < options.length; i++) {    // Each option
 addEvent(options[i], 'click', radioChanged);   // Add listener
}

function radioChanged() {
 hide = other.checked ? '' : 'hide';    // Is other checked?
 otherText.className = hide;            // Text input
visibility
 if (hide) {                            // If text input hidden
  otherText.value = '';                 // Empty its contents
 }
}
```

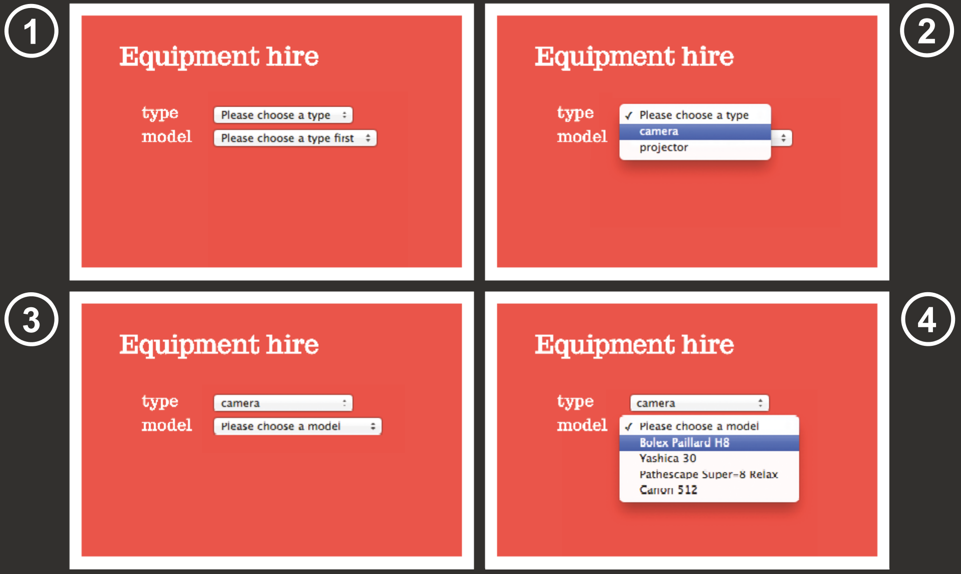
SELECT BOXES

Select boxes have more properties and methods than other form controls.

The `<option>` elements hold the values select boxes contain.

SELECT BOXES

properties	methods
`options`	`add()`
`selectedIndex`	`remove()`
`length`	
`multiple`	
`selectedOptions`	



Info for select boxes is stored in objects:

```
// Type select box
var type  = document.getElementById(‘equipmentType');

// Model select box
var model = document.getElementById('model');
var cameras = {                         // Object stores cameras
    bolex: 'Bolex Paillard H8',
    yashica: 'Yashica 30',
    pathescape: 'Pathescape Super-8 Relax',
    canon: 'Canon 512'
};
var projectors = {                      // Store projectors
    kodak: 'Kodak Instamatic M55',
    bolex: 'Bolex Sound 715',
    eumig: 'Eumig Mark S',
    sankyo: 'Sankyo Dualux'
};
```

Getting the right object:

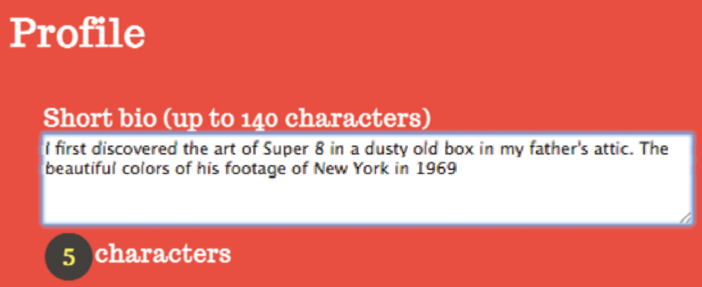
```
function getModels(equipmentType) {
  // If type is cameras return cameras object
  if (equipmentType === 'cameras') {
    return cameras;
  // If type is projectors return projectors object
  } else if (equipmentType === 'projectors') {
    return projectors;
  }
}
```

Populating select boxes:

```
addEvent(type, 'change', function() {  // Change select box

  if (this.value === 'choose') {        // No selection made
    model.innerHTML = '<option>Please choose a type
first</option>';
    return;                             // No need to proceed
  }
  var models = getModels(this.value);  // Get right object

  var options = '<option>Please choose a model</option>';
  var key;
  for (key in models) {                 // Loop through models
    options += '<option value="' + key + '">' + models[key]
             + '</option>';
  }
  model.innerHTML = options;            // Update select box

});
```

TEXTAREA

The `value` property gets and updates the value entered into a textarea or text input.



Profile

Short bio (up to 140 characters)

I first discovered the art of Super 8 in a dusty old box in my father's attic. The beautiful colors of his footage of New York in 1969

5 characters

Set-up and event handling:

```
(function () {
  var bio      = document.getElementById('bio');
  var bioCount = document.getElementById('bio-count');

  // Call updateCounter() on focus or input events
  addEvent(bio, 'focus', updateCounter);
  addEvent(bio, 'input', updateCounter);

  addEvent(bio, 'blur', function () {    // Leaving the element
    if (bio.value.length <= 140) {       // If bio not too long
      bioCount.className = 'hide';       // Hide the counter
    }
  });
```

Updating the counter:

```
function updateCounter(e) {
  var target = e.target || e.srcElement; // Get target of
event
  var count = 140 - target.value.length; // Characters left
  if (count < 0) {                       // Less than 0 chars
    bioCount.className = 'error';        // Add class of error
  } else if (count <= 15) {              // Less than 15
chars?
    bioCount.className = 'warn';         // Add class of warn
  } else {                               // Otherwise
    bioCount.className = 'good';         // Add class of good
  }
  var charMsg = '<b>' + count + '</b>' + ' characters'; // Msg
  bioCount.innerHTML = charMsg;          // Update counter
}
```

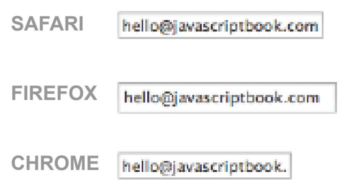
HTML5 ELEMENTS & ATTRIBUTES

HTML5 added form elements and attributes that perform tasks that had previously been done by JavaScript.

In particular, the elements can check that the user entered the right kind of information.

If not, they show an error. This is known as **validation**.

EMAIL, PHONE & URL

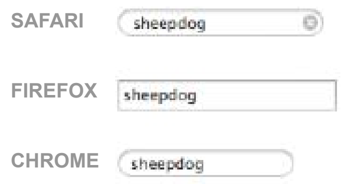

SAFARI hello@javascriptbook.com

FIREFOX hello@javascriptbook.com

CHROME hello@javascriptbook.

```
<input type="email">
<input type="url">
<input
type="telephone">
```

SEARCH

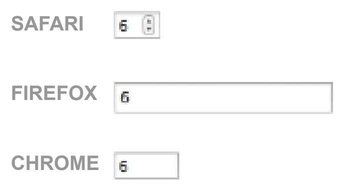

SAFARI sheepdog

FIREFOX sheepdog

CHROME sheepdog

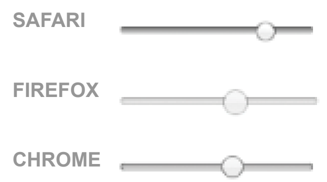
```
<input type="search"
       placeholder="search"
       autofocus>
```

NUMBER



```
<input type=“number”
       min=“0”
       max=“10”
       step=“2”
       value=“6”>
```

RANGE



```
<input type=“range”
       min=“0”
       max=“10”
       step=“2”
       value=“6”>
```

COLOR PICKER

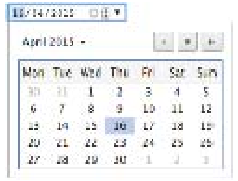
(Chrome & Opera only)

CHROME

```
<input type="color">
```

DATE

CHROME



16/04/2015

April 2015

Mon	Tue	Wed	Thu	Fri	Sat	Sun
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

```
<input type="date">
<input type="month">
<input type="week">
<input type="time">
<input type="datetime">
```

HTML5 elements are not supported in all desktop browsers. (There is much better support on mobile.)

When they are supported, they can look very different.

To get around this lack of support, you can use polyfills or feature detection.

FORM VALIDATION

Form validation checks that users enter data in the right format. If not, an error message is shown to users.

Generic checks are the kind that would be performed on different kinds of form.

If it uses the `required` attribute, does it have a value?
Does the value match what is indicated by the `type` attribute?

Custom validation tasks correspond to specific requirements of a given form.

If the user's bio less than 140 characters?
If the user is under 13, is the parental consent checkbox selected?

Check every element before submitting the form so you can show all errors at once.

You can create an object to keep track of each element and whether its entry is valid.

To check if you can submit the form, check the `valid` object.

The last example in the book uses JavaScript for validation and HTML5 validation as a fallback. This gives maximum visual consistency, and browser compatibility.

Checking if a required input has a value uses three functions:

```
function validateRequired(el) {
  if (isRequired(el)) {
    var valid = !isEmpty(el);
    if (!valid) {
      setErrorMessage(el, 'Field required');
    }
    return valid;
  }
  return true;
}
```

The `isRequired()` function checks if it has the `required` attribute:

```
function  isRequired(el) {
  return ((typeof el.required === 'boolean') && el.required)
          ||(typeof el.required ==='string');
}
```

The `isEmpty()` function checks if the element is empty:

```
function  isEmpty(el) {
  return !el.value || el.value === el.placeholder;
}
```

Error messages can be stored with the element using jQuery's `data()` method:

```
function setErrorMessage(el, message) {
  $(el).data('errorMessage', message);
}

function showErrorMessage(el) {
  var $el = $(el);
  var $errorContainer = $el.siblings('.error');
  if (!$errorContainer.length) {
    $errorContainer = $('<span class="error">
                        </span>').insertAfter($el);
  }
}
```

The type of content in a text input is validated using the `validateTypes()` function.

In turn, this function uses an object called `validateType` which has three methods to validate email addresses, numbers, and dates.

The `validateType` object uses regular expressions:

```
var validateType = {
  email: function(el) {
    var valid = /[^@]+@[^@]+/.test(el.value);
    if (!valid) {
      setErrorMessage(el, 'Please enter a valid email');
    }
    return valid;
  },
  number: function(el) {
    // Check is a number
  },
  date: function(el) {
    // Check date format
  }
}
```

The `validateType` object is used by the `validateTypes()` function:

```
function validateTypes(el) {
  if (!el.value) return true;

  var type = $(el).data('type') || el.getAttribute('type');

  if (typeof validateType[type] === 'function') {
    return valiudateType[type](el);
  } else {
    return true;
  }

}
```

Regular expressions search for characters that form a pattern. They can also replace those characters with new ones or simply remove them.

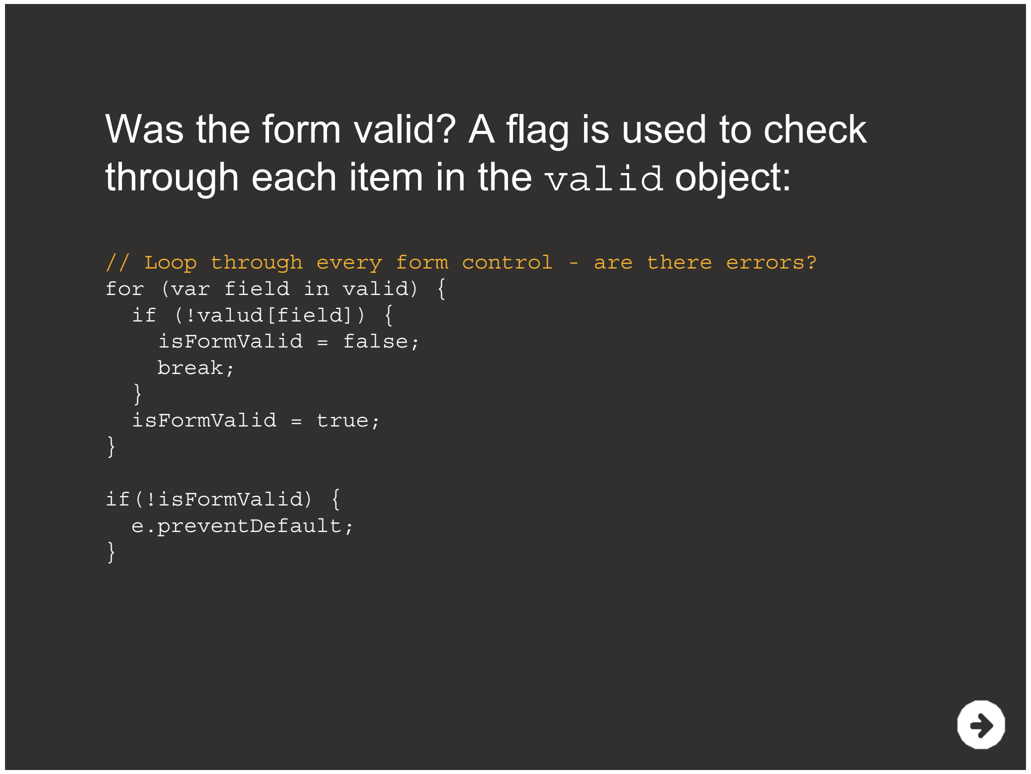
`.`	single character (except newline)
`[ ]`	single character in brackets
`[^ ]`	single character not in brackets
`\d`	digit
`\D`	non-digit character
`\w`	alphanumeric character
`\W`	non-alphanumeric character
`^`	the starting position in any line
`$`	ending position in any line
`*`	preceding element 0 or more times

Checking the length of the bio is an example of custom validation:

```
function validateBio() {
  var bio = document.getElementById('bio');
  var valid = bio.value.length <= 140;
  if (!valid) {
    setErrorMessage(bio, 'Bio should not exceed 140 chars);
  }
  return valid;
}
```

VALIDATION EXAMPLE OVERVIEW

A: Set up script
B: Perform generic checks
C: Perform custom validation
D: Did it pass validation?



Was the form valid? A flag is used to check through each item in the `valid` object:

```
// Loop through every form control - are there errors?
for (var field in valid) {
  if (!valud[field]) {
    isFormValid = false;
    break;
  }
  isFormValid = true;
}

if(!isFormValid) {
  e.preventDefault;
}
```