FORMS

FORMS

You can collect information from users to use in your code. The most common method is an HTML form

```
<form id="userForm">
  <label for="name">First Name:</label>
  <input type="text" id="firstName"/>
  <input type="radio" name="married" value="Yes" checked /> Yes
  <input type="radio" name="married" value="No" /> No
  <input type="submit" id="submitBtn" value="Submit" />
</form>
```

TEXT INPUT ELEMENTS

Text Box

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

Password Box

```
<input type="password">
```

Text Area

```
<textarea></textarea>
```

TICK BOX ELEMENTS

Checkbox

```
<input type="checkbox" />
```

Radio button

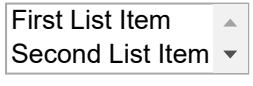
```
<input type="radio" />
```

SELECT ELEMENTS

Drop Down List


```
<select>
  <option>Option #1</option>
  <option>Option #2</option>
  <option>Option #3</option>
</select>
```

List Box




```
<select size="2">
  <option>First List Item</option>
  <option>Second List Item</option>
  <option>Third List Item</option>
  <option>Fourth List Item</option>
</select>
```

BUTTONS

Standard Button

Click Me

```
<input type="button" value="Click Me" />
```

Submit Button

Submit

```
<input type="submit" />
```

Reset Button

Reset

```
<input type="reset" />
```

ACTIVITY: CREATE A FORM

- In an index.html page, create a form with several different inputs.
- Try a standard newsletter form.

ACCESS FORM DATA

Standard Way

```
var myForm = document.getElementById('myForm');
```

Forms Collection

```
var formsList = document.forms;
var firstForm = document.forms[0];
```

```
var myForm = document.myForm;
```

GETTING INPUTS: `name` PROPERTY

You can access a particular input by name.

```
<form action="" name="myForm">
    <input type="email" name="email">

    <input type="text" name="first name">
</form>
```

```
var email = document.myForm.email;

var firstName = document.myForm["first name"];
```

value PROPERTY

```
<form action="" name="myForm">
    <input type="email" name="email">

    <input type="text" name="first name">
</form>
```

Get the information a user writes in the input.

```
var email = document.myForm.email.value;

var firstName = document.myForm["first name"].value;
```

type PROPERTY

```
<form action="" name="myForm">
    <input type="email" name="email">

    <input type="text" name="first name">
</form>
```

You can get the type of the input.

```
document.myForm.email.type; // email
```

`focus()` AND `blur()`

```
<form action="" name="myForm">
    <input type="email" name="email">

    <input type="text" name="first name">
</form>
```

Focus a particular input.

```
document.myForm.email.focus();
```

Unfocus an element.

```
document.myForm.email.blur();
```

CHECKBOXES

Choose some features to create a monster

- [x] Scales
- [ ] Horns
- [ ] Claws

```
<fieldset>
    <legend>Choose some features to create a monster</legend>

    <div>
        <input type="checkbox" id="scales" name="feature" value="scales" checked />
        <label for="scales">Scales</label>
    </div>

    <div>
        <input type="checkbox" id="horns" name="feature" value="horns" />
        <label for="horns">Horns</label>
    </div>

    <div>
        <input type="checkbox" id="claws" name="feature" value="claws" />
        <label for="claws">Claws</label>
    </div>

</fieldset>
```

GETTING CHECKBOX VALUES

Choose some features to create a monster

- [x] Scales
- [ ] Horns
- [ ] Claws

```
// Create a list of checkboxes by using the checkbox name
var checkboxList = document.form2.feature;

// Loop through your checkbox list
for (var i = 0; i < checkboxList.length; i++) {

  // If a checkbox is checked, console.log the value of the checkbox.
  if (checkboxList[i].checked) {
      console.log(checkboxList[i].value);
  }
}
```

SUBMIT BUTTONS - PREVENT DEFAULT

form.html

```
<form action="" name="myForm">
  <!-- other inputs ..... -->

  <button name="submit" type="submit">Submit</button>
</form>
```

.js file

```
// Assign your submit button to a variable
var submitButton = document.myForm.submit;

// Add a "click" event listener to your submitButton
submitButton.addEventListener("click", function(event) {

  // Prevent the default action
  event.preventDefault();

  // Run the rest of your code
  var name = document.myForm["first name"].value;
  console.log(name);
})
```

ACTIVITY: COLLECT A VALUE

- Create a simple HTML page with a form.
- Use the submit button to collect a value from an input element on the page.
- Use this value inside a function of some kind.

For example, collect a number and multiply it by five or collect a name and display a greeting.

VALIDATING FORMS

If your user enters unexpected information (or no information) into your forms, you want to let them know.

Examples:

- "This field is required."
- "Please enter your phone number in the format xxx-xxxx."
- "Please enter a valid email address."
- "Your password needs to be longer than 5 characters."

CLIENT-SIDE VS SERVER-SIDE VALIDATION

- Client-side validation occurs in the browser before the data has been submitted.
  - User-friendly
  - Faster response
  - JavaScript validation and HTML5 form validation
- Server side validation occurs on the server after data has been submitted.

BUILT-IN FORM VALIDATION

HTML5 allows us to validate forms without a lot of JavaScript.

- is the value required?
- is the length of data between the minimum and maximum length?
- is the data a number?
- is this a valid email address?

REQUIRED

Would you prefer a banana or cherry? Submit

form.html

```
<form>
  <label for="choose">Would you prefer a banana or cherry?</label>
  <input id="choose" name="i_like" required>
  <button>Submit</button>
</form>
```

styles.css

```
input:invalid {
  border: 2px dashed red;
}

input:valid {
  border: 2px solid black;
}
```

PATTERN MATCHING

Would you prefer a banana or cherry? Submit

```
<!-- Use the pattern attribute to require a specific input pattern. -->

<form>
  <label for="choose">Would you prefer a banana or cherry?</label>
  <input id="choose" name="i_like" pattern="banana|cherry" required>
  <button>Submit</button>
</form>
```

MIN AND MAX

Would you prefer a banana or a cherry?

How many would you like? 1

Submit

```
<form>
    <div>
        <label for="choose">Would you prefer a banana or a cherry?</label>
        <input id="choose" name="i_like" required minlength="6" maxlength="6">
    </div>
    <div>
        <label for="number">How many would you like?</label>
        <input type="number" id="number" name="amount" value="1" min="1" max="10">
    </div>
    <div>
        <button>Submit</button>
    </div>
</form>
```

EMAIL VALIDATION

Submit

```
<input type="email" required>
```

email@sait.ca

Submit

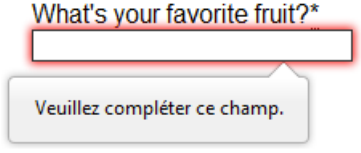
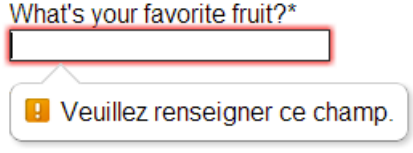
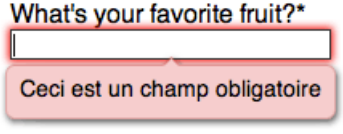
```
<input type="email" placeholder="email@sait.ca" pattern=".+@sait.ca" size="30"
required title="Must be a sait.ca email address" />
```

ACTIVITY: VALIDATE A FORM

- Create a simple index.html page.
- Add a form with a few questions.
- Make everything required.
- Add validation to text boxes.

CUSTOMIZED ERROR MESSAGES

French versions of feedback messages on an English page

Browser	Rendering
Firefox 17 (Windows 7)	
Chrome 22 (Windows 7)	
Opera 12.10 (Mac OSX)	

CREATE YOUR FORM

Please enter the following details:

Name:

Age:

Submit

```
<form action="" name="form1">
  Please enter the following details:
  <p>
    Name: <input type="text" id="name" name="txtName" />
  </p>
  <p>
    Age: <input type="text" id="txtAge" name="txtAge" />
  </p>
  <p>
    <input type="submit" id="submitBtn" value="Submit" name="btnCheckForm">
  </p>
</form>
```

ADD ERROR MESSAGES

At the bottom of your current form (before the closing `</form>` tag), add 2 error messages.

```
<form action="" name="form1">

  <!-- Form Inputs -->

  <p id="errorName" style="display:none;">Please enter your name.</p>
  <p id="errorAge" style="display:none;">Please enter your age.</p>
</form>
```

WRITE JAVASCRIPT

In your .js file, create variables for the 2 error messages and the submit button.

```
var submitButton = document.getElementById('submitBtn');
var errorName = document.getElementById('errorName');
var errorAge = document.getElementById('errorAge');
```

ADD AN EVENT LISTENER

Now, we'll add a "click" event listener to execute code when the submitButton is clicked.

```
submitButton.addEventListener("click", function(event) {
  // Code goes here
})
```

GET THE INPUT VALUES

```
submitButton.addEventListener("click"), function(event) {
  var name = document.form1.name.value;
  var age = document.form.txtAge.value;
}
```

VALIDATE THE NAME INPUT

```
submitButton.addEventListener("click"), function(event) {
  var name = document.form1.name.value;
  var age = document.form.txtAge.value;

  if (!name) {
    event.preventDefault();
    errorName.style.display = "block";
  }
}
```

VALIDATE AGE

```
submitButton.addEventListener("click"), function(event) {
  var name = document.form1.name.value;
  var age = document.form.txtAge.value;

  if (!name) {
    event.preventDefault();
    errorName.style.display = "block";
  }

  if (!age) {
    event.preventDefault();
    errorAge.style.display = "block";
  }

}
```

RESET ERROR MESSAGES

```
submitButton.addEventListener("click"), function(event) {
  errorName.style.display = "none";
  errorAge.style.display = "none";

  var name = document.form1.name.value;
  var age = document.form.txtAge.value;

  if (!name) {
      event.preventDefault();
      errorName.style.display = "block";
  }

  if (!age) {
      event.preventDefault();
      errorAge.style.display = "block";
  }

}
```

ACTIVITY: JAVASCRIPT VALIDATION

- Add some code to check if a user's age is 18 or above and display an error message if the user is too young.
- Remember that the age entered by the user is a string.

REGULAR EXPRESSIONS

'Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.'

~ Jamie Zawinski

REGULAR EXPRESSIONS

```
<!-- Use the pattern attribute to require a specific input pattern. -->

<form>
  <label for="choose">Would you prefer a banana or cherry?</label>
  <input id="choose" name="i_like" pattern="banana|cherry" required>
  <button>Submit</button>
</form>
```

Resources:

- Page 158 in Beginning JavaScript, 5th edition
- RegexOne Tutorial
- Regex Crossword

CREATING A REGULAR EXPRESSION

```
// RegExp constructor
var regEx1 = new RegExp("abc");

// literal value
var regEx2 = /abc/;
```

Pattern: a character followed by a b followed by a c.

test METHOD

You can test a string to see if it matches a defined pattern.

```
console.log(/abc/.test("abcde")); // true

console.log(/abc/.test("abxde")); // false
```

PATTERNS

Pattern	Description
/abc/	A sequence of characters
/[abc]/	Any character from a set of characters
/[^abc]/	Any character not in a set of characters
/[0-9]/	Any character in a range of characters
/x+/	One or more occurrences of the pattern x
/x+?/	One or more occurrences, nongreedy
/x*/	Zero or more occurrences
/x?/	Zero or one occurrence
/x{2,4}/	Two to four occurrences
/(abc)/	A group
/a	b
/\d/	Any digit character
/\w/	An alphanumeric character (“word character”)
/\s/	Any whitespace character
/./	Any character except newlines
/\b/	A word boundary
/^/	Start of input
/$/	End of input

ACTIVITY: REGEXONE

- Go to https://regexone.com/ and spend some time working through the tutorial.
- Once you work through the tutorial, create some regex variables in JavaScript, and use the test method to check different strings of information.