Today's Best Practice in JavaScript

Brett Taylor

What I'll be covering

- Where we've come from with JavaScript
- What is wrong with the old ways
- How we can apply functionality to a page like we can our designs

- Goal: Javascript is easier than you think
- WARNING: LOADS OF CODE

Who is this guy?

- Geek (and proud)
- Programming since 1989 (since I was 8)
- qBasic -> Visual Basic -> HTML -> CSS -> PHP -> JavaScript
- Web Professional since 2001
- Still a hobbyist
- Amateur Internet Anthropologist?

Natcoll

- Course Co-ordinator, Wellington Campus
- Diploma of Web Development
  - 9 months long
  - XHTML, CSS, JavaScript, ActionScript, PHP, MySql, XML, RSS, XSLT, Google Maps, Web Services... among other things
  - No prior programming knowledge assumed

History of Javascript

- Bad reputation for needless effects (DHTML)
- Time consuming to install and configure
- Browser compatibility
- Conflicts when trying to run more than one script at once

What changed?

- More consistant browser support
  - Document Object Model (DOM)
- Google Maps
  - People started taking Javascript seriously
  - xmlHttpRequest (XHR)
    - aka AJAX (async javascript and xml)

Problems with Old-school Javascript

- Javascript can be put in many places
- Javascript must be applied to individual pages

Unobtrusive Javascript

- Document + Markup
- Presentation (if supported)
- Interactivity and Functionality (if supported)

Seperation

- HTML: Document + Markup
- CSS: Presentation (if supported)
- JS: Interactivity and Functionality (if supported)

How does CSS do it?

- Goodbye <font> etc.
- Hello external style sheets
- Hello class='…' and id='…'

…So with Javascript

- Goodbye onclick='…', etc.
- Hello external script files
- Hello class='…' and id='…'

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Test Page</title>
  <script type='text/javascript'>
    function validateForm() {
      // do stuff
      if (!valid) return false;
    }
  </script>

</head>

<body>
<form onsubmit="validateForm()">
   <label for='email'>Email Address</label>
   <input id='email' name='email' type='text' />
</form>
</body>
</html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Test Page</title>
  <script type='text/javascript'>
    function validateForm() {
      // do stuff
      if (!valid) return false;
    }
  </script>

</head>

<body>
<form onsubmit="validateForm()">
   <label for='email'>Email Address</label>
   <input id='email' name='email' type='text' />
</form>
</body>
</html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Test Page</title>
  <script type='text/javascript'>
   window.onload = function () {
     document.getElementById('subscribe').onsubmit = function () {
       // do stuff
       if (!valid) return false;
     }
   }
  </script>

</head>

<body>
<form id='subscribe'>
   <label for='email'>Email Address</label>
   <input id='email' name='email' type='text' />
</form>
</body>
</html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Test Page</title>
  <script src="myscripts.js" type="text/javascript"></script>

</head>

<body>
<form id='subscribe'>
   <label for='email'>Email Address</label>
   <input id='email' name='email' type='text' />
</form>
</body>
</html>
```

Javascript ‘Selectors’

- HTML: <… id=’mytag’ …>
- CSS: #mytag { … }
- JS: document.getElementById(‘mytag’).onsubmit = function() { … }

Javascript ‘Selectors’

- HTML:
- CSS: img { … }
- JS: document.getElementsByTagName (‘img’).onclick = function() { … }

Javascript ‘Selectors’

- HTML: <… class=’myclass’ …>
- CSS: .myclass { … }
- JS: getElementsByClassName(document, ‘myclass’).onclick = function() { … }
  - not built into Javascript: custom function

getElementsByClassName

```
// from http://snook.ca/archives/javascript/your_favourite_1/
function getElementsByClassName(node, classname) {
    var a = [];
    var re = new RegExp('(^|\s)'+classname+'(\s|$)');
    var els = node.getElementsByTagName("*");
    for (var i=0,j=els.length; i<j; i++)
        if (re.test(els[i].className)) a.push(els[i]);
    return a;
}

var myElements = getElementsByClassName(document);
for (var i=0; i < myElements.length; i++) {
    myElements[i].onclick = myFunction; // no brackets
}
```

JS ‘External Style Sheets’

- CSS:
  <link rel="stylesheet" href="master.css"
  type="text/css" />
- JS: <script src="scripts.js"
  type="text/javascript"></script>
- Just a text file with javascript in it.

Form Validator

- On submit, validate every form widget
- Use classes as validation rule indicators
- If things aren't valid:
  - Colour input fields and/or labels
  - Display user-friendly messages

```
<form id="signupform" name="signupform" method="post" action="">
<p>
  <label for="username">Username</label>
  <input type="text" name="username" id="username" />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="password">Password</label>
  <input type="password" name="password" id="password" />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="password2">Confirm Password</label>
  <input type="password2" name="password2" id="password2" />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="email">Email Address</label>
  <input type="email" name="email" id="email" />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <input type="submit" name="submit" id="submit" value="Submit" />
</p>
</form>
```

Standards-Compliant Form

```
<form id="signupform" name="signupform" method="post" action="">
<p>
  <label for="username">Username</label>
  <input type="text" name="username" id="username" class=”required
minchars_6” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="password">Password</label>
  <input type="password" name="password" id="password" class=”required” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="password2">Confirm Password</label>
  <input type="password2" name="password2" id="password2"
     class=”required matchfield_password” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="email">Email Address</label>
  <input type="email" name="email" id="email" class=”required validemail” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <input type="submit" name="submit" id="submit" value="Submit" />
</p>
</form>
```

Add functionality classes

```
<form id="signupform" name="signupform" method="post" action="">
<p>
  <label for="username">Username</label>
  <span class="message" id="username_message"></span>
  <input type="text" name="username" id="username" class=”required
minlength_6” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="password">Password</label>
  <span class="message" id="password_message"></span>
  <input type="password" name="password" id="password" class=”required” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="password2">Confirm Password</label>
  <span class="message" id="password2_message"></span>
  <input type="password2" name="password2" id="password2"
     class=”required matchfield_password” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <label for="email">Email Address</label>
 <span class="message" id="email_message"></span>
  <input type="email" name="email" id="email" class=”required validemail” />
  <span class='required' title='Field Required'>*</span>
</p><p>
  <input type="submit" name="submit" id="submit" value="Submit" />
</p>
</form>
```

Add message fields

What JS needs to do

- On submit
  - With every form widget
    - With each class it has
      - Run the test for each class
      - Display any errors
- If no errors form wide, allow the submit
- If errors, cancel the submit

```
var btValidator = new Object; // prevents namespace collisions

// run this function to add event to form, eg:
// btValidator.setUpEventsOnForm(document.getElementById('signupform');

btValidator.setUpEventsOnForm = function (formElement) {
      if (typeof formElement == 'string') {
         formElement = document.getElementById(formElement);
      }
      formElement.onsubmit = function () {
          return (btValidator.testForm(this));
      }
}
```

in btValidator.js

```
btValidator.testForm = function (formElement) {
  // get all form fields
  var allFields;

  var isError;
  // iterate through each field
  var result;
  for (var i = 0; i < allFields.length; i++) {
    // set result to result of tests

    // if there was an error for this field, tell the user
  }

  // return if there was an error
  return (isError); // will cancel submit if isError == false
}
```

in btValidator.js

```
btValidator.getFormElementsForForm = function (formElement) {
  var a = new Array();
  // create array of all input, select and textarea elements
  a = a.concat(this.collectsToArr(formElement.getElementsByTagName('input')));
  a = a.concat(this.collectsToArr(formElement.getElementsByTagName('select')));
  a = a.concat(this.collectsToArr(formElement.getElementsByTagName
('textarea')));
  return (a);
}

btValidator.collectsToArr = function (collection) {
      var elementArray = new Array();
      for (var i = 0; i < collection.length; i++) {
          elementArray.push(collection[i]);
      }
      return (elementArray);
}
```

in btValidator.js

```
btValidator.testForm = function (formElement) {
  // get all form fields
  var allFields = this.getFormElementsForForm(formElement);
 var isError;

 // iterate through each field
  var result;
  for (var i = 0; i < allFields.length; i++) {
    // set result to result of tests

    // if there was an error for this field, tell the user
  }

  // return if there was an error
  return (isError); // will cancel submit if isError == false
}
```

in btValidator.js

```
btValidator.testForm = function (formElement) {
  // get all form fields
  var allFields = this.getFormElementsForForm(formElement);

 var isError;
 // iterate through each field
  var result;
  for (var i = 0; i < allFields.length; i++) {
    // set result to result of tests

    // if there was an error for this field, tell the user
  }

  // return if there was an error
  return (isError); // will cancel submit if isError == false
}
```

in btValidator.js

```
btValidator.checkField = function (field) {
  var feedback = ""; // returned
  var fieldValid = true; // returned

  // get the class names for the field and iterate through them
  var classNames = field.className.split(' ');
  for (var i = 0; i < classNames.length; i++) {
    // run tests that match with class names
    switch (classNames[i].split('_')[0]) {
      case "required":
        // do test
        // if test failed, generate message
        break;
      case "minlength":
        // do test
      case "email":
        // do test
      default:
        // unrecognised class name; do nothing.
    }
  }
  return ({valid: fieldValid, message: feedback});
}
```

in btValidator.js

```
btValidator.assertField_Required = function (field) {
      switch (field.tagName.toLowerCase()) {
          case "input":
              switch(field.type.toLowerCase()) {
                  case "text":
                  case "password":
                      return (field.value.length > 0);
                  case "checkbox":
                  case "radio":
                      return (field.checked);
                  default:
                      return true;
              }
              break;
          case "select":
              return (field.value != "" && field.value > 0);
          case "textarea":
              return(field.value.length > 0);
          default:
              return true;
      }
}
```

in btValidator.js

```
btValidator.generateError_Required = function (field) {
      var feedback = "";
      switch (field.tagName.toLowerCase()) {
          case "input":
              switch (field.type.toLowerCase()) {
                  case "text":
                  case "password":
                      feedback = "This field must not be blank.";
                      break;
                  case "checkbox":
                      feedback = "This checkbox must be checked.";
                      break;
                  case "radio":
                      feedback = "You must choose this option to continue.";
              }
              break;
          case "select":
              feedback = "Choose a valid option from the list.";
              break;
          case "textarea":
              feedback = "This field must not be blank.";
      }
      return (feedback);
}
```

in btValidator.js

```
btValidator.checkField = function (field) {
  var feedback = ""; // returned
  var fieldValid = true; // returned

  // get the class names for the field and iterate through them
  var classNames = field.className.split(' ');
  for (var i = 0; i < classNames.length; i++) {
    // run tests that match with class names
    switch (classNames[i].split('_')[0]) {
      case "required":
        // do test
        if (!this.assertField_Required(field)) { // if failed:
          fieldValid = false;
          // create user message dependant on field type
          feedback = feedback + this.generateError_Required(field) + "\n";
        }
        break;
      case "minlength":
        // ...
      case "email":
        // ...
      default:
        // do nothing.
    }
  }
  return ({valid: fieldValid, message: feedback});
}
```

in btValidator.js

```
btValidator.testForm = function (formElement) {
  // get all form fields
  var allFields = this.getFormElementsForForm(formElement);
  // iterate through each field, testing them
  var isError = false;
  var result;
  var msgbox;
  var removeErrorClassRegExp = /(^|\s)error(\s|$)/i;
  for (var i = 0; i < allFields.length; i++) {
    // do test
    msgbox = document.getElementById(allFields[i].id + "_message");
    result = this.checkField(allFields[i]);
    // if there was an error for this field, tell the user
    if (msgbox != null) {
      if (!result.valid) {
        isError = true;
        msgbox.innerHTML = result.message;
        msgbox.parentNode.className += " error";
      } else {
        msgbox.innerHTML = "";
        msgbox.parentNode.className =
          msgbox.parentNode.className.replace(removeErrorClassRegExp, "");
      }
    }
  }
  // return if there was an error
  return (isError); // will cancel submit if isError == false
}
```

in btValidator.js

```
<head>

...

<script type=’text/javascript’ src=’btValidator.js’></script>
<script type="text/javascript">
<![CDATA[

  window.onload = function () {
     btValidator.setUpEventsOnForm(document.getElementById(‘signupform’);
  }
-->
]]>
</script>

...

</head>
<body>

<form id="signupform" name="signupform" method="post" action="">
...
</form>
</body>
```

in HTML

Adding more tests is easy

```
btValidator.assertField_MinChars =
function (field, minChars) {
  if (minChars == "") return true;

  switch (field.tagName.toLowerCase()) {
    case "input":
      switch(field.type.toLowerCase()) {
        case "checkbox":
        case "radio":
          return true;
        break;
        case "text":
        case "password":
        default:
      }
    case "textarea":
      return(field.value.length > 0);
      break;
    case "select":
    default:
      return true;
  }
}
```

```
btValidator.generateError_MinChars =
function (field, minChars) {
  if (minChars == "") return "";

  var feedback = "";
  switch (field.tagName.toLowerCase()) {
    case "select":
      feedback = "";
      break;
    case "input":
      switch (field.type.toLowerCase()) {
        case "checkbox":
        case "radio":
          feedback = "";
          break;
        case "text":
        case "password":
      }
    case "textarea":
      feedback = "Must be at least " +
minChars + " characters long.";
      break;
  }
  return (feedback);
}
```

Adding more tests is easy

```
btValidator.assertField_ValidEmail = function (field) {
      var emailRegExp =
/(^[-!#$%&'*+\/=?^_`{}|~0-9A-Z]+(\.[-!#$%&'*+\/=?^_`{}|~0-9A-Z]+)*|^"([\001-\010
\013\014\016-\037!#-\[\]-\177]|\\[\001-011\013\014\016-\177])*")@(?:[A-Z0-9-]+
\.)+[A-Z]{2,6}$/i
      switch (field.tagName.toLowerCase()) {
          case "input":
              switch(field.type.toLowerCase()) {
                  case "checkbox":
                  case "radio":
                      return true;
                  break;
                  case "text":
                  case "password":
              }
          case "textarea":
              return(emailRegExp.test(field.value) || field.value == "");
              break;
          case "select":
          default:
              return true;
      }
}
```

in the real-world...

- http://www.formassembly.com/wForms/v2.0/documentation/input-validation.php
  - supports lots of fun tests
  - custom error messages per page
  - better than all the code I wrote here

AJAX

- Want to do AJAX? Learn a framework
- in jQuery – www.jquery.com

```
$("#username_message").load("reserveUserName.php",
{un: $("username").value});
```

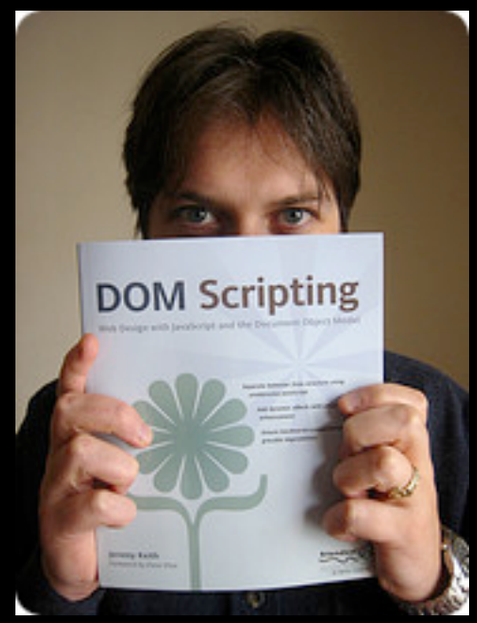
- in Prototype – www.prototypejs.org

```
new Ajax.Updater('products', '/some_url',
{ method: 'get',
  parameters: { text: $F('text') }
})
```

- Easily gets advanced quickly.
- Don’t REQUIRE Javascript or AJAX

Resources

- www.ajaxian.com – great JS blog
- www.jquery.com
- www.prototypejs.com
- script.aculo.us – animation library



- DOM Scripting – www.domscripting.com/book

- Me: http://inner.geek.nz