# Form validation

Web Pages: Behaviour

CC BY NC SA

# Forms: Validation

- Basic form operation is limited

  ***"...press submit, collect 'successful' data, send to server..."***

- Robust applications validate input prior to processing to avoid errors/issues
  - Missing data, incorrect format/type etc.
- Validation can take place at the server
  - Allows request information to be checked (referrers etc)
- But server validation requires
  - Extra round trip via HTTP
  - Maintaining of state during submit > check > resubmit

# Forms: Client-side validation

- Form data can be validated in the client
  - *Before* the HTTP request to submit occurs
- Can be used to check completeness, format, type, value etc.
- If validation succeeds then HTTP submission occurs
- If validation fails then HTTP request is never made and control drops back to the user
  - Without reloading the form/losing progress thus far

# Forms: Triggering client-side validation

- Need to capture and handle the `submit` event as it occurs in the `form` element
- Use an event handler!

```
<form method="post" action="http://..." onsubmit="return validate();" >
```

- Event handler waits for the return value from `validate()`

| Return value | Outcome |
|---|---|
| `true` | Allow submit event to continue and send data to server |
| `false` | Stop submit event and return control to the client |

# Validation: Simple checks

*"...has something been entered/selected?"*

- Access required form control objects and query appropriate state/value
  - Handle via conditional statement
  - Assemble/deliver user feedback
  - Return true/false as appropriate
- Best practice to provide one set of feedback
  - Not per question!
- Usually better to validate whole form
  - Rather than per control (e.g. via `onfocus`/`onblur` etc)

# Validation: Data checks

- Can be done via data type
  - Is it a number? string? etc
- More powerful to accurately match patterns in the input
- Unless specifically looking for a fixed string/number you'll need to enter the world of...

## Regular Expressions!

# Regular expressions

Web Pages: Behaviour

CC BY NC SA

# Regular expressions

- A "standardised" pattern matching syntax for text
  - Define pattern – test against input
- Can appear baffling at first!
- Are actually pretty logical and (relatively) straightforward to use

```
/^[\w]+([\.\w-]*)?@[\w]+(\.[\w-]+)*(\.[a-z]{2,3})(\.[a-z]{2,3})*?$/i
```

```
something(.something)@something.xx(or.xxx)(.xx or .xxx)
```

# Regular expressions: Building patterns

- Square brackets [ ]
  - Match any one of the characters or ranges in the brackets

`[ae]` matches one of a or e
`[a-z]` matches any one of the lower case letters
`[0-9]` matches any one of the digits

- Caret ^ negates a range (match anything *but*…)

`[^a-z]` anything *but* the lower case letters
`[^5-9]` anything *but* the digits 5, 6, 7, 8, 9

- Escape special characters with \

`[\[\]]` matches opening or closing square bracket
`[\.a-z]` matches a dot (`.`) or a single lower case letter

# Regular expressions: Meta-characters

- Shorthand for common ranges

| Meta-character | Matches | Equivalent range |
|---|---|---|
| `.` | Any character | `N/A` |
| `\d` | A digit | `[0-9]` |
| `\D` | A non-digit | `[^0-9]` |
| `\s` | A whitespace character | `[ \t\n\x0B\f\r]` |
| `\S` | A non-whitespace character | `[^\s]` |
| `\w` | A word character | `[a-zA-Z0-9_]` |
| `\W` | A non-word character | `[^a-zA-Z0-9_]` |

# Regular expressions: Quantifiers

| Quantifier | Effect |
| --- | --- |
| `[a-z]?` | **Zero or one** lowercase letters |
| `[a-z]*` | **Zero or more** lowercase letters |
| `[a-z]+` | **One or more** lowercase letters |
| `[a-z]{n}` | **Exactly n** lowercase letters |
| `[a-z]{n,}` | **At least n** lowercase letters |
| `[a-z]{n,m}` | **Between n and m** lowercase letters |

# Regular expressions: Greediness

- Quantifiers are *greedy* by default
  - Matching as many times as possible until end of string before *backtracking* to conclude pattern
- Try matching the opening <b> tag in... `some <b>bold</b> text`
- A simple pattern should work... `/<.+>/`
- But the quantifier + is *greedy* and keeps matching until it reaches the end of the string to find...

  `<b>bold</b> text`
- Then *backtracks* to finally match...

  `<b>bold</b>`

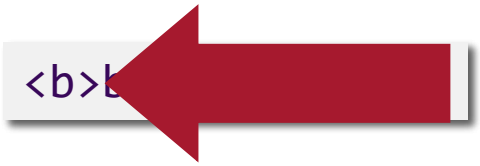
Backtrack to find the end of the pattern i.e. the *last* >

# Regular expressions: Laziness

- Append ? to the quantifier to make it *lazy*
  - Match as few times as possible before backtracking to conclude pattern

```
/<.+?>/
```

- Now it backtracks after each match to complete the pattern…meaning a match occurs after the first > character




Backtrack each time to find the end of the pattern i.e. the *first* >

http://www.regular-expressions.info/repeat.html

# Regular expressions: Anchors & flags

- *Anchors* fix expression to start/end of string or boundaries between word/non-word characters

| Anchor | Matches |
|---|---|
| `^` | The beginning of a string |
| `$` | The end of a string |
| `\b` | A word boundary |
| `\B` | A non-word boundary |

- *Flags* are appended at the end of an expression

| Flag | Matches |
|---|---|
| `i` | Use case-insensitive matching |
| `g` | Global matching (instead of stopping at first match) |
| `m` | Multiline mode |

# Regular expressions: JavaScript

- JavaScript supports regular expressions in a couple of ways:
  - via the `RegExp` object (more powerful)
  - via the `String` object (simple but less options)
- The RegExp object is defined as a pattern to match
- RegExp object methods use/test/apply pattern where needed

```
var pattern = /^[a-z]+$/i;
```

Creates a RegExp object

```
if (pattern.test(someString)){
  alert("Yay")
}
else {
  alert("Boo");
}
```

Tests string against the expression

# Regular Expressions: JavaScript RegExp methods

- JavaScript `regExp` object has two methods

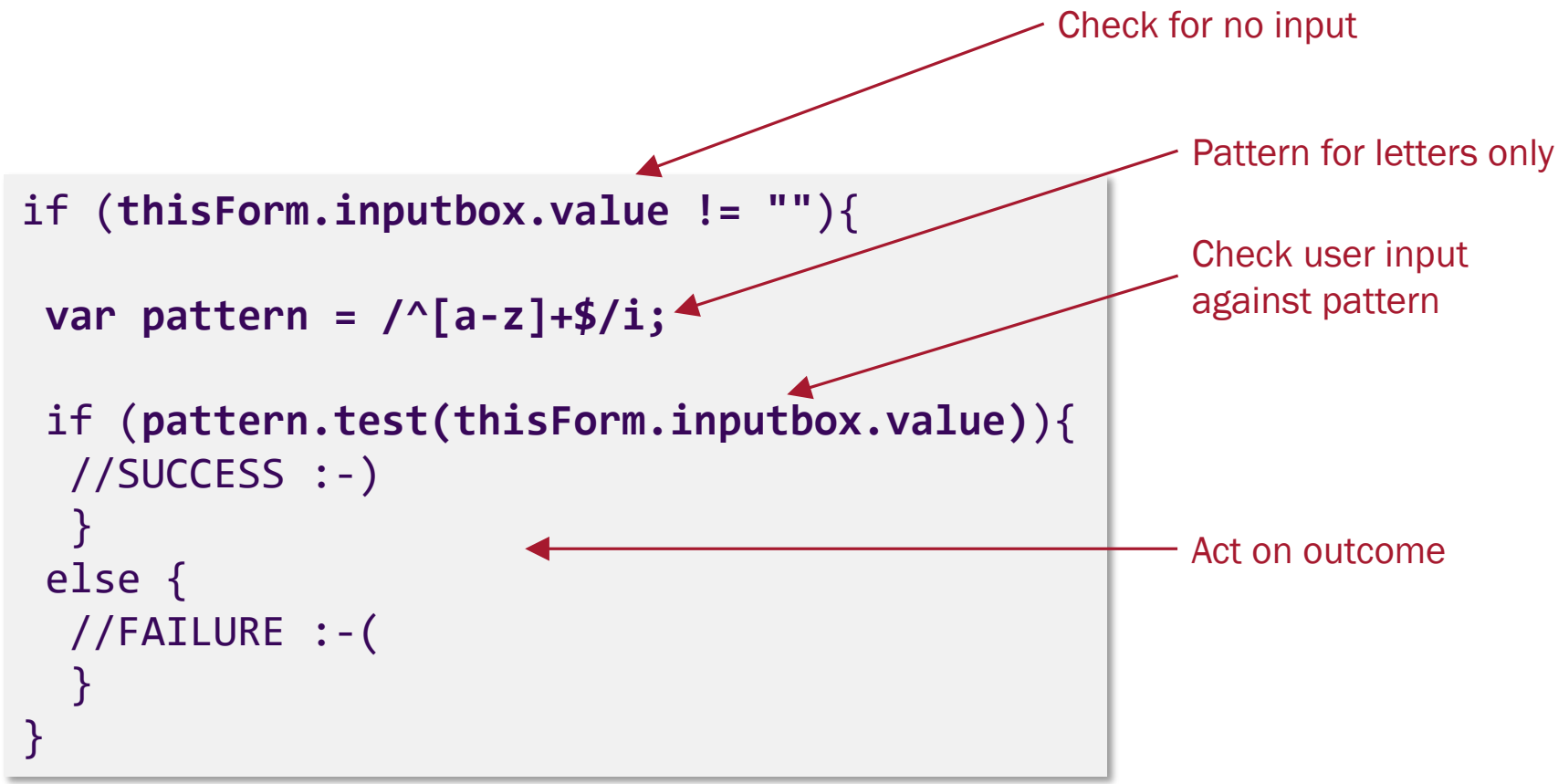
| Method | Purpose |
|---|---|
| `thisPattern.exec(someString);` | Return an array of info about the first match (or `null` if no match) |
| `thisPattern.test(someString);` | Return `true` or `false` if string contains a match |

# Regular Expressions: JavaScript string methods

- JavaScript `string` object can use regular expressions in three string matching methods

| Method | Purpose |
|---|---|
| `someString.search(/^[a-z]+$/i);` | Return position of first substring match<br>(-1 if no match) |
| `someString.replace(/^[a-z]+$/i,"X");` | Replace the text matched by expression with string in second parameter |
| `someString.match(/^[a-z]+$/i);` | Return and array containing all the matches for the expression |

# Regular Expressions: JavaScript form validation



# Regular Expressions: testing tools

- Constructing regular expressions can be fiddly
  - Try and avoid doing it in live code!
- Online testing tools are very useful – copy/paste final expression
  - Try and use a test tool using the correct expression engine i.e. JavaScript, PHP, Perl etc.
- JavaScript-based

  http://regex101.com/#javascript

  http://www.regular-expressions.info/javascriptexample.html

- General purpose (PHP-based)

  http://www.phpliveregex.com/

# Regular Expressions: Reference & tutorials

http://www.regular-expressions.info/tutorial.html

http://www.regular-expressions.info/examples.html

http://www.regular-expressions.info/reference.html

http://lawrence.ecorp.net/inet/samples/regexp-intro.php

http://mochikit.com/examples/mochiregexp/index.html

# Practical scripting

Web Pages: Behaviour

CC BY NC SA

# JavaScript in the wild

- Scripting support has stretched the horizon of what is possible in "just a web browser"
  - However poorly applied scripting can be a big contributor to poor website experiences
  - You can easily improve your scripting at two levels
- Design level
  - Use your tools wisely & add *appropriate* features
  - Practice unobtrusive scripting ☺
- Development level
  - Don't re-invent the wheel
  - Use JavaScript frameworks like **jQuery** to save time and cross-browser headaches

# Unobtrusive JavaScript

- *"Unobtrusive JavaScript"* is the name given to a collection of techniques which aim to ensure that JavaScript is used in a way that is:
  - Beneficial (to both the content and the user experience)
  - Responsible (in its use of browser resource)
  - Scalable (or removable)
- Key aims are:
  - Keep JavaScript separate from XHTML markup (separate behaviour from content)
  - *Degrade gracefully* (enhance but make sure that content is available with or without JavaScript)
  - Do not limit accessibility (and ideally enhance it)

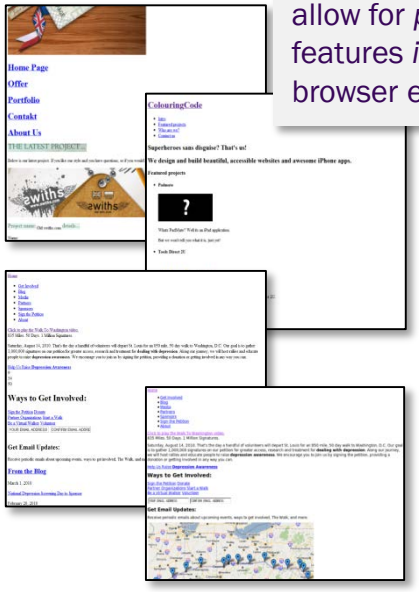
# Gracefully degrade or progressively enhance?

- Largely a question of mindset for the developer
  - Net result should be broadly similar!

**Progressive enhancement**
Design for the core functionality and allow for *progressive addition* of features *if* supported by current browser environment

**Graceful degradation**
Design for the latest browsers and allow for *graceful removal* of features *if not* supported by current browser environment

What any given user & browser combination will actually see




# JavaScript frameworks

- Pre-built libraries of common functionality
  - Save you from handling browser inconsistencies
  - Enable you to quickly and easily include complex interactions
- You do need to know something about the process at hand though!
  - Only really work if you have a clear understanding of:
    - Your XHTML structure
    - CSS selector syntax and properties
    - The desired effect (if it is appropriate for your content)

# jQuery


http://jquery.com/

- Arguably the most popular, current JavaScript library
- Easy to get started with
  - Download library, include via `<script>` tags
  - Try basic tutorial, get cracking... ☺
- Also highly extensible
  - (Lots of) third-party "plugins" for specific effects/functionality
- jQuery plugins are just more JavaScript
  - i.e. include in page alongside jQuery and run

# jQuery example

Include the core jQuery library

```
<script type="text/javascript" src="jquery-1.11.0.min.js"></script>
```

jQuery handles the DOM ready testing

jQuery handles event registration

$ sign indicates these are jQuery instructions

Built-in jQuery function for content effect

```
<script type="text/javascript">
 $(document).ready(function(){

   $("#fadeControl").mouseover(function(){
     $("#fadeAction").fadeOut('slow');
   });

   $("#fadeControl").mouseout(function(){
     $("#fadeAction").fadeIn('slow');
   });
 });
</script>
```

Use CSS selector syntax to identify where code applies in page

```
<body>
  <div id="fadeControl">Mouse over me...</div>
  <div id="fadeAction">To fade me...</div>
</body>
```

# References


http://icant.co.uk/articles/seven-rules-of-unobtrusive-javascript/

http://docs.jquery.com/Main_Page

http://docs.jquery.com/Tutorials