POWERING
POTENTIAL

Forms and regular expressions

Programming with JavaScript

Introduction

Understanding forms

- What are forms?
- Selecting form elements

Accessing form elements

- Inputs
- Radio buttons
- Select options

Events

- Form events
- Control Events
- Form validation

Regular expressions

- What is RegEx?
- Using RegEx to analyse data

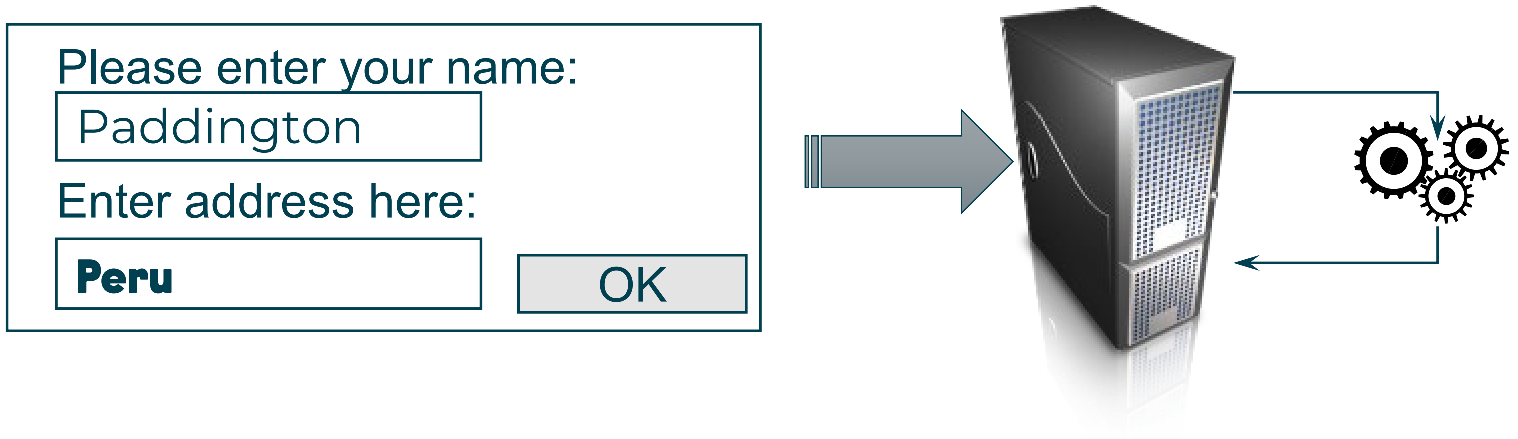
Understanding forms – what are forms?

Forms allow us to send data to the server for processing

```
<form action="/folder/enrol.aspx" method="POST">
    <input type="text" name="username" />
    <input type="text" name="address" />
    <input type="submit" value="OK" />
</form>
```

Define:

- Form – action & method
- Inputs– name, type & value



Understanding forms – HTML form inputs

- Text boxes/areas

```
<input type="text" name="username" value="">
<input type="password" name="password" value="">
<textarea name="comment" rows="10" cols="40"></textarea>
```

- Checkboxes and radio buttons

```
<input type="checkbox" name="milk" value="CHECKED">Milk?<br>
<input type="radio" name="drink" value="tea">Tea<br>
<input type="radio" name="drink" value="coffee">Coffee<br>
<input type="radio" name="drink" value="choc">Chocolate<br>
```

- Selections

```
<select name="title">
    <option value="Dr">Dr</option>
    <option value="Ms">Ms</option>
    <option value="Mr">Mr</option>
</select>
```

```
<select name="prod" multiselect>
    <option value="a">Apples</option>
    <option value="p">Pears</option>
    <option value="g">grapes</option>
</select>
```

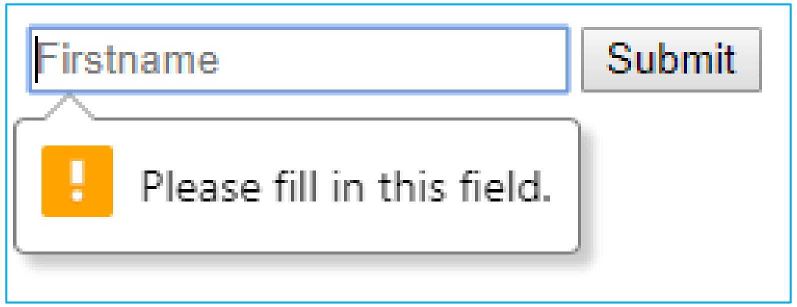
HTML5 Elements

HTML5 introduced a wave of new input types:

- `color`
- `date`
- `email`
- `month`
- `number`
- `range`
- `search`
- `tel`
- `Time`
- `datetime-local`
- `url`
- `week`

Required fields

- You can force a field to be mandatory on the client



Please fill in this field.

- On a submit action, an error message may appear:

```
<input type="text" autofocus="true" required/>
```

Pattern

- The pattern attribute allows use of regular expressions

```
<input type="text" pattern="[0-9]{13,16}" name="CreditCardNumber />
```

- We must ensure the user understands the regular expression using plain-language explanations of the requirements

Form validation

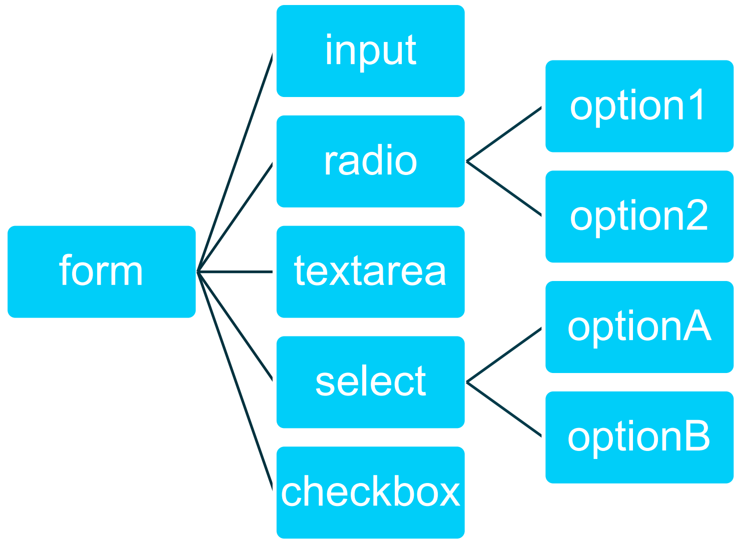
- As we have seen, some browsers ship with validation
- These are JavaScript free client validation
- Uneven support and UI feedback may be more trouble than benefit
  - You can tell a browser to switch it off
  - Still benefiting from the semantic types

```
<form novalidate>
    <input type="email" id="addr">
    <input type="submit" value="Subscribe">
</form>
```

Understanding forms – HTML hierarchy

The Form is a DOM object and container of other elements

- When a form is submitted, these details are sent to the server
  - The majority of the form fields hold a single value
  - Radio and select controls are slightly more complex



Understanding forms – selecting form elements

- You can use DOM or BOM techniques to select forms (DOM should be preferred)
  - The BOM maintains an array of form objects
  - The DOM allows id or hierarchical selection and is significantly faster
- Elements can have a name and an id attribute
  - Name is needed if you are going to submit a form to the server

```
<form name="demo" id="demo" action="process.pl">
    ...
</form>
<script>
    frm1 = document.demo;
    frm2 = document.forms[0];
    frm3 = document.forms["demo"];
    frm4 = document.getElementById("demo");
</script>
```

Accessing input elements

- Elements are sub objects of the form object
- Once selected, the input object has a series of properties.

Attribute	Value	Description
alt	text	Specifies an alternate text for an image (only for type="image")
checked	checked	Specifies that an element should be preselected when the page loads (for type="checkbox" or type="radio")
disabled	disabled	Specifies that an <input> element should be disabled
maxlength	number	Specifies the maximum number of characters allowed in an element
minlength	number	Specifies the minimum number of characters allowed in an element
name	name	Specifies the name of an <input> element
readonly	readonly	Specifies that an input field should be read-only
size	number	Specifies the width, in characters, of an <input> element
value	text	Specifies the value of an <input> element
required	n/a	HTML5 attribute that specifies this field requires a value
placeholder	text	Placeholder text to display within the element
autofocus	n/a	Instruct the browser to place focus on this field once rendered
spellcheck	n/a	Instruct the browser that spell-checking should take place on user input

Radio buttons

- Radio buttons are unique as they can share the same name value
  - But not the same id
  - Radio buttons are a mutually exclusive selection list
- To find the selected radio button in a group:
  - Select the radio buttons in the group
  - Loop over the array looking for the checked element
  - Select the value

Accessing select options

The selected element holds several useful properties to know what options have been selected:

- **`selectedIndex`**: the index of the first selected item
- **`selectedOptions`**: An **`HTMLCollection`** representing the set of **`<option>`** elements that are selected
- **`value`**: a string representing the value of the first selected item

```
let select = document.querySelector('select');
console.log(select.value);
```

Form methods and events

Properties

- Those relating to HTML attributes
  - **`action`**
  - **`encoding (ENCTYPE)`**
  - **`method`**
  - **`name`**
  - **`target`**
- Others
  - **`length`**

Methods

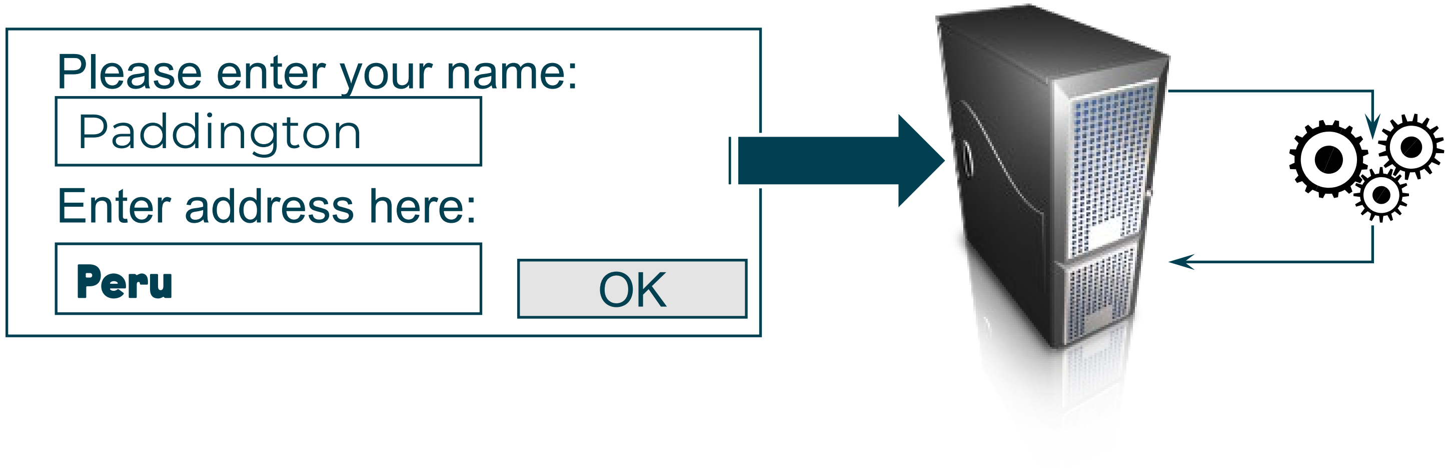
- Perform actions corresponding to form buttons
  - **`submit()`**
  - **`reset()`**

Understanding forms – form submission (GET)

- A form is submitted when a button is pressed
  - type="submit" or type="image"
- Form data is sent to server along with URL request

- JavaScript can intercept form submission for validation

```
GET /…/enrol.aspx?username=Paddington&address=Peru HTTP/1.0
```



Form events

- The events of the form object are of great importance
- When a user submits a form, they raise an **`onsubmit`** event
  - Normally the event fires, no additional interaction occurs
  - Data is transmitted to the server
- Subscribing to the **`onsubmit`** event allows you to check the data cancelling the submit action, if necessary
- There is also an **`onreset`** event
  - Fires the user has clicked a reset button
  - Use it for form clean up and wiping variables

Input element events

Main events of input elements:

- **`onfocus`**
  - The object is receiving the input focus
- **`onblur`**
  - The object is losing the input focus
- **`onchange`**
  - Edit controls only
  - The object is losing the input focus and its contents have changed
- **`onclick`**
  - Checkboxes and button types only
  - The object has been clicked
  - Especially useful with **`type="button"`** input elements
- **`onselect`**
  - Edit controls only
  - Some text has been selected within the control

Form validation – the submit event

You may either want to validate on the field or the form.

- The form validation occurs during the **`onsubmit`** event
- By using inline validation, the validation function must return a **`false`**
  - *Never do this but you may inherit it*
- In programmatic events, we can **`preventDefault()`** behaviour
  - Much the preferred approach!

```
function validateForm(evt) {
  let error = false;
  //error checking code
  if (error === true) {
    evt.preventDefault();
    //feedback to user
  }
};

window.onload = function () {
  let element = document.querySelector("#payment");
  element.addEventListener("submit",
validateForm, false);
};
```

Form validation – field validation

Field validation allows you to check the value of an individual field.

- **`blur`**, **`focus`** and **`change`** are the most important events

The **`change`** event fires when the value in a form has changed.

- For check boxes and the like, this occurs when the value changes
- Text inputs change when the user leaves the field

```
let cardtype = document.querySelectorAll('input[name=cardtype]');

for (let i = 0; i < cardtype.length; i++) {
    cardtype[i].addEventListener('change', checkSelection);
}
```

There are events to check user input when we leave or enter a field:

- **`focus`** on entry
- **`blur`** when you leave

QuickLab 12a – form validation

Creating field and page submission validation

- Hooking into events programmatically
- Checking input presence
- Reusing field validation for page submission

Regular expressions

Regular expressions are patterns used to evaluate strings

- And match character combinations
- Very useful for examining input data

In JavaScript, regular expressions are also objects

- These patterns are used with the exec and test methods of RegExp
- The String object has match, replace, search, and split methods

The regular expression pattern origin is in UNIX

- Used in JavaScript and many other Programming languages
- It’s a pattern of simple characters
- Either looking for direct matches
- Or more complex patterns

Creating a regular expression

Regular expressions are objects and must be instantiated.

- There are two ways to do this

```
let re = new RegExp("ab+c");
let quickRe = /ab+c/;
```

The regular expression object can be used to evaluate a string.

- The example below checks a string and returns a Boolean

```
let str = "bus";
/bus/.test(str); //only matches on bus
```

Excellent online testing tool: http://regexr.com/

Using regular expressions in JavaScript (1)

RegExp also has an **`exec`** method that returns an **`array`** (if there's a match) or **`null`** if there is no match

```
// Match one d followed by one or more b's followed by one d
// Remember matched b's and the following d
// Ignore case
let re = /d(b+)(d)/ig;
let result = re.exec("cdbBdbsbz");
```

The array contains:

- **`0`**: The matched text
- **`1-n`**: The parenthesized substring matches, if any. The number of possible parenthesised substrings is unlimited
- **`Input`**: The original string

Using regular expressions in JavaScript (2)

We can also revisit the string methods we looked at earlier in this course and use regular expressions:

- **`String.split`**

```
let str = 'my little string';
console.log(str.split(/\s/)); //"my, little, string"
```

- **`String.replace`**

```
let someString = 'Hello, World';
someString = someString.replace(/World/, 'Universe');
console.log(someString); // "Hello, Universe"
```

- **`String.match`**

```
let name = 'JeffreyWay';
alert(name.match(/e/g)); // alerts "e,e"
```

QuickLab 12b - Using Regular Expressions

Adding further validation

- Checking for phone numbers
- Checking for postcodes
- Checking for email addresses

Hackathon Part 1

- In this part of the Hackathon, you will build on a partially developed solution (whether that be your previous iteration or the provided starting point) for QA Cinemas' website by adding validation to the form. All the necessary tools, knowledge, and techniques have been covered in the course so far
- This part of the Hackathon is intended to help you develop your skills and knowledge to be able to use JavaScript to validate a 'Sign-Up' form for users of the QA Cinemas website before this is sent to be held on the server

REVIEW

Understanding forms

- What are forms?
- Selecting form elements

Accessing form elements

- Inputs
- Radio buttons
- Select options

Events

- Form events
- Control Events
- Form validation

Regular expressions

- What is RegEx?
- Using RegEx to analyse data

Appendix – More on Regular Expressions

JavaScript Fundamentals - Extra

Regular expression operators

- Regular expressions can look complex at times, but are just strings
- Most regular expressions are built up using special codes
  - To create the pattern we want to search for, we use special characters

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character not between the brackets
[0-9]	Find any digit from 0 to 9
[A-Z]	Find any character from uppercase A to uppercase Z
[a-z]	Find any character from lowercase a to lowercase z
[A-z]	Find any character from uppercase A to lowercase z
[adgk]	Find any character in the given set
[^adgk]	Find any character outside the given set
(red|blue|green)	Find any of the alternatives specified

Regular expression specificity and global flags

- Modifiers allow us to search for more than one match
  - Or provide case insensitive searching

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)

Regular expression specificity and global flags

- Quantifiers allow you to refine the search or sub-query a result

Quantifier	Description
n+	Matches any string with at least one n
n*	Matches any string with no occurrences of n
n?	Strings with zero or more occurrences of n
n{x}	Matches any string with a sequence of n’s
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it