
FRONT END WEB DEVELOPMENT: LESSON 15: FORMS

Objectives

- Understand how forms work on the web
- Some common form design patterns
- Build a simple form to search the web
- Understand the different form input types available to us

FORM BASICS

Form basics

- Forms are composed of a set of elements that make up an interface
- The interface might include text entry, checkboxes or radio buttons, sliders or other types of input
- A well designed form is key to conversion

USER EXPERIENCE DESIGN IMMERSIVE

LET'S CONNECT

Whether you're still exploring or you are ready to apply, you've come to the right place.

Your Full Name

Your Email

Your Phone Number

Where are you thinking of taking this course?

London

CONTINUE

Fill out your contact information to:

- Receive a prospective student handbook.
- Get in touch with an admissions counselor in your city.
- Start an application to User Experience Design Immersive.

Form design patterns

PATTERN #1

Change the layout and content of the template to maximise focus on the form (removing navigation, up-sells etc.)

PATTERN #2

Align labels to the top of the form inputs, not to the left. Eye tracking software has proven this to be more effective.

PATTERN #3

Resize form elements to fit the kind of input they are designed to take – don't have a 300-character wide input for first name



Form design patterns

PATTERN #4

Pair related form elements together into groups to enhance cohesion.

PATTERN #5

Give smart defaults for inputs where possible – for example, a user's location.

PATTERN #6

Show errors inline in the form rather than only at the top.

Form semantics

- It's possible to have more than one form on a page, so we must group inputs together inside a form element
- A form element can have non-input children, like headlines and paragraphs
- A form should have a single, obvious submission element

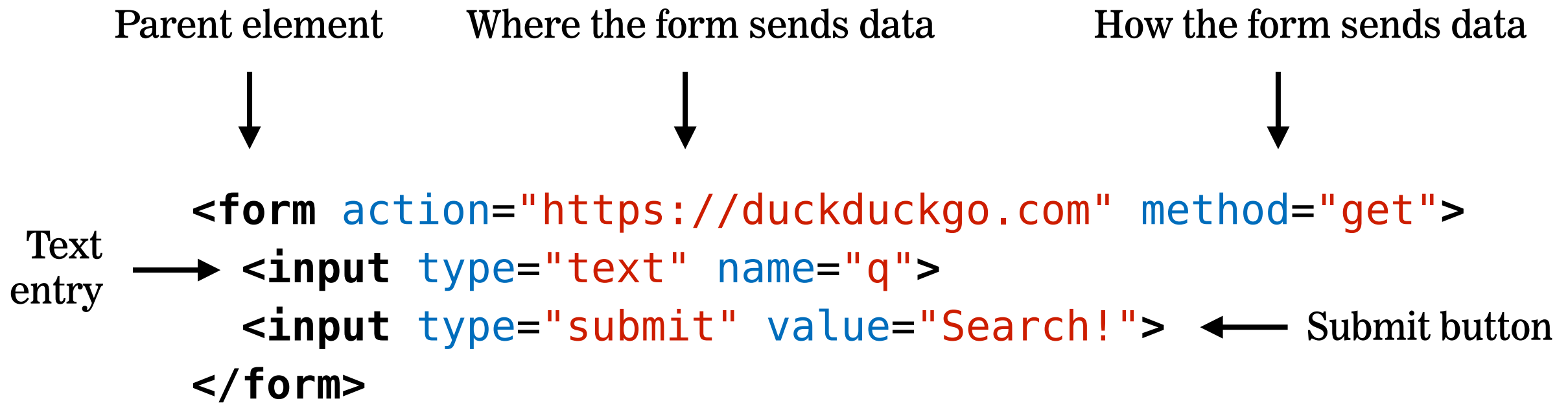
Form elements

Type	Element	Notes
Form	<code><form></code>	The parent of all a form's inputs. Has multiple attributes which set the behaviour of the form, including where it sends data to
Input	<code><input></code>	The base input type for text, checkboxes, radio buttons etc
Text area	<code><textarea></code>	The input type for long-form text content
Label	<code><label></code>	An element used to annotate a form field, associated through the name attribute
Select	<code><select></code>	Presents the user with a drop-down to select one option of many
Option	<code><option></code>	The child of the select element
Legend	<code><legend></code>	A caption for a fieldset
Fieldset	<code><fieldset></code>	A container element to group multiple inputs together logically

██████
A typical form

```
<form action="https://duckduckgo.com" method="get">  
  <input type="text" name="q">  
  <input type="submit" value="Search!">  
</form>
```

A typical form



CODE ALONG: SIMPLE FORM

UNDERSTANDING “NAME”

Form semantics

- The name attribute is applied to a form element, and is the key that gets passed with the value of the element

```
<form action="https://duckduckgo.com" method="get">  
  <input type="text" name="q">  
  <input type="submit" value="Search!">  
</form>
```

- Submitting the above form with the text “stuff” will send the user to the following URL: <https://duckduckgo.com/?q=stuff>

Form semantics

- Having further elements with further names will append the keys and values to the URL—this is called the “query string”

```
<form action="https://duckduckgo.com" method="get">  
  <input type="text" name="q">  
  <input type="number" name="limit">  
  <input type="submit" value="Search!">  
</form>
```

Result: <https://duckduckgo.com/?q=stuff&limit=20>

Form semantics

- While the **action** attribute defines where the content is sent (usually somewhere on the same domain), the **method** attribute defines how the content is sent
- When the **method** has a value of **get**, the data gets append to the **action**'s URL, like so: <https://duckduckgo.com/?q=stuff>
- When the **method** has a value of **post**, the data does not get sent through the URL, but is hidden

Form semantics

- A **get** form is generally used for searching or filtering – basically, for “getting” data
- A **post** form is generally used for sending data that gets persisted somewhere else, basically for “creating” data
- In our DuckDuckGo example, we needed to use **get** because we were retrieving data
- Read more about [HTTP verbs on Wikipedia](#)

HTML5 INPUT TYPES

HTML5 input types

- HTML5 added 13 new field types to the language
- These complement the existent text, password, checkbox, radio, file picker and similar fields
- Many of these new types add a better interface for the user should they be on a device that supports them



Alternatives to default text entry

Data	Value	Notes
Email	email	Keyboard interface with prominent ‘at sign’, period and no space bar
URL	url	Keyboard interface with forward slash, ‘.com’, etc
Number	number	Renders with a ‘spinbox’ to easily bump number up and down
Range	range	Displays a slider, given a range of two numbers, that the user can drag from left to right
Search	search	Similar to the default, but may render controls to clear the text field should the browser support it
Colour	color	Renders the operating system colour picker, returning an RGB value

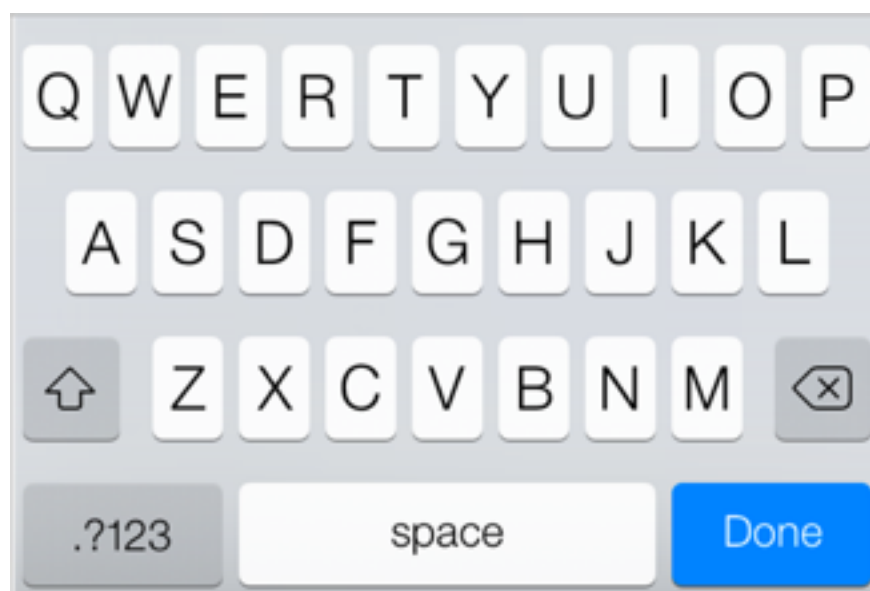




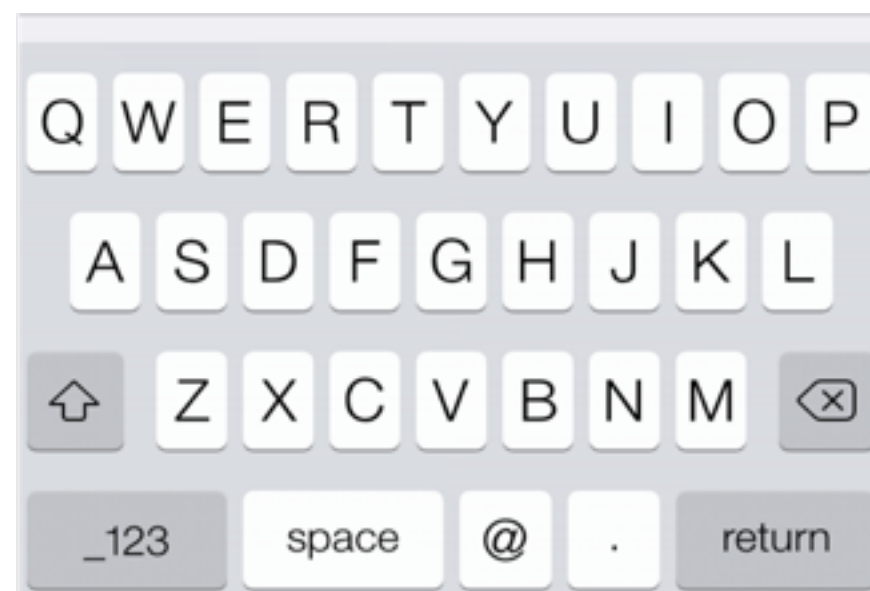
Date pickers

Data	Value	Notes
Date	date	Default YYYY-MM-DD format
Date & time	datetime	Date with time: YYYY-MM-DD HH-MM
Month & year	month	Month and year, useful for credit cards: YYYY-MM
Week	week	Pick a specific week in the year
Time	time	Similar to the default, but may render controls to clear the text field should the browser support it





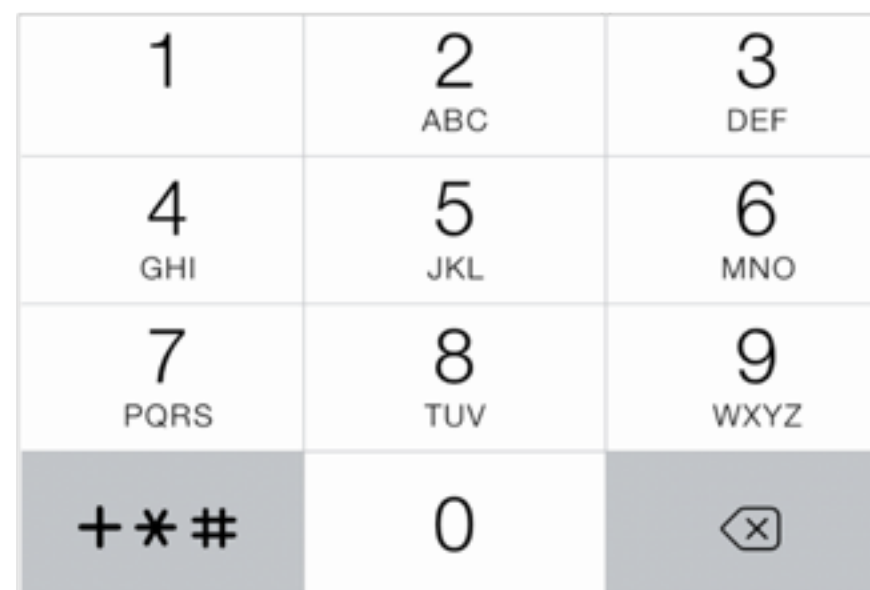
Default



Email



URL



Phone

CODE ALONG: MORE FORM ELEMENTS


FORM VALIDATION

Form validation

- It's important to validate input to a form before you submit it
- Forms should have validation of their content on both the front-end and the back-end (as front-end validation can be turned off)
- Typically JavaScript is used for form validation, but HTML5 added some form validation directly into the language
- When HTML5 validation attributes are present, the browser will validate by default

Form validation


- For example, putting a malformed email address inside an `<input type="email">` will cause the browser to stop submission of the form
- Likewise, non-URLs inside `<input type="url">` and non-numbers inside `<input type="number">` will force validation errors



Name:

Phone:

! Please fill out this field.



Name:

Phone:

Please fill out this field.

Form validation

- One of the most common validation techniques is to make a field required – this can be done with the boolean `required` attribute:

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

- Sometimes you might not need HTML5 validation in place, and you can turn it off per-form:

```
<form action="https://website.com"  
method="get" novalidate>
```

CODE ALONG: FORM VALIDATION

FORM ENHANCEMENT

Form enhancement

- It's often useful to provide placeholder text for a text field, perhaps to give an example

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

- Sometimes a field might want to steal focus on page load to make it easier for the user to enter their details – sign in forms are a good example

```
<input type="text" name="username" autofocus>
```

Form enhancement

- Sometimes you need to disable a form input but still show it

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

- You can also create hidden fields that store data to be passed along without the user interfering

```
<input type="hidden" name="somefield" value="somevalue">
```

THANK YOU

- james.willock@generalassemb.ly
- [@niceguyjames](#)