# Coding Cheat Sheet: HTML, Bootstrap

HTML

is a **markup language** which in itself is used to structure data, or present data. (vs a Programming language that is compiled and executed. A scripting language is interpreted at run-time)

uses <tags>

<!-- Comment

-->

**Structure**

<html> // 1. surrounded by html tags

<head> // 2. contains information about page

</head>

<body> // 3. everything on web page

</body>

</html>

**HTML5 Layout**  
instead of just using div for grouping everything, have new nelements.

In past had:

<body>

<header><nav></nav>

</header>

<hgroup><h1...h6></hgroup>

<div id = “content”>

<section>

<article></article>

<article></article>

</section>

<aside><section></section>

</aside>

</div>

<footer></footer>

</body>

might also be a sidebar{} using aside.

<hgroup> used to group together set or more <h1...6> elements so that they could be treated as one heading.

**Head**

**Meta**

contains information about page for browser.

**Link**

<rel = “stylesheet” type = “text/css href=”fileName.css” >

**Title**

<title>TitleName</title>

**Style**

**<**style**>** style information</style>

**Script**

**<**script type = “text/javascript” src = “directory”**>**

**Body**

Like in Head, all things including the body are **elements**.

If you give your body element font styles or colour, then they will be **inherited** by your other elements unless overrided

**Elements**

Elements are made up of enclosing tags and useful information such as **attributes**. **Enumerated attributes** have fixed set of possibilities, while **Boolean attributes** are either present (defaulting as true) or not present (equal to false).

**Global Attributes**

* accesskey =”x”– used to activate/focus element
* autocapitalize =
  + off or none – no capitals
  + on or sentences – first letter of sentence capital
  + words – first letter capital
  + characters – all letters capitals
* class
* contenteditable
  + true or *empty* string
  + false
* data-\*
  + save data in element as data-chooseDataName
  + using JS document.getElementById(‘elementName’).chooseDataName
* dir =
  + directionality of element
  + ltr – left to right
  + rtl – right to left
  + auto – algorithmic decision
* draggable =
  + true -
  + false -
  + with global event handler ondragstart="dragstart\_handler(event);
  + use Drag and Drop API <https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API>
* dropzone
  + copy – dropping will create copy of element
  + move – element will be moved
  + link – will create link to dragged data
  + use Drag and Drop API <https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API>
* hidden
  + Boolean, for element is not yet or no longer relevant
* id
* is
  + Allows you to specify that a standard HTML element should behave like a registered custom built-in element, after new element has been “extended” from existing.
* itemid
  + The itemid [global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes) provides microdata in the form of a unique, global identifier of an item. An itemid attribute can only be specified for an element that has both itemscope and itemtype attributes. Also, itemid can only be specified on elements that possess an itemscope attribute whose corresponding itemtype refers to or defines a vocabulary that supports global identifiers. <http://fortuito.us/diveintohtml5/extensibility.html>
* itemprop
  + Used to add properties to an item. Every HTML element may have an itemprop attribute specified, where an itemprop consists of a name and value pair.
* itemref
  + The [global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes) itemref properties, which are not descendants of an element with the itemscope attribute, can be associated with the item using an itemref. itemref provides a list of element id's (not itemids), with additional properties elsewhere in the document.
* itemscope
  + itemscope (usually) works along with [itemtype](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes" \l "attr-itemtype) to specify that the HTML contained in a block is about a particular item. itemscope creates the Item and defines the scope of the itemtype associated with it. itemtype is a valid URL of a vocabulary (such as [schema.org](https://schema.org/)) that describes the item and its properties context.
* itemtype
  + Specifies the URL of the vocabulary that will be used to define itemprops (item properties) in the data structure. [itemscope](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/itemscope) is used to set the scope of where in the data structure the vocabulary set by itemtype will be active.
* lang
  + The lang [global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes) helps define the language of an element: the language that non-editable elements are written in, or the language that the editable elements should be written in by the user. The attribute contains a single “language tag” in the format defined in [Tags for Identifying Languages (BCP47)](https://www.ietf.org/rfc/bcp/bcp47.txt).
* slot
  + The slot [global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes) assigns a slot in a [shadow DOM](https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM) shadow tree to an element: An element with a slot attribute is assigned to the slot created by the [<slot>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/slot) element whose [name](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/slot" \l "attr-name) attribute's value matches that slot attribute's value.
* spellcheck
  + The spellcheck [global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes) is an enumerated attribute defines whether the element may be checked for spelling errors. It may have the following values:
* true, which indicates that the element should be, if possible, checked for spelling errors;
* false, which indicates that the element should not be checked for spelling errors.
* style
  + Contains [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) styling declarations to be applied to the element. Note that it is recommended for styles to be defined in a separate file or files. This attribute and the [<style>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/style) element have mainly the purpose of allowing for quick styling, for example for testing purposes.
* tabindex
  + tab order of an element when tab button is used for navigating.
* title
* translate

**1. Content Sectioning**

elements that are used to break document up into logical pieces.

* Address
* article
* aside
* footer
* header
* h1 – h6
* hgroup
* nav
* section

**2. Text Content**

organise blocks or sections of content

* blockquote
* dd
* div
* dl
* dt
* figcaption
* figure
* hr
* li
* main
* ol
* p
* pre
* ul

**3. Inline Text Semantics**

used to define meaning, structure, or style.

* a
* abbr
* b
* bdi
* bdo
* br
* cite
* code
* data
* dfn
* em
* i
* kbd
* mark
* q
* rb
* rp
* rt
* rtc
* ruby
* s
* samp
* small
* span
* strong
* sub
* sup
* time
* u
* var
* wbr

**4. Image and multimedia**

* area
* audio
* img
* map
* track
* video

**5. Embedded Content**

* iframe

**6. Scripting**

adding scripting language support, esp Javascript

* canvas
* noscript
* script

**7. Demarcating Edits**

Shows that webpage has been edited

* del
* ins

**8. Table Content**

used for tabular data

* caption
* col
* colgroup
* table
* tbody
* td
* tfoot
* th
* thead
* tr

**Forms**

* button
* datalist
* fieldset
* form
* input
* label
* legend
* meter
* optgroup
* option
* output
* progress
* select
* textarea

**Interactive Elements**

* details
* dialog
* menu
* menuitem
* summary

**Lists**

<ul>

<li>unordered list 1</li>

<li>unordered list 2</li>

</ul>

<od>

<li>ordered list 1</li>

<li>ordered list 2</li>

</od>

**Headings**

All headings must go through h1 –> h6 in order. Only 1 h1 on each page.

<input type=”text”> //**text field**

<input type=”text” placeholder=”placeholder text”>

<**form** action=”/URL-where-you-send-form-data”></form>

<form action=”/URL”><input type=”text” **required** placeholder=”text”></form> // this blocks from submitting until form is filled out

**Input Types 5 for HTML5**

* button
* checkbox
* color 5
* date 5
* datetime-local 5
* email 5
* file
* *hidden*
* image
* month
* number
* password
* radio
  + grouping as set, use name = “groupName”
* range 5
* search
* submit
* tel 5
* text
* time 5
* url 5
* week 5

Input Validation – use **required** attribute

**Dropdown Menu**

Wrap in form with submit button

<select name="cars">  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="fiat">Fiat</option>  
  <option value="audi">Audi</option>  
</select>

**buttons**

<button type=”submit”>submit</button> //submit button used with form

**Input**

<**label**><input type=”**radio**” name=”name of radio buttons”>button1</label>

<**label**><input type=”**checkbox**” name=”name of checkboxes” **checked**> checkbox1 </label> //checked is checked by default

adding **properties**

<h1 **style**=”color:red”>inline style</h1>

<h2 **class**=”blue-class”>

<h2 **id** = “element id”> // can be used as handles

<h3 style = “**border-color**:red**”**>element</h3>

<h3 style = “**border-style**:solid**”**>element</h3>

<h3 style = “**border-width**:5px**”**>element</h3>

<h3 style = “**border-radius**:10px”>element</h3>

<h3 style = “**border-radius**:50%”>element</h3>

**importing font-families**

//place at top of page

<link href=”<https://fonts.googleapis.com/css?family=Lobster>” rel=”stylesheet” type=”text/css”>

**images**

<img src=”https://www.imagesource.com/image.jpg”>

resizing images <img src=”URL” style:500px> //can be done in CSS

<figcaption>text underneath image</figcaption>

**anchoring**

<a href=”URL”>link’s text</a>

<a href=”URL”><img src=”https://www.can\_embed\_image.com/image.jpg”>

resizing images <img src=”URL” style:500px></a>

<a href=”#”>dead link</a>

<a target="\_blank">**Opens in different tab**</a>

**div element containers**

<div>elements</div> // can use this to add properties to group of elements.

**Bootstrap – Responsive Design**

<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css"/> // add to top of page

<div class=”container-fluid”> all other stuff so that browser can resize page to fit screen</div>

<img class=”img-responsive” [image source and other stuff]>

<h2 class=”text-center”> // centres text

<button class=”**btn**”>some text</button>

<button class=”btn **btn-block**”>fill entire screen</button>

<button class=”btn btn-block **btn-primary**”>gives it a great colour</button>

<button class=”btn btn-block **btn-info**”> gives a different secondary “info” colour</button>

<button class=”btn btn-block **btn-danger**”>gives a danger colour </button>

**Bootstrap Grid**

This grid helps organise the horizontal length of the screen into 12 units. You can assign elements size elements.

<div class=”col-xs-4”> //gives 4 size elements for extra small pages

<div class=”col-md-4”> // givesd 4 size elements for medium sized pages

<p>This helps change properties within<**span** class=”text-danger”>elements</span></p>

<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css"/> //to include for font-awesome

**text input**’s can be given **form-control** classes which give it 100% width.

**icons**

<i class=”fa fa-info-circle”></i>

**row and wells**

<div class=”**row**”> //this creates columns

<div class=”col-xs-6”>

<div class=”**well**”>

//elements

</div>

</div>

<div class=”col-xs-6”>

<div class=”**well**” **id**=right-well> // can give ids to wells so that you can handle them

//elements

</div>

</div>

</div>

FONT-SIZE:

The two main types of length units are absolute and relative. Absolute units tie to physical units of length. For example, in and mm refer to inches and millimetres, respectively. Absolute length units approximate the actual measurement on a screen, but there are some differences depending on a screen's resolution.

Relative units, such as em or rem, are relative to another length value. For example, em is based on the size of an element's font. If you use it to set the font-size property itself, it's relative to the parent's font-size.