Development environment

1. Install **Node.js v8.1.2** <https://nodejs.org/en/>
2. Use npm (Node Package Manager) (<https://www.npmjs.com/get-npm>)

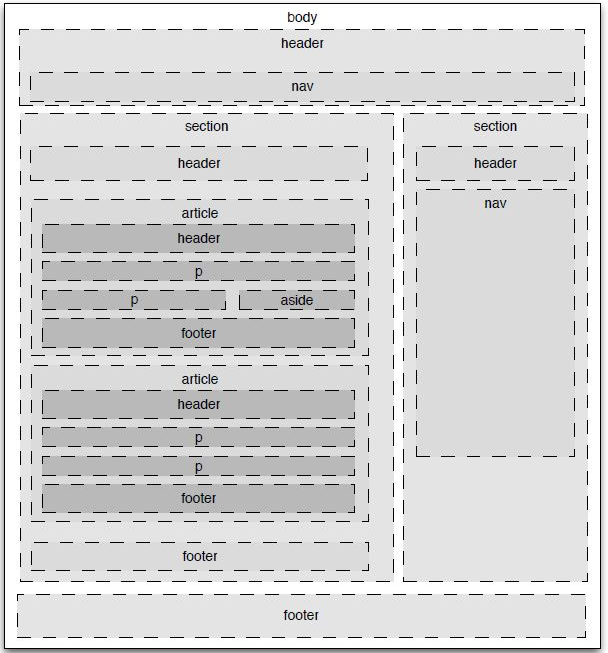
- to install all dependencies in project after pulling updated files from GitHub run command in CLI: ***npm install***

- to start a project run command in CLI: ***npm start***

- to view a web page open [http://localhost:8080/](http://localhost:3000/)

HTML

* Write valid semantic markup including semantic tags in the code



* **Document structure tags**:
  + header: an element to be used for a web page header which typically contains the site logo, heading elements, and site navigation.Also you can use a header in any section on your site.
  + footer: an element to be used for a web page footer which typically contains authorship, contact, and copyright information in addition to navigational links and a link back to the top of the web page. Also HTML5 allows us to have footer within sections.
  + main: an element used to contain all of the content that is unique to a single web page and not repeated across multiple web pages.
  + nav: an element to contain blocks of site navigation links. This element is typically placed in the page header and footer, and may also be used in an aside (sidebar) element as well.
  + section: an element is used to mark off sections of a document.Typically content identified by a heading.
  + aside: an element to identify content that is related to the main content on the page but not part of the primary flow of the document. For example, the aside element may contain a glossary definition of a term that appears in [a blog](http://www.artofblog.com/how-to-start-a-blog/) post or it may contain advertisements related to the contents of the page.
  + article: an element specifies independent, self-contained content. An article should make sense on its own, and it should be possible to read it independently from the rest of the web site. Must be identified by a heading.
  + details: an element specifies additional details that the user can view or hide on demand.The <details> tag can be used to create an interactive widget that the user can open and close. Any sort of content can be put inside the <details> tag.
* **Textual meaning tags:**
  + h1, h2, h3, h4, h5, and h6 : heading element tags are used to identify text that should appear as a heading. The highest level, or most important, heading is h1 which is followed by heading levels h2 through h6 in order of descending importance.
  + strong: text that is marked with strong tags is given added importance and is usually displayed in a bold typeface.
* **Media type tags:**
  + Audio: an element used to add one or more sources of audio content to a document and to allow the browser to pick the best option based on the visitor’s device and browser.
  + video: an element used to add video content to a markup document.
  + picture: an element is used to allow a web browser to pick the best image from the available options based on the results of a media query.
  + figure: an element is used whenever you use img, video tag with caption (figcaption) on it.If you have a number of related images (or other content) with caption text, you can use nested figure elements to associate both a group caption and an individual caption to each instance using the figcaption element.
* **Correlation tags**
  + ul: an element to signal a relationship between the items on the list and to indicate that they do not need to be understood in a specific order.
  + figure: an element is used to group together a piece of content, such as an image, chart, graph, or text, and a caption marked off by figcaption tags. By nesting the caption and the content between figure tags a relationship between the nested elements is identified. Our [images](http://html.com/images/) page contains more information about implementing this helpful tag.
* Use [The W3C Markup Validation Service](https://validator.w3.org/) to check your code <https://validator.w3.org/>
* Use CanIUse service to check if tag can be used in current version of the browser and for possible bugs.

<https://caniuse.com/>

CSS

* Use BEM (Block, Element, Modifier) methodology as a naming convention for classes in HTML and CSS.

<http://getbem.com/>

The naming convention follows this pattern:

.block {}  
.block\_\_element {}  
.block--modifier {}

* .block represents the higher level of an abstraction or component (header, container, wrapper, menu,nav)
* .block\_\_element represents a descendent of .block that helps form .block as a whole (.block\_\_header, .block\_\_item, .block\_\_img, .block\_\_inner, .block\_\_text, .block\_\_button)
* .block--modifier represents a different state or version of .block (.block--hidden, .block--visible, .block--disabled, .block--red, .block--state-success)

**Examples:**

<form class="site-search site-search--full">  
 <input type="text" class="site-search\_\_field">  
 <input type="Submit" value ="Search" class="site-search\_\_button">  
</form>

<div class="media">  
 <img src="logo.png" alt="Foo Corp logo" class="media\_\_img media\_\_img--rev">  
 <div class="media\_\_body">  
 <h3 class="alpha">Welcome to Foo Corp</h3>  
 <p class="lede">Foo Corp is the best, seriously!</p>  
 </div>  
</div>

* Use CSS nesting <http://tabatkins.github.io/specs/css-nesting/>

Nesting rules combined with BEM:

.block {  
 /\* CSS declarations for `.block` \*/  
  
 &\_\_element {  
 /\* CSS declarations for `.block\_\_element` \*/  
 }  
  
 &--modifier {  
 /\* CSS declarations for `.block--modifier` \*/  
  
 &\_\_element {  
 /\* CSS declarations for `.block--modifier\_\_element` \*/  
 }  
 }  
}

* DO NOT add additional vendor prefixes for CSS properties for that goal we use Autoprefixer that does it automatically.

<https://github.com/postcss/autoprefixer>

* Use CSSNext features <http://cssnext.io/>
* Use relative units: percents, rem and em.

**Percents** - use for layout components which will stay relational to each other (width, padding, margin)

Padding and margin in % is calculated with respect to the width of the containing block.

**REM** -i s relative to the font size of the root element. Use for parameters that will scale with page (base font-sizes for section, fixed width and height, paddings, margins etc)

**EM** - is relative to the font size of the element on which they are used.This unit is often used to define line-height, paddings, margins around headings, paragraphs, texts and other elements.

For the large screen:

button {

font-size: 1rem;

line-height: 1em;

padding: 1em 1.5em;

margin: 1em 0;

}

For the small screen we change only font-size and the rest scales accordingly:

button {

font-size: 0.7rem;

}

Font-size could be also defined in EM if the font-size of the parent block has been already defined in REM:

.header {

font-size: 1.5rem;

& h1 {

font-size: 1em;

line-height: 1em;

padding: 1em 1.5em;

margin: 1em 0;

}

}

* Use Flexbox for content layout <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
* Use media queries to make website responsive. It should be responsive and show content properly in Chrome, Firefox, Safari, IE11.

}  
  
}