* Pages: Static vs Dynamic (google news is dynamic coz it keeps updating).
* Responsive site means responsive to change in size of browser.
* HTML, CSS, JavaScript of any site is accessible.
* http://codepen.io/

**HTML**

* MDN HTML elements reference to find all the elements that can be used like body, h1, p..
* Title is what we see on google search and when opened, on the tab name.
* Headings, paragraphs are block-level elements. Not inline (unlike bold). So each heading (h1, h2,..) or paragraph take a separate line.
* Em > italic (emphasis is meaningful)
* Strong > bold (strong is meaningful)
* Divs and Spans are generic containers (For Styling)
  + Divs groups things together (for example, group images, texts and have a box around them)
  + But Divs are block level unlike Spans. So span groups inline.
* Attributes – Additional information for HTML elements. MDN HTML Attributes reference for details.
* Anchor tag is inline. Also, when mentioning a site, mention explicitly (including protocol (HTTP)). For example, <http://ww.google.com> instead of [www.google.com](http://www.google.com). Explicit is required because otherwise it would consider it a relative path. Like another html file in the same folder with have a common preceding path.
* Imgur to store images online which can be used in HTML file.
* Tables, use <thead>..</thead> and <tbody>..</tbody> to separate headings and rows of a table so that search engine understands. It doesn’t change anything visually. <tr>..</tr> denotes a row and <th>..</th> represents table heading. Table data <td>..</td> is for columns in a row.
* Form tag is like a container in which we put all the different types of inputs like radio button, drop-downs, check boxes, text boxes.
  + Method means what HTTP method (get/post) and action means where the form sends the data to.
  + Get method is by-default and if action isn’t specified then it refreshes the current \*.html file.
  + Get method (Google) doesn’t store in database, it just searches and gets data back.
  + Post method is like signup page of Facebook which gets stored in Facebook’s database.
  + Type attribute of Input element decides the type of input (MDN input).
  + Labels are important for visually impaired people. It tells what each part of the form corresponds to. Input can be nested inside label or can be separate but have same value of for in label and id in input.
  + Validation- required (blank not allowed) and data type (correct format).
    - Minlength and maxlength attributes to use for the input validation has less browser support than the pattern attribute.
    - Input type = email ensures that email if entered is in correct format.
    - <https://validator.w3.org/> (Markup validation service)
  + For radio input, we can use the same value for the name attribute in the input element so that we select only one of the radio buttons as a choice.
    - Value of value attribute is reflected in the URL on submitting. As name’s value=value’s value (radio) or the input text in the textbox.
* Textarea element is different from input type text. It lets us choose size of textbox.
* Dropdown, selected disabled will act like a placeholder.

**CSS**

* It’s better to separate HTML from CSS. Hence inline styling is a bad idea.
* If there are conflicting styles for the same element (suppose h1), then the one used later will be considered.
* Colors are
  + 6 digits hexadecimal (RGB). Use [www.colorpicker.com](http://www.colorpicker.com)
  + Or represented as rgb(rval, gval, bval) where each ranges from 0 to 255.
  + Rgba(rval, gval, bval, alpha) where alpha (0.0-1.0) for transparency.
* Background can be a color or image (set background-repeat to no if you don’t want the image to be tiled, and set background-size to cover if you want to stretch the image the entire screen).
* Border: width, style, color.
  + border: 5px solid purple;
* CSS Selectors: Element, #id and .class.
  + Same id values can be used only once a page.
  + Class values can be same in a page.
  + Descendant selectors- Select anchor tags inside <li>..</li> only.

li a{

    color: red;

}

* + Adjacent Selector: Allows us to select siblings. Here, all uls that come after h4 but in the same level, not inside.

h4 + ul{

    border: 4px solid red;

}

* Attribute Selector: All inputs of type text suppose. Or all anchors with link to google.com.
  + - * input[type=”text”]{
        + background: blue;
      * }
* Select nth of type selects suppose every 3rd li. Or select every even number.
* Inheritance and Specificity
  + Inheritance: If a body is set color pink, all elements within it will be pink.
  + Specificity: If a body is set pink, but paragraph within it is set blue, the paragraph wins as it’s more specific. Example, ***Id>class> li>ul>body***
  + Calculator*:* [*https://specificity.keegan.st/*](https://specificity.keegan.st/)
  + Type Selectors < Class, Attribute, Pseudo-Class Selector < ID Selector.
* Fonts availability In the different systems: <https://www.cssfontstack.com/>
  + Font-family, font-size, font-weight, line-height, text-align, text-decoration
  + Put font-family inside “” if the name consists of number.
  + Font-size either in px, em, rem.
    - Size of em value is for relative hence dynamic.
    - Em set 2.0 means twice the size of parent element.
    - If we set body font-size as 40 px, then we can relatively specify the size of h1, p, span (nested inside) and so on by using em.
    - Rem is not decided by parent element unlike em. They are decided by root element on the page.
    - If font-size of the root element is 16px then 1 rem = 16px for all elements. If font-size is not explicitly defined in root element then 1rem will be equal to the default font-size provided by the browser (usually 16px).
  + Font-weight can also increase page load.
  + Google fonts: <https://fonts.google.com/?sort=date>
* Box Model!!: Padding, Margin, Border, and More!
  + Every element has a rectangular box around it.
  + Spacing between elements achieved by margin.
  + Padding: Space between border and element inside it. Can choose between top, left, bottom and right like border.
  + Width: 50% means 50% of parent element. Like p inside body.
  + Blue=content, green=padding, orange=margin
  + Margin: top right bottom left. Or can put two values only which means top-bottom and right-left. Auto value is also acceptable.
* Class=”vertical middle” means two classes separated by space.
* ***FLOAT***
  + Since each div is block level, we can use ***float*** left which would consider no gravity means divs will be pushed up and then left.
  + Float also removes the whitespaces between images for suppose. These whitespaces are not through padding or margin. They are default.
* There should be only one h1 in the page. It should be the biggest thing. Otherwise use h2 or its descendants.
* This command sets width as 700px when possible otherwise 80%.

max-width: 700px;

width: 80%;

* Different hr styles: <https://css-tricks.com/examples/hrs/>

**Bootstrap**

* <https://getbootstrap.com/docs/3.3/> for version 3.3.7.
* Helps make good looking *responsive* websites very quickly.
* Navbars (converts to hamburger), Grid System, forms, buttons, jumbotrons,…
* Bootstrap.min.css will load faster than bootstrap.css but both are same.
* Bootstrap.css is more readable than bootstrap.min.css.
* Bootstrap CDN: Link to hosted version of same bootstrap file we downloaded.
* We can always overwrite the bootstrap styles.
* Container class gives nice padding and margin.