Celebrating 20 Years

Good Practices for making Responsive sites

A Larsen & Toubro Group Company

Contents

1 Dynamic content Approach

Following are some generic guidelines / good practices which can be adopted for the dynamic content of web sites

1.1 Table base content

1) Horizontal Scroll

- To create responsive table using pure CSS, table content can be wrapped by horizontal scroll with an “overflow: auto” property
- As per bootstrap solution, wrap the table inside a div element with a class of Table-responsive. By default, bootstrap overflow-y: hidden property is applied in small screen. Therefore, on small devices you can see the contents of your tables by scrolling horizontally.

2) If columns have little content it might squash horizontally on a small screen without changing the layout
3) If table has fix columns and limited records then it can easily reformat (via only CSS) to make each row a bit like its own table.

Reference link: https://css-tricks.com/examples/ResponsiveTables/responsive.php

1.2 Lazy Loading

Transforming lazy loading with ‘load more’ button to scroll. It will be user friendly, so user can view the content as per his requirement.

2 Recommended practices to follow while developing responsive application

As per the current trend, making an application responsive is a necessity as lot of users try to access the application or sites from their devices. Users can access application on the go and perform the required activity. Supporting multiple devices through responsive can be achieved by implementing @media query into CSS or using any responsive framework available in the market such as Bootstrap, Foundation etc. Below are the best practices which need to be followed while making application responsive.

1) Keep Clean and Simple Layout Structure
   Application layout design plays an important role in making a responsive site. It is always good to have Clean and Simple Grid base layout which can be easily be modified into responsive.

   Some web applications have few complex layout patterns which are difficult to fit on smaller devices. For example, tables that have more than 6 columns, edit functionality applied on 'Edit' section. For smaller screens, all big tables and complex functionality need to be redesigned considering the smaller resolutions. Suggestion – High Priority columns can be displayed by default and the tables can be expanded if user requires. Tables can be made to fit in to a wrapper for smaller resolution with a horizontal scroll functionality

2) Use Common Design and layout Patterns
   Grid base framework helps to build a common design or template pattern with inbuilt responsive behavior. For example, as per resolution we can manage to show multiple blocks in a single row. For higher resolutions, more number of blocks can appear in a single line but on smaller devices it will appear one below the other. Also, top navigation will be visible on higher resolution but on smaller screens it will handled by hamburger menu.

3) Set Content Priority
   Setting Priority of application content is important because smaller screens cannot show all your data in the same way as bigger resolutions. Setting priority to the content allows developer to control the data to be shown on the smaller devices.

4) Responsive Images/ Icons.
   - Instead of using single image for each icon it would be good to use Sprite Group base images which help to avoid multiple http request and help to reduce the load time.
   - Latest browsers support SVGs (Scalable Vector Graphics) formatted image and icons. By looking at this support developers can use SVG base icons like Font Awesome (http://fontawesome.io/), IcoMoon (https://icomoon.io/), Fontello (http://fontello.com/) and GLYPHICONS (http://glyphicons.com/) etc. This helps to present content visually to the users on smaller devices.
   - Further to above two solutions, icon images can be converted into base64 version and used by defining its values into CSS files.
   - Use ICONS where it's appropriate

   In case in an existing site, multiple icon images are being used as background through CSS in a filter section to define each filter visually. Instead of using icon image, it can be converted to base 64 (https://www.browserling.com/tools/image-to-base64) and used as background image to remove icons http request and improve performance.

5) **Handle Interactive Areas**
Applications will have may interactive areas (buttons etc.) where user need to click to perform specific activity. These all areas need to be standardized by keeping them in good and bigger size, so user can easily interact with this element. As a recommended, clickable area should have at least 45 pixels in height.

6) **Responsive Typography and Font**
It is important that Content of the application is readable on small devices. To achieve content readability, Typography need to be made responsive with help of font responsiveness. Developer can use “em” and “%” unit combination to define font responsive rules into CSS file in a @media query.

7) **Minifying application files**
While doing actual development user writes a code in a proper format to have a better code understanding and easy to modify. However, once code is finalized and ready to be deployed on production, it is recommended to minify all the files used in the application. Files like JS, HTML and CSS can be minified and use in the final deployment version. Minifying files help to reduce file size and help to improve the load performance.

8) **Design from Smallest to largest View**
Designing a UI layout is an important process in application development. Designing for smaller devices first will help to set the content and element priority from the lower one. It is recommended to have mobile wireframe first and use them as model for larger desktop design. Responsive layout is also a content first approach as just making layout responsive will not help if useful content is not present on the smaller devices.

For this activity, user feedback can be taken to set priority of the content and functionality which needs to be available on small devices.

9) **Keep minimum breakpoint**
Today, different several devices with different resolutions are available. However before starting on responsive layout it is good to finalize your resolution breakpoints. This will be helpful to prepare responsive template. For example, Bootstrap framework using four resolution breakpoints to support larger (1200px and above), medium (992px and above), small (768px and above) & very small devices (below 768px). These breakpoints will help to finalize the layout for each resolution.

3 General Coding Practices to write HTML and CSS

Following are the general coding practices used for writing HTML and CSS code.

Recommended HTML code practices

1) Define a DocType

DocType is the head of every HTML file. Doctype sets the rule while displaying the HTML page into browser as it communicates with the browser verifying whether the page contains HTML, XHTML or a combination of both. DocType helps browser to interpret the markup correctly. In HTML5, defining Doctype is simpler than old version. As per HTML5, doctype get is define as

```
<!— Start of Code-->
<!DOCTYPE html>
<!—End of Code -->
```

2) Define Proper HTML Syntax

HTML language has many tags and each tag has specific syntax to use. Proper closing tag should be used in HTML. For example: HTML code follows basic format as shown below:

```
<!— Start of Code-->
<!DOCTYPE html>
<html>
<head>
<title>Document Title</title>
</head>
<body>
        HTML Content
</body>
</html>
<!—End of Code -->
```

In the above code, each tag has a closing tag associated to it. However, in HTML two types of tags are present and they are known as “Paired” and “Unpaired” tags. Paired tags follow opened and closed tag rule. However Unpaired tag gets closed by its own.

Paired tag example:

```
<!— Start of Code-->
<p>Some Text</p>
<ul><li></li></ul>
<!—End of Code -->
```

Unpaired tag example:

```
<!— Start of Code-->
<img src=”” alt=”” />
</br>
<!—End of Code -->
```

3) Avoid using Inline Style

Defining inline style in the HTML markup makes project maintenance difficult. Inline style is not recommended to use in any HTML file. Also in a responsive web application it is difficult to overwrite inline style for particular resolution and can lead Style sheet with unwanted “!important” string. It is better to create appropriate class and reference that class from external stylesheet.

4) **Declare External CSS Files into Head Tag**
   For better application maintainability, it is always good to have external CSS file to hold your style rules. As such there is no rule to define CSS in your html file. However, it is recommended to declare your external CSS file into HEAD tag. It helps application pages to load progressively and faster.
   <!— Start of Code-->
```
<head>
      <title>Application Title</title>
      <link rel="stylesheet" type="text/css" media="screen"
href="file-path/to/stylesheet.css" />
</head>
```
   <!—End of Code -->

5) **Declare JavaScript File at Bottom and Avoid Using Inline JS**
   In the current technology era, whole solutions are developed by using different JS framework and libraries. All these approaches require multiple JS files reference in one single HTML files. However, it's also important to load the application quickly for user interaction. Browser loads JS files in a sequence and hold next continuity till the entire file gets loaded. It is recommended to place all the JS files at the bottom of HTML page before closing of BODY tag. It helps to load the page content quickly and sync the JS files later.

6) **Validate HTML and CSS Code with W3C standard**
   W3C (World Wide Web Consortium) is an international community that develops open standard and group of people work together to develop Web Standard. W3C has inbuilt tool which help to validate HTML and CSS code written by developer. These tools help to reduce code error and keep the code as per the industry standard.
   HTML Validation Tool: https://validator.w3.org/
   CSS Validation Tool: https://jigsaw.w3.org/css-validator/

   For further details, please go through following W3C link:
   https://www.w3.org/standards/

7) **Define HTML Tags with Lower Case**
   It is recommended to define HTML tags in lower case to keep the code consistency and readability. For Example:
   <!— Start of Code-->
```
<div>
        <p> Content Here </p>
 </div>
```
   <!—End of Code -->

8) **Use Semantic Element to Build Appropriate Section**
   For a list of links in application - it would be good to define the structure of navigation with Unordered list and handle the look and feel with CSS properties. HTML5 holds semantic tags like HEADER, NAV, SECTION, ASIDE, FOOTER, etc. which can be used to build proper layout structure.

9) **Use Minified OR Compress Version of HTML and CSS Files**

It is always recommended to minify or compress final version of HTML and CSS files. Default HTML and CSS files follow certain syntax formats and code indentation to keep code understandable and easy to modify. This increases file size unnecessarily and such files can impact application performance and load time. These issues can be handled by minifying OR compressing all the files while putting application on the production.

10) Use Alt Attribute into IMG Tag
Some of the attributes are default to use into tags for example Alt attribute into IMG tag. This attribute helps user if in case image could not load into the browser. Alt tag text be presented into the browser at broken image place as an informer to the user.

```
<!— Start of Code-->

<img src="filter-icon.jpg" alt="Filter by Age" />
<!—End of Code -->
```

11) Use Proper ID and Class Attribute
ID's are majorly used to identify HTML elements from JavaScript and they are unique in nature. Classes are used for applying look and feel to the element by defining required CSS properties into STYLE file. Avoid using ID's for look and feel. It's good to use IDs for template creation stuff so each template section will have unique ID associated to it.

```
<!— Start of Code-->
<div id=”leftpanel”></div>
<div id=”rightpanel”></div>
<!— End  of Code-->
```

Recommended CSS code practices

1) Use Reset CSS
All the HTML elements have default browser properties assigned to it and they render differently in multiple browsers. It is always recommended to use reset CSS in your Style. Most of the CSS Framework has inbuilt reset CSS features. Reset CSS can be retrieved from following links
https://meyerweb.com/eric/tools/css/reset/index.html

2) Use Comments Flags to Define Area Specific Style
CSS files contains whole presentation layer code and at one movement finding specific area related style is difficult. Keep the style organize with sections and put comments at the start of each section.

```
<!— Start of Code-->
/* Header Section */
.header { CSS Properties }

/* Content Section */
.tile { CSS Properties }
<!— End  of Code-->
```

3) Single Property in a Line
Defining multiple CSS properties in single line is not a good practice. This creates difficulties to find property and its value for future change. While developing a style file, it always recommended to declare one CSS property at one line. It increases file readability and understanding the CSS code.

```
<!— Start of Code-->
```

```
        .cssClass {
                background: #F0F0F0;
                color: #000;
        }
<!— End  of Code-->
```

4) Use Shorthand Properties

CSS provide facility to define properties with shorthand approach. For example, developer can define four different padding properties to handle each side or use shorthand approach.

```
<!— Start of Code-->
.cssClassName {
                padding-top: 10px;
                padding-right: 5px;
                padding-bottom: 10px;
                padding-left: 5px;
        }
```

Shorthand approach

```
        .cssClassName {padding: 10px 5px;}
<!— End  of Code-->
```

5) Use Color Code

CSS files support color name as values. User can mention color property value as a color name (Red) which is not a good practice to match the output with UX provided by designer. It would be and recommended to use Hexadecimal color code values while defining color.

```
<!— Start of Code-->
        .cssClassName {
                Background-color: #FF0000;
        }
<!— End  of Code-->
```

6) Not to use units with ZERO value

It's observed that users define ZERO value with units like em, rem, percentage and pixels. ZERO is ZERO whether it has unit value or not. To cut down the extra code it's good not to have unit for ZERO value.

```
<!— Start of Code-->
        .cssClassName {
                padding: 0 5px;
        }
<!— End  of Code-->
```

7) Flexible Modularize Style

Many a time it's seen that developers define multiple classes with same properties. It can be avoided in order to create more flexible and modularized style by defining common set of properties within one class and can be used in multiple instance.

```
<!— Start of Code-->
```

Non Flexible code

```
        .pass-event {
                padding: 0 5px;
                background: #F0F0F0;
                border: 1px solid #ccc;
        }
```

```
    .fail-event {
        padding: 0 5px;
        background: #F0F0F0;
        border: 1px solid #ccc;
    }
```

Flexible Modularize code

```
    .event {
        padding: 0 5px;
        background: #F0F0F0;
        border: 1px solid #ccc;
    }
```

<!— End of Code-->

8) Use CSS Frameworks

Currently in the web technology open source CSS frameworks are available and can be explored to speed up a development process. For example, developers can start using Bootstrap or Foundation framework. All this framework provides lot of inbuilt layout features and components which can be leveraged and help to complete Frontend development in a shorter time.

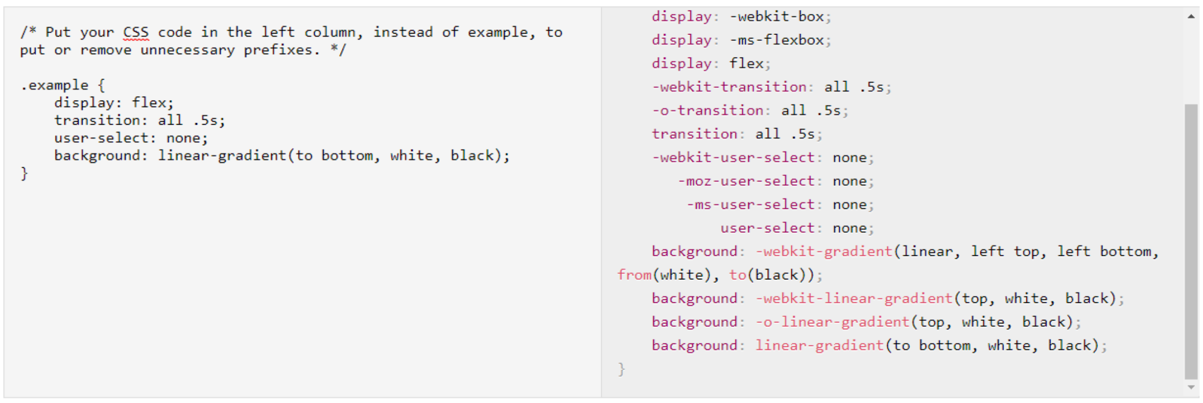
9) Using Modularized CSS with CSS Preprocessor

CSS preprocessor provides lots of feature which are not available in plain CSS version for example- defining variables, reusable function, nesting, mixins, extends and if else condition etc. All these extra features help to avoid writing lots of code and reuse the existing code. Also, this helps create section wise CSS files and merge into a result file. Currently SASS (http://sass-lang.com/) and LESS (http://lesscss.org/) are the two best CSS preprocessors available as open source.

10) Use Browser Specific Auto Prefixers

Certain CSS properties are supported by browser with specific prefixers. It is difficult to maintain all the CSS files with all prefixers. Either developer can define all prefixers in CSS files while defining such properties or simply use a tool which does this task.

Auto Prefixers: https://autoprefixer.github.io/



```
/* Put your CSS code in the left column, instead of example, to
put or remove unnecessary prefixes. */

.example {
    display: flex;
    transition: all .5s;
    user-select: none;
    background: linear-gradient(to bottom, white, black);
}
```

```
    display: -webkit-box;
    display: -ms-flexbox;
    display: flex;
    -webkit-transition: all .5s;
    -o-transition: all .5s;
    transition: all .5s;
    -webkit-user-select: none;
       -moz-user-select: none;
        -ms-user-select: none;
            user-select: none;
    background: -webkit-gradient(linear, left top, left bottom,
from(white), to(black));
    background: -webkit-linear-gradient(top, white, black);
    background: -o-linear-gradient(top, white, black);
    background: linear-gradient(to bottom, white, black);
}
```

4 Reference

Google Developer:
https://developers.google.com/search/mobile-sites/mobile-seo/responsive-design

Bootstrap Framework Official site
https://getbootstrap.com/docs/3.3/

Smashing Magazine
https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/
https://www.smashingmagazine.com/2011/07/responsive-web-design-techniques-tools-and-design-strategies/

CSS Tricks
https://css-tricks.com/responsive-data-tables/
https://css-tricks.com/nine-basic-principles-responsive-web-design/
https://css-tricks.com/snippets/css/media-queries-for-standard-devices/

HTML and CSS best practices
https://code.tutsplus.com/tutorials/30-html-best-practices-for-beginners--net-4957
https://www.sitepoint.com/css-architectures-new-best-practices/
https://css-tricks.com/css-style-guides/
https://hackhands.com/70-Expert-Ideas-For-Better-CSS-Coding/
https://www.w3.org/standards/techs/htmlbp#w3c_all
https://www.w3schools.com/html/html5_syntax.asp