# Responsive Web Design Techniques

Waseem I. Bader
Al-Salt College for Human Sciences,
Al-Balqa Applied University,
Al-Salt, Jordan

Abdelaziz I. Hammouri
Department of Computer
Information Systems,
Al-Balqa Applied University,
Al-Salt, Jordan

## ABSTRACT
As new devices and technologies are invented to access the internet, from computer desktops, laptops, mobile phones to smart TVs, there has been a great need to upgrade the techniques used in the field of website design, because these new devices come along with their own specific sizes and views. Although most devices & technologies try to be as compatible as possible with the common web design features, but there has been an absolute need for website designers to do a lit bit more to adapt to the fast growing race in internet devices and provide all their viewers with the best possible experience while accessing their websites. In this paper, different responsive website design techniques are presented that could adapt to different technologies and devices while at the same time focusing on cutting down the time and effort needed for a website designer or programmer to maintain and edit it. The code samples in this paper were implemented using the ASP.net programming language, but it can be implemented in other development languages and environments using the same concepts.

## Keywords
Responsive Website Design, Dynamic Website, Adaptive Design

## 1. INTRODUCTION
Web application development has come a long way since the beginning of the World Wide Web. The web environment today uses HTML and CSS to view data and content to users while JavaScript is used to interact with the client. These technologies are called "front-end" or "client-side" technologies. "Back-end" or "server-side" technologies on the other hand refer to the technologies that execute on the server before the response is sent to the client, which are responsible of data storage and processing technologies, etc. [1][2]

In the early web development period, website designers didn't often need to worry much about how their website will look like to different clients, because in general most of the users would access their work from often similar desktop computers with close screen resolution ranges, and their work would look very similar to the original design made by the designer. This has enabled many web designers to even use static specific dimensions in their web design.

But today internet technologies are upgrading fast and different devices are used and developed to access the internet, therefore websites developed and designed by web designers and programmers are viewed and accessed by a large number of devices with different screen resolutions, orientations and views.

This fast upgrade in internet accessing devices has significantly improved the number of users accessing the internet around the world [3][4], but at the same time it has added more work on website designers, because now they have to deal with many viewing devices and technologies accessing their work. These devices have different size ranges and capabilities making their work a wonderful beauty on one device and a total mess on the other.

Nowadays users access the same website from desktop computers, laptops, mobile phones, iPhones, iPads, Blackberries, notebooks, feed readers and even smart TVs. Each platform displays the same page in a different feel from the others depending on its size and viewing capabilities.

Whenever a user enters a website, the client looks for a user-friendly interface, quick access to his/her needs and a comfortable content view without the need to worry about how they are accessing it. [5] On the other hand, web designers and programmers have to guarantee, as much as possible, to provide such an experience to all their clients and from all the different internet-accessing technologies.

Nowadays, with the huge improvement in mobile phones and its ever growing increase in their popularity, as published reports indicate that at least 65% of people aged 18-29 are accessing the internet using mobile devices [6], therefore more responsive or adaptive website design techniques are needed and used to ensure the best possible viewing experience to the users.

**Figure 1: Different Devices with various screen sizes and resolutions**

Responsive Web Design (RWD) is a name given to the set of techniques used to develop one single website which adapts itself on different devices and is capable of reshaping itself depending on various screen sizes, resolutions and orientations from the largest devices like the internet TV to the smallest ones on mobile devices, another name used to describe it is "Adaptive Web Design" which as its name indicates refers to the techniques used in a website to enable it to adapt to different viewing devices. **[7][8]**

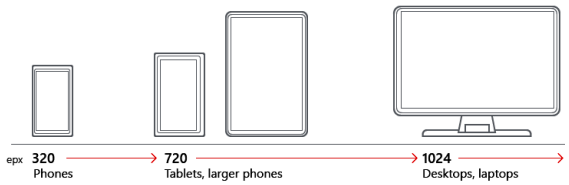


**Figure 2: Different Internet Devices Resolutions**

## 2. RESPONSIVE WEB DESIGN TECHNIQUES

There are three common techniques to prepare a responsive website design:

- Fluid grid layouts (Relative-based grid).
- Flexible images and media.
- Media queries and screen resolution.

### 2.1 Fluid Grid Layouts

A website is generally built in a grid-based layout using HTML tables or divs. Responsive web design is applied on the website grid layout to enable it to be viewed smoothly on multiple devices by specifying percentage-based dimensions to the web page grid, this allows that part to resize or rearrange based on the current viewing size. **[9]**

For example, if a website contains a header, a menu and a content part with a width of 1024 pixels as a whole as shown in Figure.

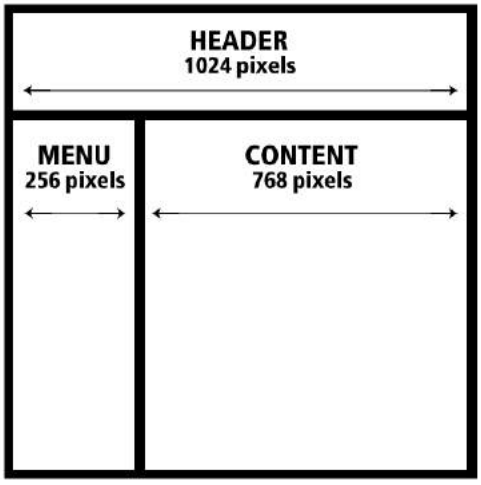


**Figure 3: Static Grid Layout**

If the widths of the three parts are specified as static numbers then the view of the website will be constant on all the different devices, thus, larger screened devices will show the content to be very small, while smaller ones will be too big to view it as shows in figure:

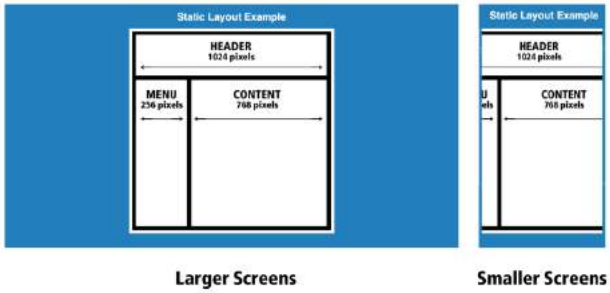


**Figure 4: Static Layout Example**

So by using the Fluid Grid Layout technique the dimensions of the desired page are set in proportion to the maximum size of the page. Thus the width of the header will be 100% which is correspondent to the 1024 pixel of the whole page, for the menu, the 256 pixels is equivalent to 25% of the maximum size which is calculated as (256 / 1024) * 100 = 25% and the content part will be (768 / 1024) * 100 which is equal to 75% of the total page, so the resulting relative-based values of the same page will be:

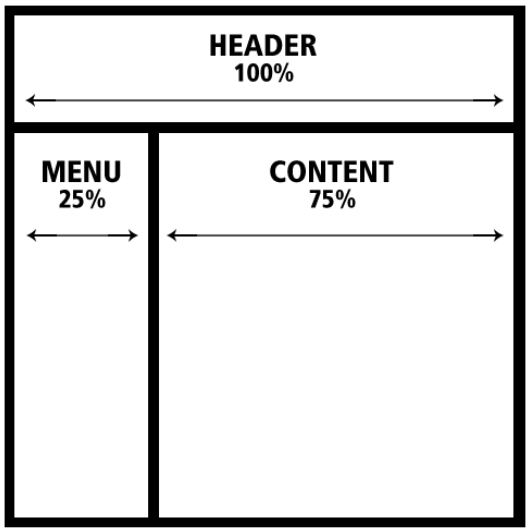


**Figure 5: Dynamic Grid Layout**

This can be achieved using HTML as follow:

```
<html>  <body>
  <table  width="100%">
    <tr>
      <td colspan="2">HEADER</td>
    </tr>
    <tr>
      <td  width="25%">MENU</td>
      <td width="75%">CONTENT</td>
    </tr>
```

```
                    </table>
                </body></html>
```

**Figure 6: Fluid Grid Layout Code**

Or by using the cascading style sheet (CSS) as shown:

```
<html>    <head>
          <style type="text/css">
              #header { width:100%;}
              #menu { width:25%;}
              #content { width:75%;}
          </style>
    </head>
    <body>
          <table width="100%">
              <tr>
                  <td id="header"
colspan="2">HEADER</td>
              </tr>
              <tr>
                  <td id="menu">MENU</td>
                  <td
id="content">CONTENT</td>
              </tr>
          </table>
    </body></html>
```

**Figure 7: Fluid Grid Layout Code with CSS**

Therefore with the Fluid Grid Layout technique, the webpage will be viewed with regards to the percentage perspective of the current viewing device.


**Figure 8: Fluid Grid Layout on different devices**

## 2.2 Flexible images and media

In this technique images or media in a webpage will automatically resize or crop in regards with the current viewing size or resolution, this can be done either by using relative dimensions as in the previous section, cropping extra parts of an image in smaller screen sizes or even by completely hiding an image as screen size decreases **[10]**.

**Using relative dimensions:**

Similar to the Fluid Grid Layout technique, relative dimensions are used for images (or media) in a webpage instead of using static values, for example for a 512px image in an 1024px page, a relative percentage number is used instead of 50% so now its size will expand or shrink based on the entire page dimensions to fill exactly half the page.

**Figure 9: Relative Image Dimensions**

This can be done by just specifying a 50% width for the image width as follows:

```
<img src="aqsa.png" width="50%"/>
```

**Figure 10: Flexible Images Code**

**Cropping images:**

An image can also be cropped to a specific width by using some cascading style sheet (CSS) techniques as follows:

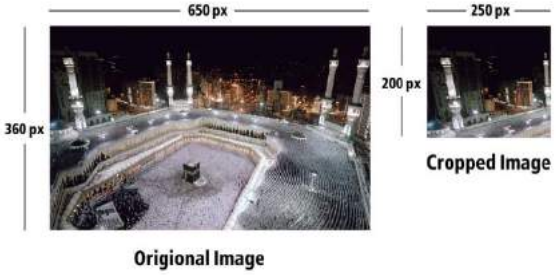


**Figure 11: Image Cropping**

```
<html>          <head>
        <style type="text/css">
                #cropdiv
{       width:  200px;          height:  150px;
overflow: hidden;       }
        </style>
</head>
<body>
        <div id="cropdiv">
        <img src="kabaa.png" />
        </div>
</body>    </html>
```

**Figure 12: Image Cropping Code**

In this technique an image is cropped to a certain view window and all the extra part of the image will be hidden.

Although in this example cropped width and height must have static values but in the next section another technique is discussed on how to prepare different sizes for a crop container or even hide it completely using media queries.

## 2.3 Media Queries And Screen Resolutions

This RWD technique uses CSS3 media queries capabilities to provide multiple fixed versions of an element for each screen resolution range as follows [10][11][12]:

```
<html>
    <head>
        <style type="text/css">
            @media screen and (max-width:400px)
            {
                img { width: 200px; }
            }
            @media screen and (min-width:401px) and (max-width:799px)
            {
                img { width: 400px; }
            }
            @media screen and (min-width:800px)
            {
                img { width: 800px; }
            }
        </style>
    </head>
    <body>
        …
        <img src="aqsa.png" />
        …
    </body>
</html>
```

**Figure 13: Media Queries Code Sample**

In the above example CSS3 media queries will be selected based on the current viewing resolution, that is all browsers viewing the current page from any source with a width less than 400px will show the image with a fixed 200px width, likewise browsers with resolution widths between 401px to 799px will show the same image but with a 400px width, and all other browsers with widths over 800px will show the image with an 800px width. This techniques is clearly different from the relative values used before which will change the width of an image in proportion with every resolution value, here in this example on three different "steps" of widths are provided for specific resolution ranges.

Notice that the "**screen**" value means use this media query if the content is viewed on any screen, there is another value that can be used here which is "**print**" which means that this media query will be used when printing it out only. So there can be different styles between the screen and print out versions of a webpage besides their resolutions.

Some other important options for the media query are using "**min-height**" and "**max-height**" which clearly means the current minimum and maximum height of the viewing browser. There is also an option to check whether the current view is in "**portrait**" or "**landscape**" view, by checking the "**Orientation**" value of the media query, for example the following media query will be used only for viewing browsers with minimum width of 400px and in its landscape orientation and only when printed:

```
@media print and (min-width:400px) and (orientation: landscape)
            { //add desired css style here }
```

**Figure 14: Media Queries Code Sample B**

This technique can also be used to choose different sources for an image, for example one source would be prepared for high screen resolutions and another for lower ones and by using media queries it is possible to check the users screen resolution and choose the right source to view. Thus a low-resolution copy of an image can be prepared to save some transmitting bandwidth to smaller resolution devices as shown in the next example:


**Figure 15: Different Image Sources**

```
<html>
    <head>
        <style type="text/css">
            @media screen and (max-width:400px)
            {
                img [data-src-A]
                {
                    content: attr(data-src-A, url);
                }
            }
            @media screen and (min-width:401px)
            {
                img [data-src-B]
                {
                    content: attr(data-src-B,
```

```
url);
                    }
                }
            </style>
        </head>
        <body>
            …
        <img src="aqsa.png" data-src-A="aqsa.png"
data-src-B="aqsa_big.png" />
            …
        </body>
</html>
```

**Figure 16: Media Queries Code Sample with Different Image Sources**

So depending on the current viewing resolution and using CSS3 media queries one of the two prepared images will be used, notice how in the HTML "img" tag, the urls of both copies are set with the user defined attributes to be matched in the style sheet.

CSS3 media queries is a very powerful technique to create Adaptive web design, it can also be used to change the whole viewing experience of a website, for example it can be used to hide or show elements based on different sizes so that less important parts can be eliminated in the viewing port in order to save more space for more important parts of the page.

**Figure 17: CSS3 Media Query Example**

As seen in the previous figure, the same webpage is shown in different ways depending on the viewing size. Larger screens show 3-columned page layout, medium screens show 2 columns and smaller screens will show only 1 column. To do so, CSS3 media queries and "**display**" property is used as follows:

```
<html>
    <head>
        <style type="text/css">
            @media screen and (max-
width:400px)
            {
                #div1 { display: block;}
                #div2 { display: none;}
                #div3 { display: none;}
            }
            @media screen and (min-
width:401px) and (max-width:799px)
            {
                #div1 { display: block;}
                #div2 { display: block;}
                #div3 { display: none;}
            }
            @media screen and (min-
width:800px)
            {
                #div1 { display: block;}
                #div2 { display: block;}
                #div3 { display: block;}
            }
        </style>
    </head>
    <body>
        …
        <div id="div1">….</div>          …
        <div id="div2">….</div>          …
        <div id="div3">….</div>          …
    </body>
</html>
```

**Figure 18: Hiding Elements Using Media Queries**

As shown in example, "display" property will be toggled between "**block**" which makes the element visible in the webpage and "**none**" which indicates a hidden element based on different screen sizes.

Also a complete external style sheet file with its specific look and feel can be used for different screen size by using media queries as follows:

```
<html>
    <head>
        <link rel="stylesheet" type="text/css"
media="screen and (max-device-width: 480px)"
href="styleA.css" />
        <link rel="stylesheet" type="text/css"
media="screen and (min-device-width: 481px)"
href="styleB.css" />
    </head>
    <body>
        <p> Hello World! </p>
    </body>
</html>
```

**Figure 19: Different External CSS Style Sheets With Media Queries**

Where both external CSS files contain their own specific styles for example:

```
...
p { font-size:12pt; }
...
```

**styleA.css**

```
...
p { font-size:18pt; }
...
```

**styleB.css**

**Figure 20: External CSS Style Sheets Example**

This will show the "Hello World!" paragraph with a font size 12pt in devices with maximum resolution width 480px while it will be shown with font size 18pt for others as shown in figure:


**Figure 21: Different external style sheets applied by media queries**

Notice that the difference between "**min-width**" and "**min-device-width**" is that "min-width" specifies the minimum width of the browser, while "min-device-width" specifies the minimum device screen width, which will be different in case the used browser, is not maximized to full screen of the device screen as shown in figure.


**Figure 22: Difference between min-width and min-device-width**

Also notice that there are others means of achieving responsive typography that can be used such as by specifying the font size using "em" or "%" values instead of pixels or points. As pixels and points represent fixed font-sizes, while both "em" and "%" values will represent relative font size of the general page font size.

For example if the website general font size is 16px, a containing part with a font size equal to "1em" is equivalent to "100%" which is 16px in this case. If the font size is changed to "2em" it is equivalent to "200%" which is 32px, "0.5em" or "50%" can also be used to represent half the general font size which is 8px in the example.

| | body { font-size: 16px; } | body { font-size: 32px; } |
|---|---|---|
| font-size: 1em | Hello World 16px | Hello World 32px |
| font-size: 12pt | Hello World 16px | Hello World 16px |
| font-size: 16px | Hello World 16px | Hello World 16px |
| font-size: 100% | Hello World 16px | Hello World 32px |
| font-size: 0.5em | Hello World 8 px | Hello World 16px |
| font-size: 200% | Hello World 32px | Hello World 64px |

**Figure 23: CSS font-size examples**

An important note should also be kept in mind that CCS3 media queries are not supported for all web browsers available. Though it is supported on most newer browsers versions such as in Internet Explorer (IE) 9+, Firefox 3.5+, Safari 3+, Opera 7+, as well as on smart phones and others. **[13]**

For older browsers versions there are some JQUERY libraries that can be imported and used instead to perform the same results.

## 3. SERVER-SIDE RESPONSIVE WEB DESIGN

The three common responsive web design techniques discussed in the previous section is of great help to create an adaptive website design for most of the internet devices, at the same time, there are some server-side RWD techniques that could take some more time to prepare, but serve as a great help in achieving a Responsive Web Design.

Taking into considerations that there are some devices and browsers which are not compatible with CSS3 media queries, these server-side coding techniques can serve as a good help as well. In these techniques the webpage is prepared to suite the current client viewing window as much as possible, and the web designer will determine the best techniques to use in his work.

These techniques will be explained through a code example, so first a "**RWD**" class (layer) will be prepared that will deal with the adaptive web design functionalities:

```
public class RWD
{    ...    }
```

**Figure 24: Responsive Web Design Server-Side Layer**

### 3.1 Desktop/Laptop vs. Mobile Requests:

The first function needed in the "**RWD**" layer is "**isMobileRequest**" which determines if the current request to the website is coming from a desktop computer or from a mobile device, where the function return true if so and false otherwise.

```
public class RWD
{
    public static Boolean isMobileRequest( )
    {
        Boolean res = false;
```

```
            ...
            return res;
    }
}
```

**Figure 25: isMobileRequest function in RWD Layer**

Every Http Request sent from any internet device to a website contains a value of the current internet browser used to send the request and is stored in the server variable "**HTTP_USER_AGENT**", for example the value of the User-Agent Server variable in the HTTP request sent from a Chrome browser would look like: **[14]**

**Environment**

| Variable | Value |
|---|---|
| PATH | /usr/local/bin:/usr/bin:/bin |
| CTK_ERROR_DOCUMENT | 404.html |
| DOCUMENT_ROOT | /www |
| HTTP_ACCEPT | text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5 |
| HTTP_ACCEPT_CHARSET | ISO-8859-1,*,utf-8 |
| HTTP_ACCEPT_ENCODING | gzip,deflate,bzip2 |
| HTTP_ACCEPT_LANGUAGE | en-US,en |
| HTTP_REFERER | http://www.google.com/search?q=phpinfo+HTTP_USER_AGENT |
| HTTP_USER_AGENT | Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/525.13 (KHTML, like Gecko) Chrome/0.2.149.27 Safari/525.13 |
| REDIRECT_STATUS | 200 |
| REDIRECT_URL | /identifiant/info.php |

**Figure 26: Google Chrome User Agent Sample**

This server variable can be accessed through code to determine the current request source device. In the following code snippet the result of the HTTP_USER_AGENT is referenced by the String reference "**useragent**".

```
public class RWD
{
    public static Boolean isMobileRequest( )
    {
        Boolean res = false;
        String useragent =
HttpContext.Current.Request.ServerVariables["HTT
P_USER_AGENT"];
        ...
        return res;
    }}
```

**Figure 27: Accessing HTTP_USER_AGENT in Server-Side Code**

Then the source browser should be compared with the available mobile browsers and if the match is found then this request was sent from a mobile phone, else it is sent from a desktop computer or a laptop. In the example Regular expressions will be used to store the available mobile browsers and regular expression functions to match the user agent values. **[15]**

In general, if the HTTP_USER_AGENT value contains one of the following patterns then it is coming from a mobile phone:

- Mobile
- iPhone, iPod, iPad -> this can be translated in Regex as "iP(hone|od|ad)"
- Android
- BlackBerry
- IEMobile
- Kindle
- NetFront
- Silk-Accelerated
- hpqOS, webOs -> "(hpw|web)OS" in Regex
- Fennec
- Minimo
- Opera Mobi, Opera Mini -> "Opera M(obi|ini)" in Regex
- Blazer
- Dolfin
- Dolphin
- Skyfire
- Zune

Thus the complete Regex expression would be: "**/Mobile|iP(hone|od|ad)|Android|BlackBerry|IEMobile|Kindle|NetFront|Silk-Accelerated|(hpw|web)OS|Fennec|Minimo|Opera M(obi|ini)|Blazer|Dolfin|Dolphin|Skyfire|Zune/**".

These user agent values are specific for mobile browsers and cover almost all of them. In the following code snippet a complete extended list of mobile phone browsers brands is used to determine if the current request is coming from a mobile phone.

```
public class RWD
{
    public static Boolean isMobileRequest( )
    {
        Boolean res = false;
        String useragent =
HttpContext.Current.Request.ServerVariables["HTT
P_USER_AGENT"];
        Regex brand = new
Regex(@"(android|bb\d+|meego).+mobile|avantgo|
bada\/|blackberry|blazer|compal|elaine|fennec|hipto
p|iemobile|ip(hone|od)|iris|kindle|lge
|maemo|midp|mmp|mobile.+firefox|netfront|opera
m(ob|in)i|palm(
os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)
0|symbian|treo|up\.(browser|link)|vodafone|wap|win
dows ce|xda|xiino", RegexOptions.IgnoreCase |
RegexOptions.Multiline);
            if (brand.IsMatch(u))
            {
                res = true;
            }
        return res;
    }}
```

**Figure 28: Complete isMobileRequest function in RWD Layer**

Where the brand Regular Expressions try to match the current browser with the available mobile browsers and if matched a "true" value is returned from the function.

After preparing the "**isMobileRequest**" it can be used anywhere in the website to determine if the current request is coming from a mobile device or not as follows:

```
if (RWD. isMobileRequest( ) )
{           //code specific for mobile requests
}
else
{       //code specific for desktop computers
& laptops requests   }
```

**Figure 29: Using isMobileRequest Function Example**

Next a "**Pick**" function can be prepared that has 3 parameters:

- **Computer value**: this value should be used if request is coming from a desktop computer or a laptop.
- **Mobile value**: this value should be used if the request is coming from a mobile phone.
- **All value**: this value should be used in all cases.

This function will automatically pick the correct value based on the current request source as follows:

```
public class RWD
{
    public static Boolean isMobileRequest( )
   {… }
public static String Pick(String compVal, String
mobVal, String all)
{
      if (isMobileRequest())
    {           return mobVal + " " + all;
}
    else
   {           return compVal + " " + all;
}
  }}
```

**Figure 30: Pick function in RWD Layer**

After preparing the "**Pick**" function it can be used anywhere in the website to choose between two values one for the desktop/laptop computers and another for mobile computers for example:

```
String imgwidth = RWD.Pick( "800","400","px"
);
```

**Figure 31: Using Pick Fucntion Example**

This function will return "800px" for desktop/laptop computers and "400px" for mobile phones. Then the "**imgwidth**" can be used in the webpage as a dynamic value as follows:

```
<img src="logo.png" width="<%= imgwidth %>"
/>
```

**Figure 32: Dynamic Binding Example**

The previous code creates an image in the webpage with the width based on the "**imgwidth**" value returned from the "**Pick**" function. Notice that the dynamic value is bind to the html "**img**" tag attribute using ***ASP.net inline displaying expression***. **[2]**

An Overloaded version of the "**Pick**" function can be used without the third parameter "**all**" for faster usage.

```
public class RWD
{
    public static Boolean isMobileRequest( )
   {       …      }
public static String Pick(String compVal, String
mobVal, String all)
{       …       }
public static String Pick(String compVal, String
mobVal)
{       return Pick(compVal, mobVal,"");       }
}
```

**Figure 33: Overloaded version of Pick**

Where the same above example can be used as follows:

```
String imgwidth = RWD.Pick( "800px" ,
"400px" );
```

**Figure 34: Using Overloaded Pick Example**

## *3.2 Server-side Client Resolution Access:*

Moreover, with a lit bit of work the client screen resolution can be accessed from the server-side code with the use of ASP.net hidden fields and some JavaScript **[16]**, here is a code sample:

```
<html><body>
<form id="form1" name="form1" runat="server">
    <asp:HiddenField id="clientScreenWidth"
runat="Server" >
    <asp:HiddenField id="clientScreenHeight"
runat="Server"/>
        <%
            try
            {
                int hw = Int32.Parse("" +
clientScreenWidth.Value);
                int hh = Int32.Parse("" +
clientScreenHeight.Value);

Session.Add("clientResolutionWidth",hw);

Session.Add("clientScreenHeight", hw);
                if (hw > 1200)
                {    %>    BIG SCREEN
LAYOUT    <%    }
                else
                {  %>    SMALL SCREEN
LAYOUT    <%    }
            }
```

```
                catch (Exception exp)
                {//EMPTY SESSION VALUES
}
        %>
        </form></body></html>

<script language="javascript"
type="text/javascript">
        function getClientResolution() {
                document.getElementById('<%=
clientScreenWidth.ClientID %>').value =
window.screen.availWidth;
                document.getElementById('<%=
clientScreenHeight.ClientID %>').value =
window.screen.availHeight;
        }
        window.onload = function (e) {
if(<%=(""+Session["clientResolutionWidth"]).Equal
s("")).ToString().ToLower()%>)
                {
                getClientResolution();
                document.forms['form1'].submit();
                }
        }
</script>
```

**Figure 35: Server-Side Client Resolution Access Code**

In this code, first two asp.net hidden field values are prepared to store the client's screen resolution width and height. After that through a try/catch block an attempt is made to convert the two values stored in the hidden fields from String to number, if an exception occurred then there is no values stored for the client screen width and height yet in the hidden fields. These values will be stored later through the JavaScript code, since these values cannot be accessed directly from server-side code.

If no exception occurs then the two hidden fields contains the client screen width and height and the two values will be stored in the server session so that it can be accessed everywhere in the website. Using the Session to store these values will help us perform this code only once during the client visit to the website. Then these values can be used simply through conditions like in the previous example to show different layouts based to specific resolutions.

Finally the JavaScript Code at the end of the page, contains a function called "**getClientResolution**" which will store the current client screen resolutions into the hidden fields, The last step is on the window "**onload**" event, that the server session value for "**clientResolutionWidth** " is checked if not found –which means this is the first time this user accessed the page- then the "**getClientResolution**" is executed and the form is submitted to execute the code again with the client resolution values submitted this time and stored in the hidden fields. The client resolution values will be stored in the Server Session for the whole time the user is communicating with the website even for all the other pages thus this step will be performed only once per website session. **[2]**

Although preparing such server-side layers might appear to take a lot of time, but the programmer should take into consideration that such layers are prepared just once and then can be used in any project later. So the time is only consumed once to prepare the layer.

# 4. RESPONSIVE WEB DESIGN ANALYSIS

Responsive Web Design techniques have given website designers great capabilities to achieve dynamic adaptive web contents and have been a great breakthrough in the web development process. These techniques have given web designers a lot of benefits and at the same time have had some limitations.

## 4.1 Advantages of Responsive Web Design

Some of the advantages of Responsive Web Design **[13]** are:

- Broadcasting content from a single webpage to multi-devices and automatically adapting and resizing content to the screen by making it easily read and friendly viewed on each device, instead of making a different copy of the webpage (or even a different website) for each view which is very hard to maintain, edit or upgrade.
- Users experience will be more fun and comfortable without the need to zoom in or out or find it hard to reach their needs since the display of the website is properly viewed based on the current user device.
- Responsive Web Design can be used to serve up lower-bandwidth images to mobile devices and save browsing time and bandwidth.
- Hide non-essential elements on smaller screens to save space for more important elements.
- Provide larger finger-friendly links and buttons for mobile users.
- The Company saves more time and money by not maintaining a mobile friendly site.

## 4.2 Disadvantages of Responsive Web Design

Here are some of the disadvantages of Responsive Web Design:

- It is harder to implement by website designers and programmers.
- Some of the Client-side Responsive Web Design is not supported on all browsers or devices.
- The need to update the list of browser agents in the server-side RWD for newly developed browser companies.
- More processing time needed to switch the current view into the proper display for the current device.

# 5. CONCLUSION

Since internet devices and technologies are changing very fast there has been a great need of continuously adapting responsive web designs to go along with it. In this paper, different techniques are represented to achieve Responsive

Web Design showing by example how it can be implemented, but these techniques are still in their early stages, new ones will be produced as more internet devices and technologies arrive. Web designers will also provide different new opinions and techniques on how to achieve complete Adaptive website design with minimum overhead costs and will also continue to work in order to find new ideas to match the changing world of devices, browsers and programming technologies.

## 6. REFERENCES

[1] Dragos-Paul Pop, Adam Altar, "Designing an MVC Model for Rapid Web Application Development" presented at the 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

[2] J. Liberty, D. Hurwitz, Programming ASP.NET, 3rd ed. O'Reilly Media. California, United Stated of America. (2006).

[3] "Individuals using the Internet 2005 to 2014", Key ICT indicators for developed and developing countries and the world (totals and penetration rates), International Telecommunication Union (ITU). Retrieved 25 May 2015.

[4] "Internet users per 100 inhabitants 1997 to 2007", ICT Data and Statistics (IDS), International Telecommunication Union (ITU). Retrieved 25 May 2015.

[5] Sharki, C. & Fisher, A. (2013). Jump Start Responsive Web Design, Sitepoint Pty. Ltd: Australia.

[6] A. Gustafson "Adaptive Web Design", 2nd ed. New Riders, United Stated of America. (2015).

[7] Smith, A., (2010). Mobile Access 2010. Pew Internet & American Life Project, Jusly 7, 2010.

[8] S. Hay "Responsive Design Workflow", 1st ed. New Riders, United Stated of America. (2013).

[9] MOHAMED, A.A., An Enhanced Approach to Responsive Web Design Influid Grid Concept, 2015, JKUAT.

[10] B. Frain "Responsive web design with HTML5 and CSS3", 2nd ed. Birmingham : Packt Publishing Limited. (2015).

[11] C. Simmons "Instant Responsive Web Design". Birmingham : Packt Publishing Limited. (2013).

[12] Gardner, B.S., Responsive web design: Enriching the user experience. Sigma Journal: Inside the Digital Ecosystem, 2011. 11(1): p. 13-19

[13] Meltem Huri Baturay, Murat Birtane, "Responsive web design: a new type of design for web-based instructional content", Ipek University, Ankara, Turkey, 2013.

[14] D. Gourley, B. Totty, M. Sayer, A. Aggarwal, S. Reddy "HTTP: The Definitive Guide", 1st ed. O'Reilly Media, Inc. (2002).

[15] J. Friedl "Mastering Regular Expressions", 2nd ed. O'Reilly Media. California, United Stated of America. (2002).

[16] E. Brown "Learning JavaScript: JavaScript Essentials for Modern Application Development", 3rd ed. O'Reilly Media. California, United Stated of America. (2016).