Responsive Web Design

Responsive Web Design

- RWD is one of the simplest and quickest ways to make a website work nicely on a lot of devices, you can use the web skills you already have.
- Depending on the current value of certain browser conditions like window size, device orientation, or aspect ratio, we can apply different CSS in different circumstances.
- Three primary techniques for building a responsively designed website:
    1. CSS3 media queries
        1. Evaluating certain aspects of the current browser environment to determine which CSS to apply. Example, browser window width, aspect ratio, and orientation.
    2. Fluid-grid layouts
        1. Using relative CSS proportions instead of absolute sizes for page layout elements. (Use percentages instead of pixels)
    3. Fluid images and media
        1. Making our images and media scale to fit within the bounds of their parent elements, scaling proportionally with the rest of the layout.

CSS media queries

- CSS3 media queries are logical expressions that evaluate the current values of media features in the user's browser. If the media query expression evaluates as TRUE, the contained CSS is applied.
- Examples,
  - Does it render content on a screen, and is the window currently at least 480 pixels wide? If yes, apply these CSS rules.
    - **@media screen and (min-width:480px) { /* CSS Rules */ }**
  - Is this being rendered on a printer OR is it being rendered on a screen that is monochrome (black and white)? If yes, use these styles!
    - **@media print, screen and (monochrome) { }**
  - Apply these styles to all media types when in landscape orientation.
    - **@media all and (orientation: landscape) {}**
  - Apply the rules in this external stylesheet to color screens.
    - **<link rel="stylesheet" type="text/css" href="my.css" media="screen and (color)" />**
  - Apply these styles to black and white printers
    - **@media print and (monochrome) {}**
  - Apply these rules to color screens.
    - **@media screen and (color) { }**

Checklist for making existing desktop website to support mobile devices

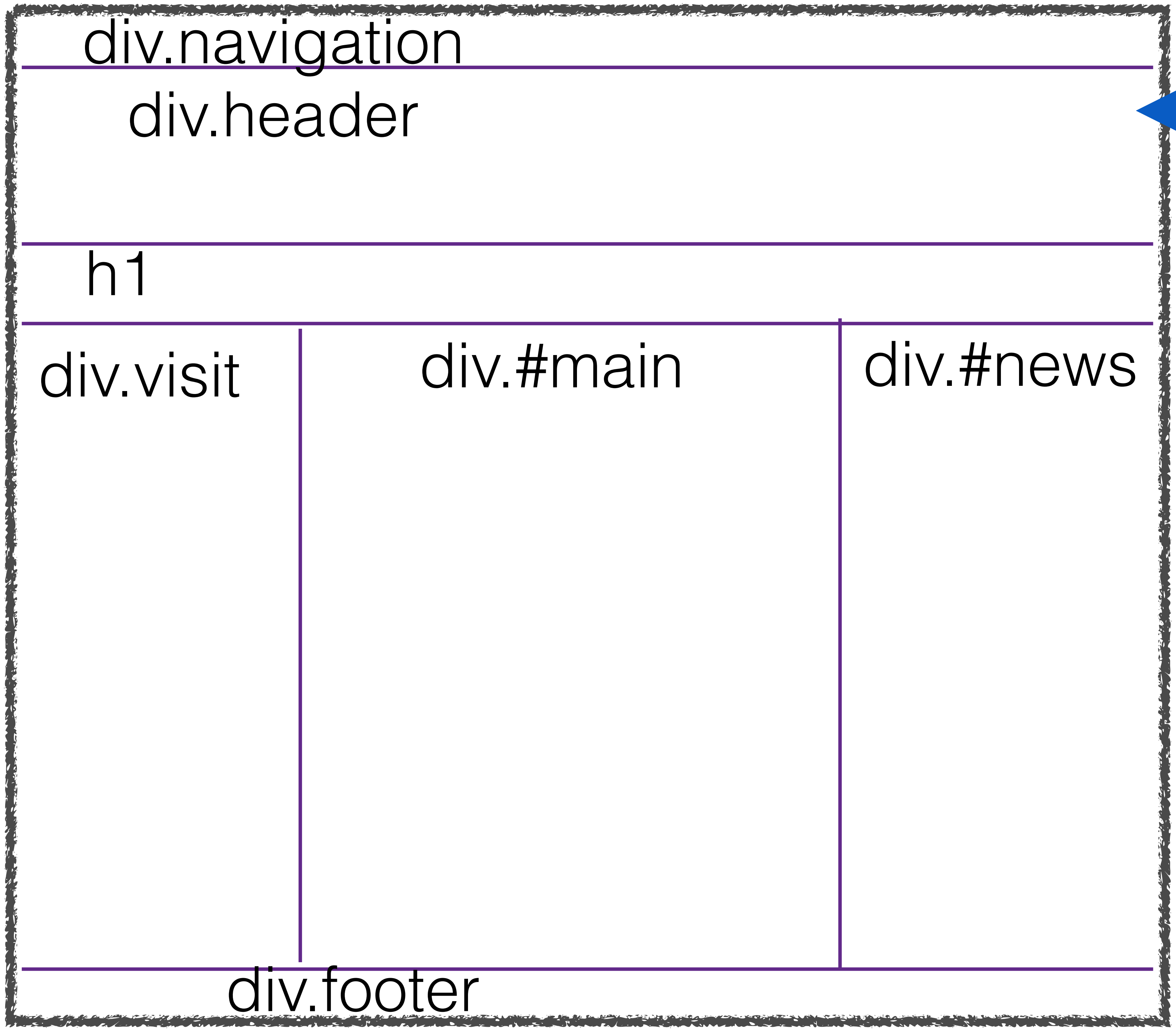
A. Check out the current layout of your existing website and analyze its structure.
B. Identify layout pieces that need to change to work better on mobile browsers.
C. Generate mobile-adapted CSS for those identified elements.
D. Organize our CSS and selectively apply the mobile and desktop CSS using media queries.

Example of supporting mobile devices

Basic HTML file

```
<div class="navigation">...</div>
<div class="header">...</div> <h1>...</h1>
<div id="visit" class="column">...</div>
<div id="news" class="column">...</div>
<div id="main" class="column">...</div>
<div class="footer">...</div>
```

Example of supporting mobile devices



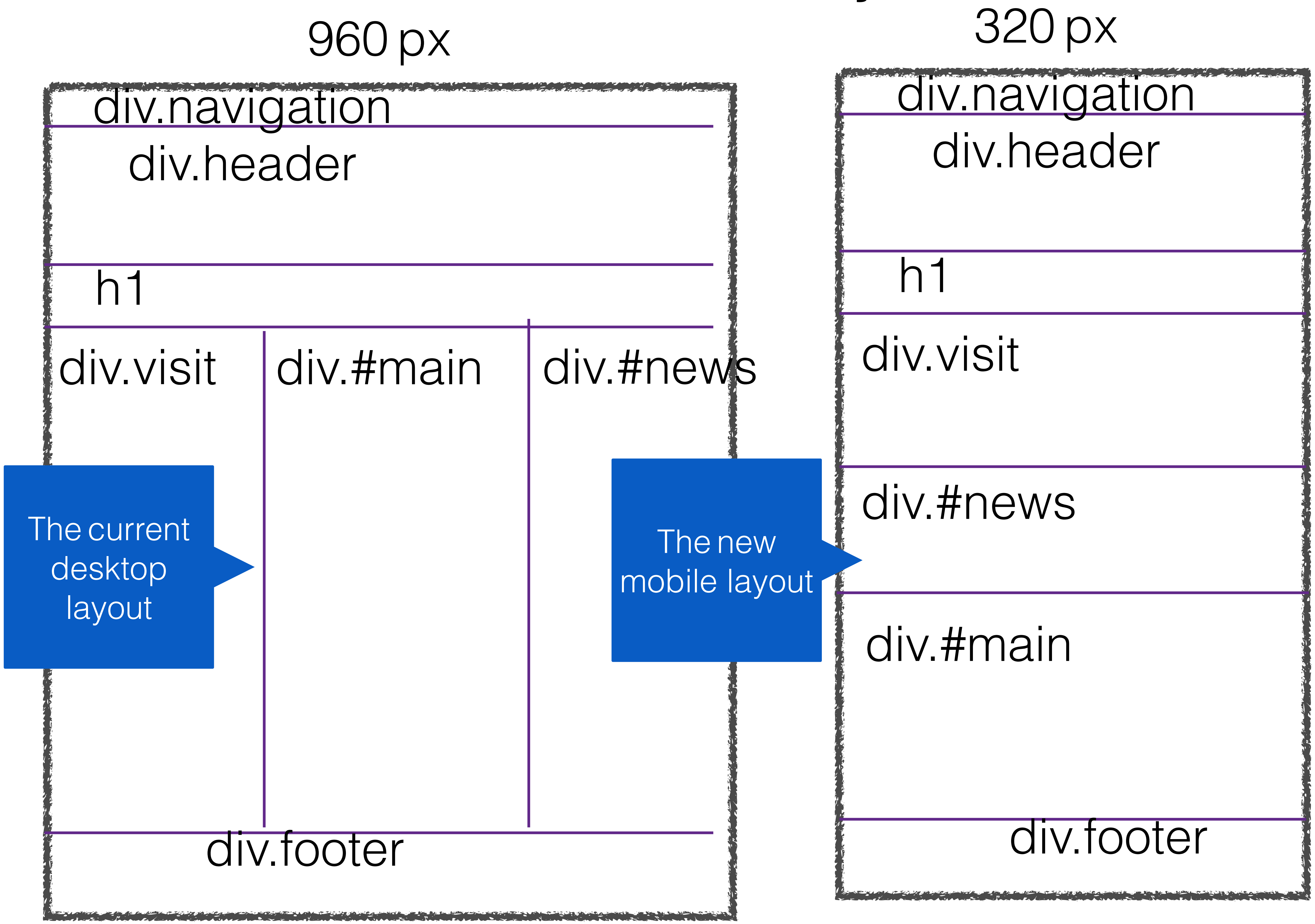
This current layout of the desktop version

Example of supporting mobile devices

```
body, .header, .navigation, .footer {
width: 960px; }
.header, .navigation, .footer { clear: both;
}
.column {
margin: 10px 10px 0 0;
}
.navigation { min-height: 25px;
}
.navigation ul li {
width: 320px;
}
.header {
background:url(images/x.png) no-repeat; height: 200px;
}
#visit {
width: 240px; float: left;
}
#news { width: 240px; float: right;
}
#main {
margin: 10px 260px 0 250px; width: 460px;
}
```

The current CSS file for the desktop version

What needs to modify?



Identify the CSS that needs to change

body, .header, .navigation, .footer {
width: 960px; }
.header, .navigation, .footer { clear: both;
}
.column {
margin: 10px 10px 0 0;
}
.navigation { min-height: 25px;
}
.navigation ul li {
width: 320px;
}
.header {
background:url(images/x.png) no-repeat; height: 200px;
}
#visit {
width: 240px; float: left;
}
#news { **width: 240px; float: right;**
}
#main {
margin: 10px 260px 0 250px; width: 460px;
}

The bolded CSS rules are needed to be changed for support the mobile layout

Steps to creating the mobile-specific CSS

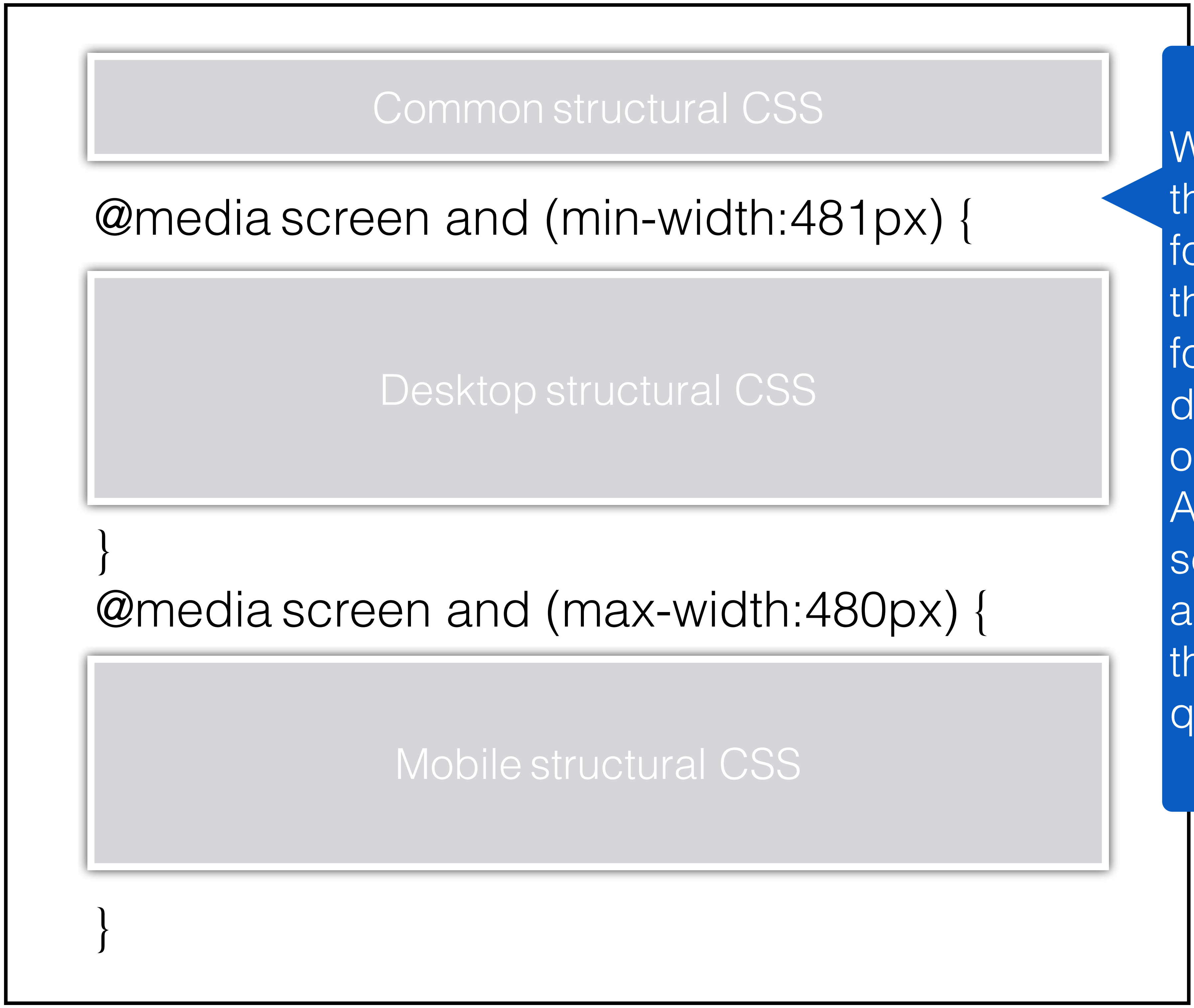
1. Change the width of the highlighted CSS rules.
2. Get rid of the CSS rules we don't need.
3. Factor out the common CSS rules.

```
body, .header, .footer, .navigation {
width: 320px;
}
.column {
margin: 10px 0;
border-bottom: 1px dashed #7b96bc; }
.navigation ul li {
width: 106.6667px;
}
#visit, #news, #main {
width:320px;
}
```

This new CSS rules are needed for the mobile layout

Restructure the CSS File

- Organize our CSS and selectively apply the mobile and desktop CSS using media queries.



```
Common structural CSS

@media screen and (min-width:481px) {

Desktop structural CSS

}
@media screen and (max-width:480px) {

Mobile structural CSS

}
```

We can combile the CSS rules for desktop and the CSS rules for mobile devices into one CSS file. And they will be selectively applied using the media querires

The modified CSS File

Common structural CSS

```
.header, .footer, .navigation {
clear: both;
}
.header {
background:url(images/x.png) no-repeat;
height: 200px;
}
.navigation {
min-height: 25px;
}
```

Desktop structural CSS

```
@media screen and (min-width:481px) {
            body, .header, .footer, .navigation {
            width: 960px;
            }
            .column {
            margin: 10px 10px 0 0;
            }
            .navigation ul li {
            width: 320px; /* 960/3 */
            }
            #visit {
            width: 240px;
            float: left;
            }
            #news {
            width: 240px;
            float: right;
            }
            #main {
            margin: 10px 260px 0 250px;
            }
}
```

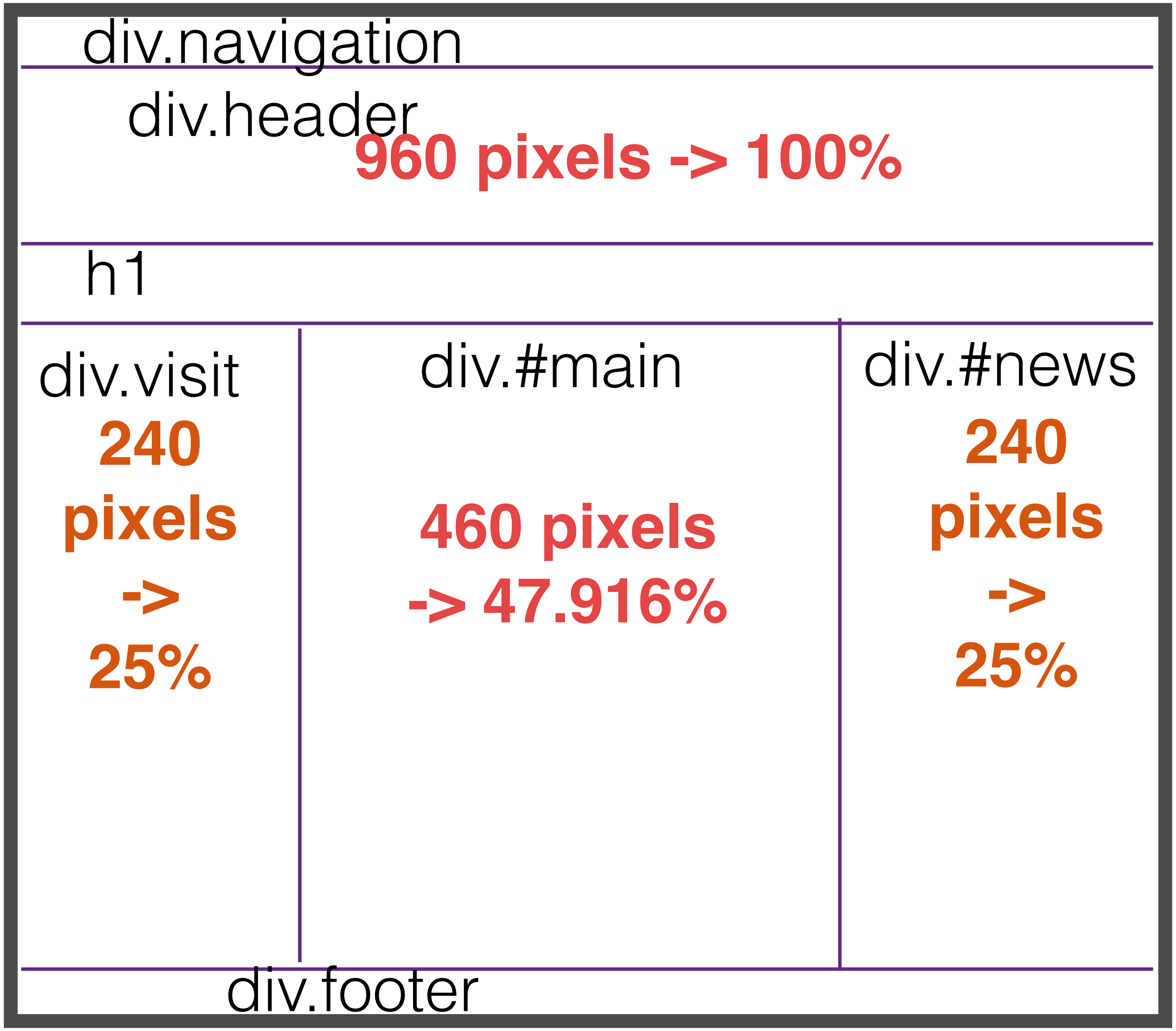
Mobile structural CSS

```
@media screen and (max-width:480px) {
            body, .header, .footer, .navigation {
            width: 320px;
            }
            .column {
            margin: 10px 0;
            border-bottom: 1px dashed #7b96bc; }
            .navigation ul li {
            width: 106.6667px;
            }
            #visit, #news, #main {
            width:320px;
            }
```

Fluid layout

- The Web browser window size has never been this controllable. Sometimes there is no such standard window widths like 640, 960, or 1,024 pixels. Even on mobile devices, the user can put it in landscape orientation, the width becomes wider.
- Fluid layouts use proportional units (percentages) instead of pixels for widths. For examples, we can specify the left and right columns span one-quarter of the page width by defining their widths as 25%, instead of 240 pixels.
- Things need to be done to support the fluid layout:
    - Convert the pixel-based layout to a fluid one, using proportional widths instead of fixed.
    - Make the default body font size 100% so our page's fontscan scale up and down proportionally.
    - Fix the image that is too wide.

Converting Pixels to Percentages



Converting Pixels to Percentages

```
@media screen and (min-width:481px) {
	body, .header, .footer, .navigation {
	width: 100%;
	}
	.column {
	margin: 10px 1.04166667% 0 0;
	}
	.navigation ul li {
	width: 33.333333%;
	}
	#visit {
	width: 25%;
	float: left;
	}
	#arrows {
	width: 25%;
	float: right;
	}
	#main {
	margin: 10px 27.0833333% 0 26.0416667%;
	}
}
@media screen and (max-width:480px) {
	body, .header, .footer, .navigation {
	width: 100%;
	}
	.column {
	margin: 10px 0;
	border-bottom: 1px dashed #7b96bc; }
	.navigation ul li {
	width: 33.333333%;
	}
	#visit, #arrows, #main {
	width:100%;
	}
}
```

Other Details

Fluid images

- They prevent any image or embedded media object from being wider than its containing element. Because they are limited to 100% width of their containing element so they don't try to break outside of the boundaries.

```
img, object {
max-width: 100%;
}
```

Flexible fonts

- By setting the baseline font size for the page to be 100%, so we can adapt our content to the user's environment. If a user has changed the browser's font size, 100% will represent a different font size.

```
body {
font: 100% "Georgia", "Times New Roman", serif;
}
```