RESPONSIVE WEB DESIGN

List materi :

- Apa itu RWD ?
- Kenapa RWD ?
- RWD dengan media query
- Viewport
- RWD media query magic
- RWD Choosing Breakpoints
- RWD min-max width and height
- Jenis RWD
- Latihan RWD
- Latihan Studi Kasus Navigasi Bar

What is RWD ?

"

Responsive Web Design (or RWD) is a design and production approach that allows a website to be comfortably viewed and used on all manner of devices. The core principle is that all devices get the same HTML source, located at the same URL, but different styles are applied based on the viewport size to rearrange components and optimize usability.

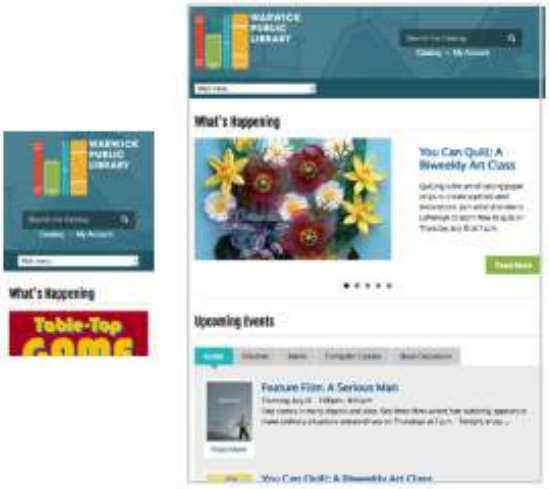
"

Jennifer Robbins, Learning Web Design 5th Edition

Why RWD ?

In 2016, mobile internet usage surpassed desktop usage—an important milestone. **The percentage of web traffic that comes from devices other than desktop browsers is steadily increasing.** For roughly 10% of Americans, a smartphone or tablet is their only access to the internet because of lack of access to a computer or high-speed WiFi at work or home.* Younger users may be mobile-only by choice. Furthermore, the vast majority of us access the web from a number of platforms (phone, tablet, computer) over the course of the day. And guess what—**we expect to have a similar experience using your content or service regardless of how we access your site.** That's where RWD fits in. With one source, you ensure that mobile visitors receive the same content as other visitors (although it might be organized differently)

Link Research : Pew Research Center, "Smartphone Use in 2015," www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/.



Warwick Public Library
warwicklibrary.org

Meta attribute “Smart Zoom”

Mobile browsers use so-called “smart zoom” to provide users with a ‘superior’ reading experience. Basically, smart zoom is used to proportionally reduce page size. This can manifest itself in two ways: (1) user-initiated zoom (for example, tapping twice on an iPhone screen to zoom in on the current website), and (2) initially displaying a zoomed-in version of a web page on load.

Given that we can just use responsive media queries to solve any of the problems at which smart zoom might be targeted, it’s often desirable (or even necessary) to disable zoom and ensure that your page content always fills the browser:

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

By setting `initial-scale` to 1, we control the initial page zoom level (that is, the amount of zoom upon page load). If you’ve designed your web page to be responsive, then your fluid, dynamic layout should fill the smartphone screen in an intelligent way without requiring any initial zoom.

This means that the browser will (probably) render the width of the page at the width of its own screen. So if that screen is 320px wide, the browser window will be 320px wide, rather than way zoomed out and showing 960px (or whatever that device does by default, in lieu of a responsive meta tag).

More information about meta tag :
https://www.w3schools.com/tags/tag_meta.asp

RWD Using media query

CSS media queries

Media queries give us a way to deliver sets of rules only to devices that meet certain criteria, such as width and orientation. Media queries apply different styles based on characteristics of the browser: its width, whether it is vertically or horizontally oriented, its resolution, and more.

```
@media type and (feature: value) {
   /* styles that browsers that meet this criteria */
}
```

The following media queries look at whether the viewport is on a screen and in landscape (horizontal) or portrait (vertical) orientation.

```
@media screen and (orientation: landscape) {
   body {
      background: skyblue;
   }
}
@media screen and (orientation: portrait) {
   body {
      background: coral;
   }
}
```

When the viewport is in portrait mode, the background color is “coral.”

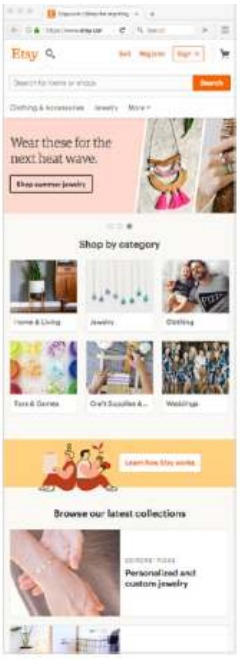
When the viewport is in landscape mode, the background color is “skyblue.”

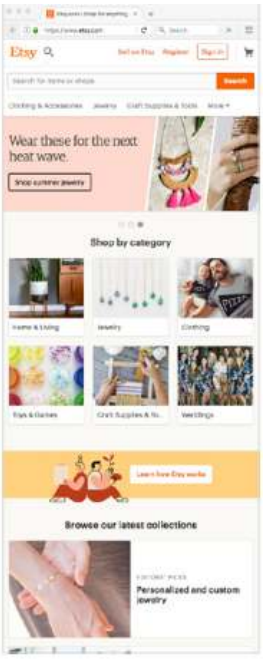
When the query detects that the viewport is in landscape mode, the background color of the page is “skyblue”; when it is in portrait orientation, the

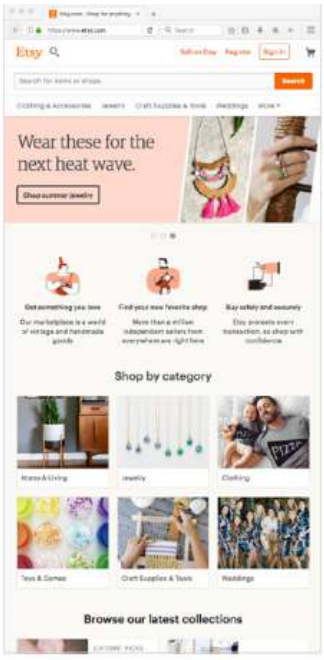
background is “coral” If this were displayed on a smartphone that tips from vertical to horizontal and back again, the colors would change as it tilts. This isn’t a very practical design choice, but it does provide a very simple illustration of media queries at work.

RWD Choosing Breakpoints

A breakpoint is the point at which we use a media query to introduce a style change. When you specify min-width: 800px in a media query, you are saying that 800 pixels is the “breakpoint” at which those particular styles should beused.



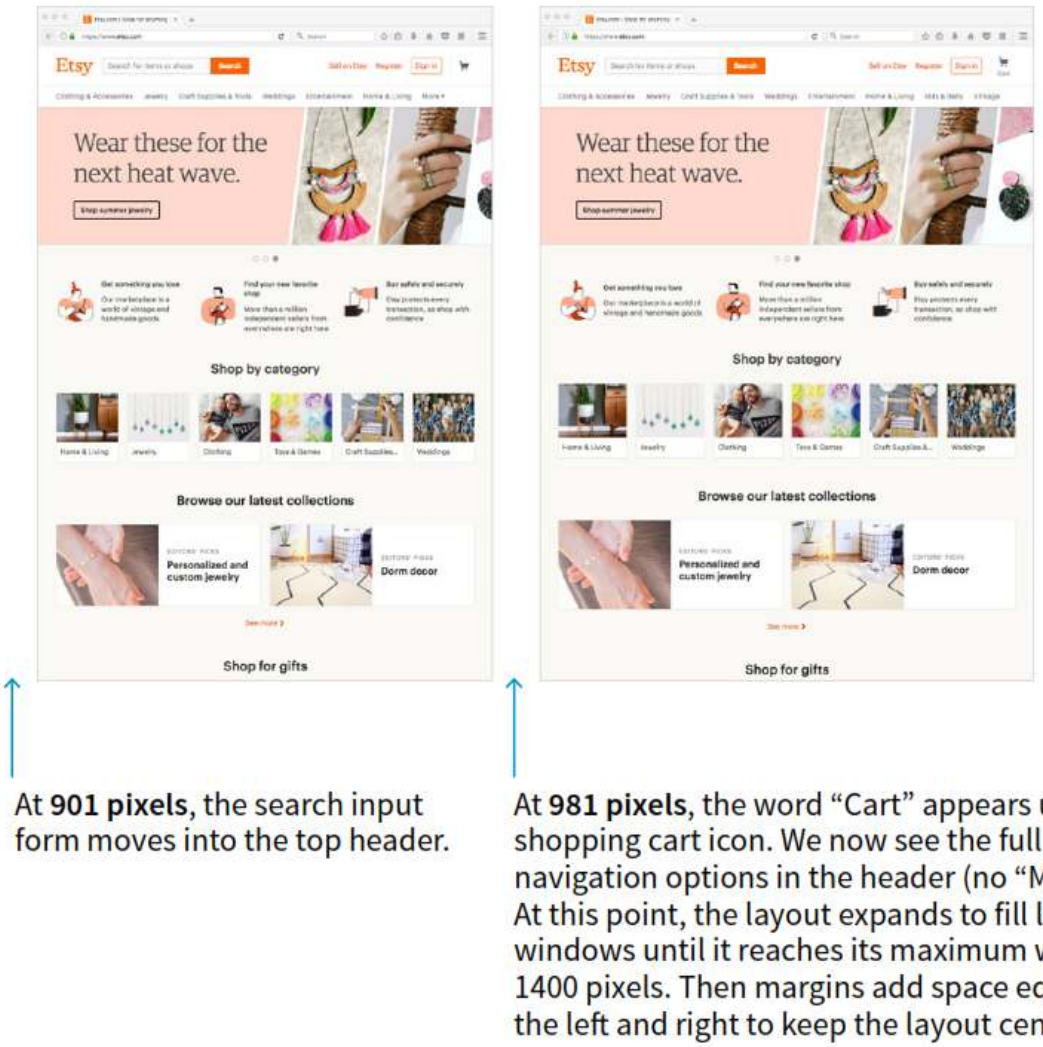




At the **480-pixel breakpoint**, the category navigation changes from a list to photos. “Register” is added to the top navigation bar.

At **501 pixels**, “Sell” becomes “Sell on Etsy” (a very subtle adjustment). You can also see more links in the navigation bar under the search field.

At **640 pixels**, the “How Etsy Works” images and messages move above the categories. In smaller views, they were accessible via the “Learn how Etsy works” link in a yellow bar.



At **901 pixels**, the search input form moves into the top header.

At **981 pixels**, the word “Cart” appears unde shopping cart icon. We now see the full list o navigation options in the header (no “More” At this point, the layout expands to fill larger windows until it reaches its maximum width 1400 pixels. Then margins add space equally the left and right to keep the layout centered

The difference

In short, the **main difference** between the two is the **condition** when the styles will be applied:

`@media (min-width: 800px) { ... }` - for browser’s viewport width equal or wider than 800px;

`@media (max-width: 800px) { ... }` - for browser's viewport width equal or narrower than 800px.

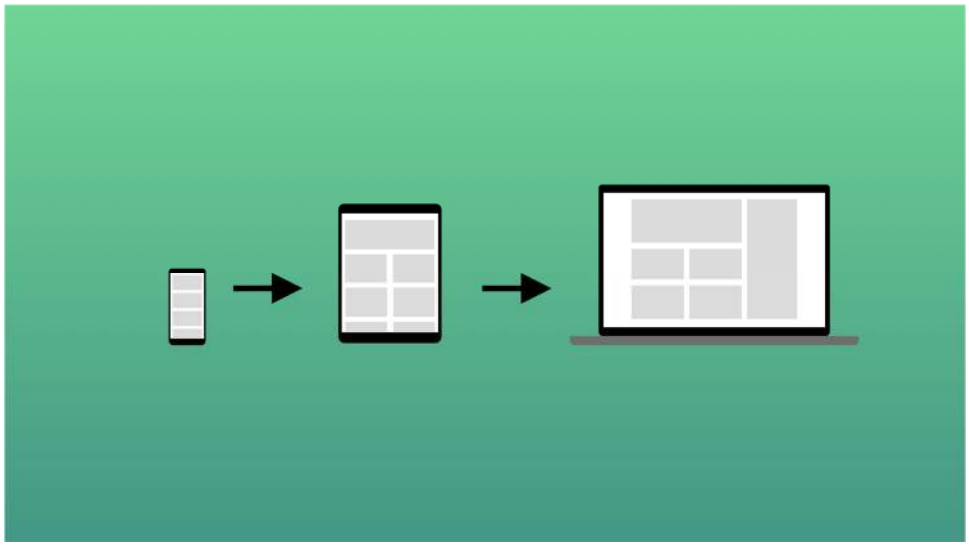
If you're starting a new project and have to choose between `min-width` and `max-width` media queries, that means you're choosing between two approaches:

- Mobile-first;
- Desktop-first.

Mobile-first approach

The mobile-first approach means that you design/develop styling for your app starting from a mobile device going all the way up to a desktop computer and even TVs. For every following device, the breakpoint is specified with the media query `min-width` rule.

The mobile-first approach is considered a good practice, as mobile user count takes more than 50% of the market.



Mobile-first approach with min-width property

In the following example, the container width will be 100% on viewports from 0 up to 768px. All viewports above 768px will have a container width of 80%, unless you specify it with another rule with a higher width value (breakpoint).

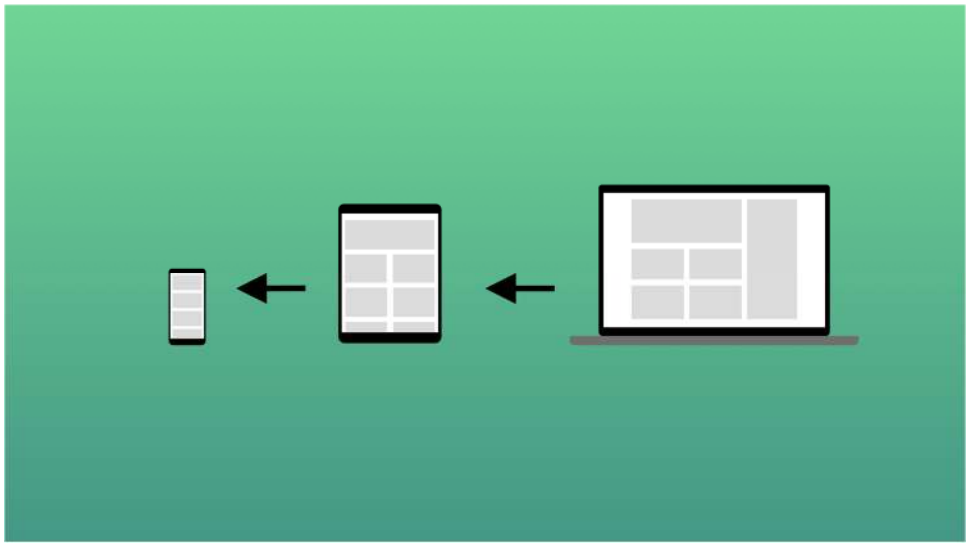
The min-width example:

```
.container{
width:100%;
padding:0 20px;
}
```

```
@media(min-width:768px)
{
.container{width:80%;}
}
```

Desktop-first approach

Unlike the mobile-first approach, the desktop-first means that you apply styles to the large viewports first (like the desktop computers), and then specify rules to target smaller viewports.



Desktop-first approach with max-width property

In the following example, the container width will be 80% on all viewports down to 768px. All viewports below 768px will have a container width of 100% and a side paddings of 20px.

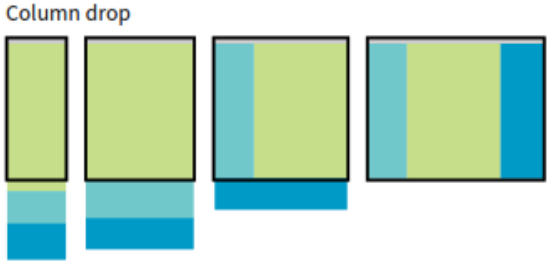
With the desktop-first approach, we're essentially achieving the same result as with mobile-first.

The max-width example:

```
.container{
width:80%;
}
@media(max-width:768px){
.container{
width:100%;
padding:020px;
        }
}
```

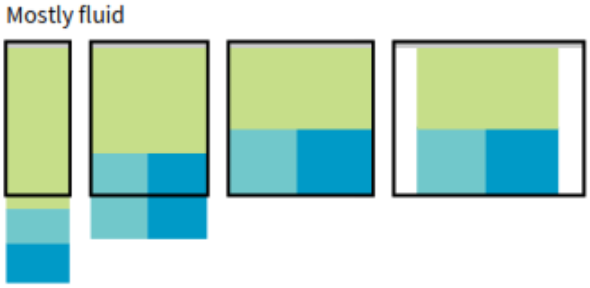
Jenis RWD

Column Drop (https://codepen.io/bradfrost/full/neYPVY)



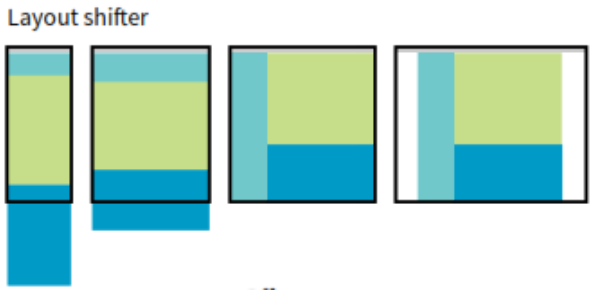
This solution shifts between one-, two-, and three-column layouts based on available space. When there isn't room for extra columns, the sidebar columns drop below the other columns until everything is stacked vertically in the one-column view.

Mostly Fluid (https://codepen.io/bradfrost/full/kyLRpd)



This pattern uses a single-column layout for small screens, and another fluid layout that covers medium and large screens, with a maximum width set to prevent it from becoming too wide. It generally requires less work than other solutions

Layout Shifter (https://codepen.io/bradfrost/full/kXWxZb)



If you want to get really fancy, you can completely reinvent the layout for a variety of screen sizes. Although expressive and potentially cool, it is not necessary. In general, you can solve the problem of fitting your content to multiple environments without going overboard

Daftar Referensi :

Introduction to Responsive Web Design :

From <https://www.toptal.com/responsive-web/introduction-to-responsive-web-design-pseudo-elements-media-queries>

Viewport Concepts :
https://developer.mozilla.org/en-US/docs/Web/CSS/Viewport_concepts#mobile_viewports