META
+LAB

META+LAB PRESENTS:

Responsive Web Design (RWD)

Front-End Immersive 2018

Responsive web design (RWD) is the practice of ensuring that your website **adapts to the device** your users are viewing it on, **regardless of screen size**.

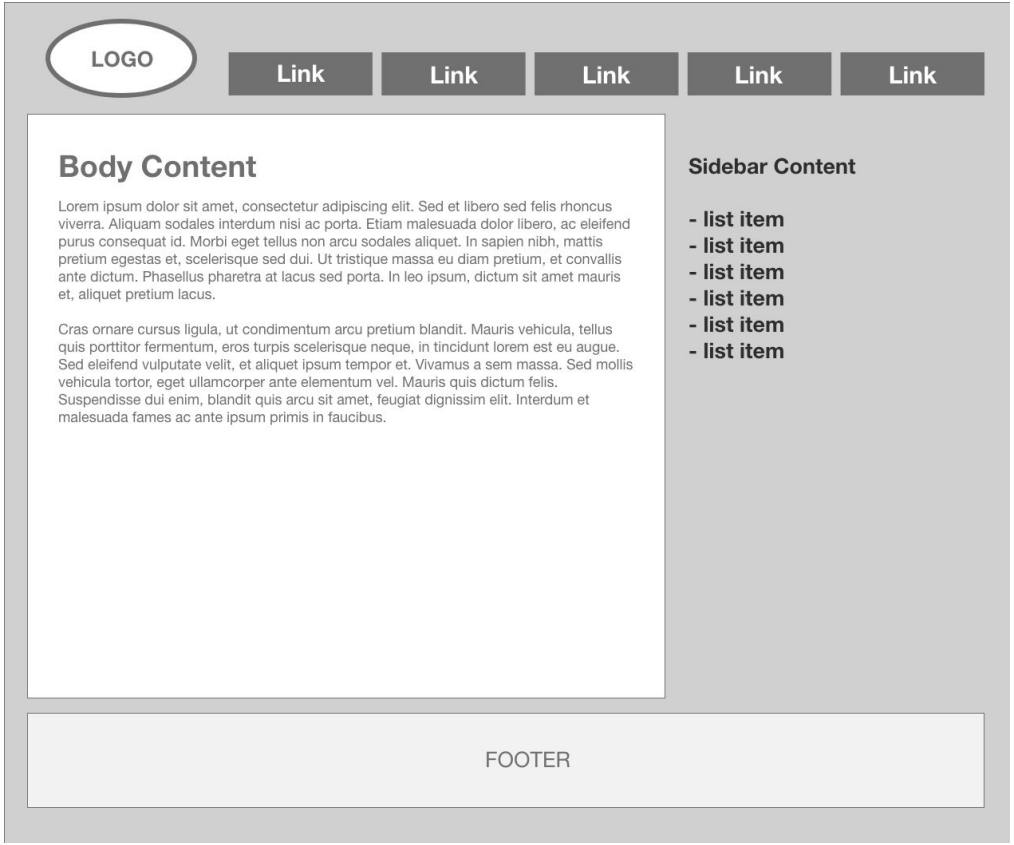
Your website should be usable and readable on all devices, including:

- TVs
- Large Monitors
- Laptops
- Tablets
- Phablets
- Phones

Responsive Web Design (RWD)

Most websites are usually made up of the following components:

- Header
- Logo
- Navigation
- Body Content
- Sidebar Content
- Footer

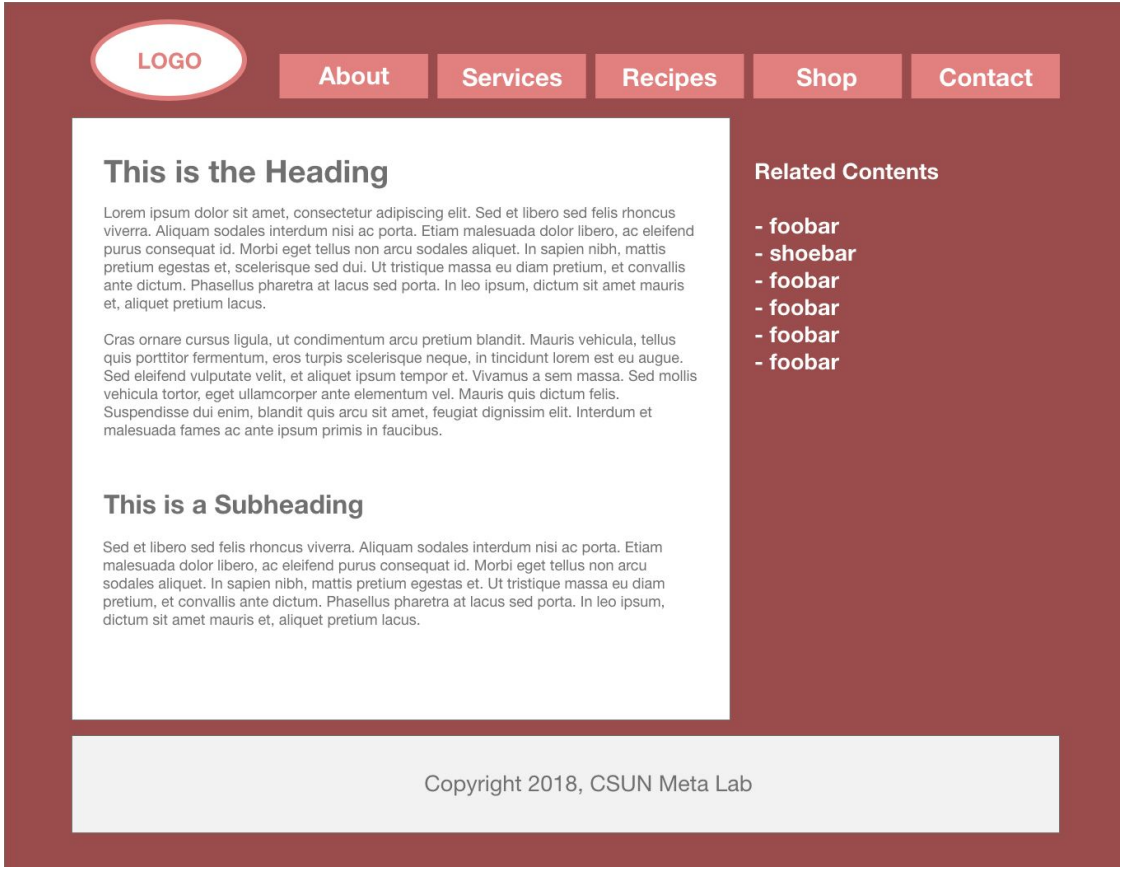


META +LAB

Can you build this website?

Container: 1200px wide

- Header: 1200px wide
  - Logo: 200px wide
  - Nav: 1000px wide
    - Nav links: 180px wide
- Body Content: 900px wide
- Sidebar: 300px wide
- Footer: 1200px wide



META +LAB

Responsive Web Design (RWD)

https://codepen.io/andrewMETALAB/full/KRbgEm/

META
+LAB

Good job. Your website looks great on a desktop, but not on smaller screens.

How do we make it responsive?

RWD is achieved with 3 steps:

1. Flexible layout
2. Flexible Media
3. Media Queries

But first, add this line of code inside the <head> of your webpage:

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

This resets the viewport width to be the size of the device itself and it sets the scale to 1 (which means it’s not zoomed in or out).

Flexible Layout

Let’s make our website flexible.

Instead of using **pixels**, we will use **percentages**.

To determine the correct percentage value for each layout element *(header, body, sidebar, footer, etc)*, we use this formula:

Divide the target by its context

for each PX value we’re going to take the value, then work out the parent contain size (the context), and divide

Responsive Web Design (RWD)

Flexible Layout

```
.container {
  width: 1200px;
  margin-left: auto;
  margin-right: auto;
}
```

original

The container has no parent, so we will set the width to 100%. We also set a max-width of 1200px so that the website design remains as-is on large viewports instead of stretching farther.

```
.container {
  width: 100%;
  max-width: 1200px;
  margin-left: auto;
  margin-right: auto;
}
```

flexible

META +LAB

Responsive Web Design (RWD)

Flexible Layout

```
.header {
  width: 1200px;
  height: 100px;
  float: left;
}
```

original

The header has a parent, the container. So let's do the math. 1200 / 1200 = 1 * 100 = 100%

```
.header {
  width: 100%;
  float: left;
  height: 100px;
}
```

flexible

META +LAB

Responsive Web Design (RWD)

Flexible Layout

```
.logo {
  width: 200px;
  float: left;
}
```

original

The logo is 200px wide and it's parent is the header (which is 1200px wide). So we will do 200 / 1200 = .166 * 100 = 16.66%

```
.logo {
  width: 16.66%;
  float: left;
}
```

flexible

META +LAB

Responsive Web Design (RWD)

Flexible Layout

```
.nav {
  width: 1000px;
  float: left;
}
```

original

The nav is 1000px wide and it's parent is the header (which is 1200px wide). So we will do 1000 / 1200 = .8333 * 100 = 83.33%

```
.nav {
  width: 83.33%;
  float: left;
}
```

flexible

META +LAB

Responsive Web Design (RWD)

Flexible Layout

```
.nav ul li {
  width: 180px;
  margin-right: 10px;
  float: right;
}
```

original

The <li> element is 180px wide and it's parent is the nav (which is 1000px wide). So we will do 180 / 1000 = .18 * 100 = 18%

For the margin, we will do 10 / 1000 = .01 * 100 = 1%

```
.nav ul li {
  width: 18%;
  margin-right: 1%;
  float: right;
}
```

flexible

META +LAB

Flexible Layout

```
.nav ul li a {
  width: 180px;
  padding: 10px;
  color: white;
  text-decoration: none;
  background: #E27F7F;
  display: inline-block;
  text-align: center;
  font-size: 20px;
}
```

original

```
.nav ul li a {
  width: 100%;
  padding: 5.55%;
  color: white;
  text-decoration: none;
  background: #E27F7F;
  display: inline-block;
  text-align: center;
  font-size: 20px;
}
```

flexible

The <a> element is 180px wide and it's parent is the <li> element (which is 180px wide). So we will do 180 / 180 = 1 * 100 = 100%

For the padding, we will do 10 / 180 = .0555 * 100 = 5.55%

Responsive Web Design (RWD)

Flexible Layout

```
.content {
  width: 900px;
  padding: 30px;
  float: left;
  background: white;
}
```

original

```
.content {
  width: 75%;
  padding: 2.5%;
  float: left;
  background: white;
}
```

flexible

The .content is 900px wide and it's parent is the .container (which is 1200 wide). So we will do 900 / 1200 = 0.75 * 100 = 75%

For the padding, we will do 30 / 1200 = .025 * 100 = 2.5%

META +LAB

Flexible Layout

```
.sidebar {
  width: 300px;
  padding: 20px;
  float: left;
  color: white;
}
```

original

```
.sidebar {
  width: 25%;
  padding: 1.66%;
  float: left;
  color: white;
}
```

flexible

The .sidebar is 300px wide and it's parent is the .container (which is 1200 wide). So we will do 300 / 1200 = 0.25 * 100 = 25%

For the padding, we will do 20 / 1200 = .01666 * 100 = 1.66%

Responsive Web Design (RWD)

Flexible Layout

```
.footer {
  width: 1200px;
  padding: 30px;
  margin-top: 20px;
  margin-bottom: 20px;
  float: left;
  background: #F2F2F2;
  text-align: center;
}
```

original

```
.footer {
  width: 100%;
  padding: 2.5%;
  margin-top: 1.66%;
  margin-bottom: 1.66%;
  float: left;
  background: #F2F2F2;
  text-align: center;
}
```

flexible

The header has a parent, the container. So let's do the math. 1200 / 1200 = 1 * 100 = 100%

For the padding, we will do 30 / 1200 = .025 * 100 = 2.5%

For the margin, we will do 20 / 1200 = .0166 * 100 = 1.66%

META +LAB

Responsive Web Design (RWD)

Flexible Layout

https://codepen.io/andrewMETALAB/full/GdPmMq/

META
+LAB

Flexible Layout

So we now have a flexible layout. *(Step 1 is complete!)*

But what about images?

Our next step is making our media flexible.

Flexible Media

By “media”, we are referring to:

Images — <img>
Iframes — <iframe>
Video — <video>

For this demonstration, let’s focus on images.

Responsive Web Design (RWD)

Flexible Media

If we add an image to our body, it remains the same width, regardless of screen size. We want to get the image to become flexible within the container itself.

To achieve this, let's apply the following CSS to our image:

```
.img-responsive {
  max-width: 100%;
  height: auto;
}
```

These CSS rules ensure that the image will not grow beyond its own width or the width of the container, while maintaining it's ratio (not stretching).

META +LAB

Responsive Web Design (RWD)

Flexible Media

https://codepen.io/andrewMETALAB/full/JvwNvg/

META
+LAB

Responsive Web Design (RWD)

Media Queries

Ok, now we have a flexible layout *(step 1 complete)* and our media is flexible *(step 2 complete)*. But our website still doesn't look very good on small screens.

Let's take a look at how **media queries** will help us with that.

CSS Media queries allow you to conditionally apply CSS styles based on screen width.

META +LAB

Responsive Web Design (RWD)

Media Queries

```
@media (max-width: 600px ) {
  body {
    background: red;
  }
}
```

This code will make the background of the page red ONLY when the screen width is 600px or **smaller**

```
@media (min-width: 600px ) {
  body {
    background: red;
  }
}
```

This code will make the background of the page red ONLY when the screen width is 600px or **wider**

META +LAB

Responsive Web Design (RWD)

Media Queries

```
@media (max-width: 800px) and (min-width: 500px) {
    body {
        background: red;
  }
}
```

This code will make the background of the page red ONLY when the screen width is **between** 500px and 800px.

META +LAB

Responsive Web Design (RWD)

Media Queries

https://codepen.io/andrewMETALAB/pen/mLQqVp

META
+LAB

Media Queries

Note:

You can also use media queries to target **print** (websites in print-preview mode) and **speech** (speech synthesizers). You can target not only the screen **width**, but also the **height** as well as other characteristics such as **orientation** and **resolution**.

Responsive Web Design (RWD)

Media Queries

Now let's use media queries to make our website layout change on smaller viewports

```
@media (max-width: 767px ) {
  .header {
    height: auto;
  }

  .logo,
  .nav {
   width: 100%;
  }

  .logo img {
    display: block;
    margin-left: auto;
    margin-right: auto;
  }

  .nav ul {
    margin-bottom: 2%;
  }
}
```

```
@media (max-width: 767px ) {
  .nav ul li {
    width: 100%;
    display: block;
    float: none;
  }

  .nav ul li a {
    background: none;
    padding: 2% 0;
  }

  .content,
  .sidebar{
    width: 100%;
  }

  .sidebar {
    background: #F2F2F2;
    color: #707070;
  }

  .footer {
    background: none;
    color: white;
  }
}
```

META +LAB

Responsive Web Design (RWD)

Media Queries

https://codepen.io/andrewMETALAB/full/pVqwoQ/

META
+LAB

We have now built a responsive website using:

- A Flexible layout
- Flexible media
- Media queries

Good job!

What about websites with more complex designs and layouts?

https://mediaqueri.es/

The developers who built these sites utilized the same techniques that we just learned to make the websites responsive. However, due to the complexity of the sites, the front-end developers probably didn't code everything from scratch…

Instead, they probably used a **mobile-first approach** and a **framework** to make their jobs easier.

Responsive Web Design (RWD)

Mobile First

“Mobile-first” is an approach to designing and building websites.

In the United States:

- Over 40% of all web traffic is on a **mobile** device
- About 8% is on a **tablet**
- 51% is on a **desktop** or **laptop**

Worldwide:

- Over 51% of all web traffic is on a **mobile** device
- About 4% is on a **tablet**
- About 44% is on a **desktop** or **laptop**

META +LAB

Mobile First

These statistics have convinced many web designers & developers to take a different approach to building websites. Instead of the traditional approach of:

1. Design mockups of the website for a **desktop**
2. Build the website for **desktop**
3. Use media queries to make the website look good on **smaller** screens

Mobile-first, instead, consists of the opposite strategy:

1. Design mockups of the website for a **mobile** device
2. Build the website for **mobile**
3. Use media queries to make the website look good on **larger** screens

Responsive Web Design (RWD)

Mobile First

Why does it matter!? These two approaches seem to accomplish the same thing...

When you start with the desktop, you tend to take advantage of everything that platform has to offer. This can lead to watered down mobile products that feel more like an afterthought than a polished, finished product.

Alternatively, when you start with mobile, you've already gone through the problem of trimming down the content to its most vital elements. You've created a product that looks and functions well despite the restraints of mobile. Now when it's time to bring this design to the desktop, you get to focus on how to make the product more robust instead of less robust.

META +LAB

CSS Frameworks

CSS frameworks are a collection of code (HTML, CSS, and Javascript) that you can add to your website.

The framework provides a “starting off” point for your website. The code included in a framework aims to tackle common recurring issues that you will encounter when building a website. The generic functionality that frameworks offer can be overridden and tweaked as you see fit.

Frameworks make it easier and faster to build a website.

CSS Frameworks

Frameworks usually consist of:

- **CSS Resets**
  - necessary to normalize browser default styles
- **Typography settings**
  - ensure appealing typographic rhythms by setting default fonts, sizes, line-heights, and headings
- **Color settings**
  - set the color of links, buttons, etc
- **Grid System**
  - used for creating page layouts through a series of rows and columns that house your content.
- **Reusable components/UI elements**
  - could be anything from navbars and menus to modals, carousels, and tooltips.

CSS Frameworks

There are **many** different CSS frameworks available, including:

- Bootstrap (getbootstrap.com)
- Foundation (foundation.zurb.com)
- Materialize (materializecss.com)
- Pure.css (purecss.io)
- Bourbon (neat.bourbon.io)
- Base (basscss.com)
- And many more

CSS Frameworks - Bootstrap

Let's explore **Bootstrap**, one of the most popular CSS frameworks.

Bootstrap was created by 1 designer and 1 developer at Twitter in 2010, originally serving as the internal style guide for the company. It has now become one of the most popular front-end frameworks and open source projects in the world.

- Version 1 was released in 2011.
- Version 2 added responsive functionality to the entire framework as an optional stylesheet.
- Version 3 made it responsive by default with a mobile first approach.
- Version 4 was released in 2018, accounting for two key architectural changes: a migration to Sass and the move to CSS's flexbox.

Responsive Web Design (RWD)

CSS Frameworks - Bootstrap

When you want to learn about a new CSS Framework or other front-end tool…

Always read the docs!

https://getbootstrap.com/docs/4.1/getting-started/introduction/

META +LAB

CSS Frameworks - Bootstrap

Frameworks usually consist of:

- **CSS Resets**
  - https://getbootstrap.com/docs/4.1/getting-started/introduction/#reboot
- **Typography settings**
  - https://getbootstrap.com/docs/4.1/content/typography/
- **Color settings**
  - https://getbootstrap.com/docs/4.1/utilities/colors/
- **Grid System**
  - https://getbootstrap.com/docs/4.1/layout/overview/#responsive-breakpoints
  - https://getbootstrap.com/docs/4.1/layout/grid/
  - https://getbootstrap.com/docs/4.1/examples/grid/ (example)
  - https://getbootstrap.com/docs/4.1/examples/carousel/ (example)
- **Reusable components/UI elements**
  - (next slide)

CSS Frameworks - Bootstrap

Reusable components/UI elements

- Buttons
  - https://getbootstrap.com/docs/4.1/components/buttons/
- Card
  - https://getbootstrap.com/docs/4.1/components/card/
- Carousel
  - https://getbootstrap.com/docs/4.1/components/carousel/
- Modal
  - https://getbootstrap.com/docs/4.1/components/modal/#live-demo
- Navbar
  - https://getbootstrap.com/docs/4.1/components/navbar/#supported-content

CSS Frameworks - Bootstrap

Themes

https://themes.getbootstrap.com/

The Bootstrap Theme library offers many different themes for sale - all designed, built, and supported by the Bootstrap Team.

Each theme was designed as its own extended version of Bootstrap, built for a specific set of problems. Not only have they extended many parts of Bootstrap, but also introduced dozens of completely new utilities, components, and plugins.

Every component and plugin is thoroughly documented with live examples and code blocks for easier use and customization—just like Bootstrap itself.

CSS Frameworks - Bootstrap

Let’s adjust our responsive website to use the Bootstrap grid

- Add the Bootstrap stylesheet <link> inside the <head> of your document
- Remove “float” and “width” CSS rules that were originally setting the layout (including those CSS rules inside of media queries)
- Update the HTML to use the grid classes
    - Wrap everything inside a with the class of “.container”
    - Wrap the logo in a with a class of “.col-md-3”
    - Wrap the nav in a with a class of “.col-md-9”
    - Make sure any s with a class of “.col-xx-x” have a direct parent with a class of “.row”
    - Continue in this fashion with the rest of the layout

Responsive Web Design (RWD)

CSS Frameworks - Bootstrap

https://codepen.io/andrewMETALAB/full/vjMXad/

META
+LAB

CSS Frameworks - Bootstrap

You now have a responsive website that uses the Bootstrap grid.

Your website can also take advantage of the many other CSS utilities and components that Bootstrap offers.

Any questions?

THAT'S A WRAP FOLKS,

THANK YOU!

META
+LAB