We create innovate software products that appeal to global audiences

Hackathon 2014

Responsive Web Design

Octubre 2014

Globant
we are ready

What is RWD?

Responsive Web Design is a growing movement in the web design community, but it's still at the early stages of its development, and many of the concepts and techniques behind it are still evolving.
This session was created as a way of introducing what responsive design is and what you should focus on if you are chosen to begin building responsive UI.
We are going to share and discuss during this course different techniques, concepts, strategies about responsive design.

According to Ethan Marcotte

*"RWD is a **flexible grid** (**with flexible images**) that incorporates **media queries** to create a responsive, adaptive layout."*

What is RWD?

Group of techniques and strategies to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from desktop computer monitors to mobile phones).

When we create content for the web, we have to deal with several set of variables in how that content will be consumed.
We have to consider the wide of variety of browsers and OS that people might be using, as the many devices as well like: Desktop, Mobile, Tablet, Old Mobile.

One of the most important thing that we have to consider is the screen size. Screen sizes are not only getting smaller, they're getting bigger as well. That means you always have to show your content considering all the environment that different devices offers eg: OS, Browsers, Screen Size.

What is RWD?

There are 3 fundamentals techniques:

* Media queries

* Fluid grid

* Flexible media

What is RWD?

* Media queries

Sizing in proportions
target % context = result
24px % 16px = 1.5em

* Fluid grid

target % context = result
/* 700px / 988px = 0.7085 */
h1 {width:70.85%;}

* Flexible media

img {max-width:100%;}
media queries breakpoint

What is RWD?

Needs for Responsive Web Design

In the past, we've had a pretty good handle on what the target screen resolution for our design should be. However, over the last five years **we've seen some dramatic changes in how sites are viewed**. Some of those changes were relatively minor, like the aspect ratio of monitors changing from 4 X 3 to 16 X 9. Other changes, like t**he rise of mobile devices, will fundamentally change how we design sites**. Majority of your viewers would use a specific screen size. That's not the case anymore. The time and energy required to target each one of these different screens isn't even worth thinking about. That's what makes responsive design so important.

What is RWD?

What makes responsive?

Clean, well-structured HTML helps create content that is meaningful and easier to manipulate based on context.

HTML5 also includes a few features that helps sites be more functional across multiple devices. As you would imagine, **CSS plays a major role in creating responsive design**. **CSS media queries allow us to apply different sets of styles based on factors such as screen size, orientation, or resolution.** This allows us to change layout and typography to a design that's more suited to the context in which it's viewed. Responsive layouts are typically fluid in nature so that they can flex to fit the screen on which they're viewed.

Newer CSS features, such as transitions and transforms, allow designers to change how content displays and interacts with the user without requiring additional scripting. **CSS-based graphics can also help reduce the amount of resources requested by the page by replacing icons or other images.**

It's primarily used to control resource loading, s**upply the correct images and media based on the needs of different devices**, and to add device-specific functionality. Essentially, **these three technologies--HTML, CSS, and JavaScript--form the backbone of RWD**.

What is RWD?

Exploring a responsive site

http://ethanmarcotte.com

http://www.bostonglobe.com/

http://www.smashingmagazine.com/

Mobile Viewport

Mobile Safari introduced the "viewport meta tag" to let web developers control the viewport's size and scale. Many other mobile browsers now support this tag.

A typical mobile-optimized site contains something like the following:

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
```

The width property controls the size of the viewport. It can be set to a specific number of pixels like width=600 or to the special value device-width value which is the width of the screen in CSS pixels at a scale of 100% The initial-scale property controls the zoom level when the page is first loaded. The maximum-scale, minimum-scale, and user-scalable properties control how users are allowed to zoom the page in or out. Currently, there are two mechanisms for overriding the user agent's default viewport, One inside the head tag like above and the other by CSS media queries. If you choose the css viewport rule It should be placed before media queries.

Mobile Viewport

Another way to declare the viewport

```
/* CSS Document */
@viewport {
        width: device-width;
        zoom: 1;
        min-zoom:1;
        max-zoom:2;
        user-zoom:fixed;
}
```

Screen Densities

Screen density is measured by the number of pixels within a specified space. For example, 326ppi means there are 326 pixels within every inch of a display. In addition to the increased diversity of screen sizes, you'll also need to deal with the growing issue of screen densities.

Newer screens are featuring higher and higher screen densities every year.
The same image could be shown on different sizes depending on which device we are using. eg: lower density device or high density device.

eg: https://github.com/h5bp/mobile-boilerplate/wiki/Mobile-Matrices

We can apply a scaling factor to the page elements. **This scaling factor is often referred to as device-pixel-ratio** that is, the ratio between reference pixels and hardware pixels.
Knowing the proper ratio **is crucial for targeting and properly preparing graphics for higher-density screens.**

Screen Densities

You can targeting device pixel ratio with media queries:

```
@media screen and (device-pixel-ratio:2) {
        .high-res {
                background-image: url(img/high-res.jpg);
        }
}

@media screen and (device-pixel-ratio: 1.5)
   and (device-width: 683px)
   and (orientation: landscape),
 screen and (device-pixel-ratio: 1.5)
   and (device-width: 400px)
   and (orientation: portrait)
```

Understanding Media Queries

Media queries are an important part of responsive design, as they **allow you to write styles that respond to changes in things like screen size or orientation.** They're not really a new feature of CSS, but rather, an extension of the existing media declarations.

```
<link rel="stylesheet" type="text/css" href="core.css" media="screen" />
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

Media queries were created to allow designers to extend the media declarations to include various media properties when filtering style application. This is actually incredibly more useful, as it allows designers to control styling based on factors such as screen size, orientation, or color, which is far more flexible than simply defining a device type.

Media queries contain a media type and one or more expressions. **These expressions contain media features**, which are then evaluated and used to determine whether styles are applied or not.

Understanding Media Queries

The query contains two components:

1. a media type (screen), and
2. the actual query enclosed within parentheses, containing a particular media feature(max-device-width) to inspect, followed by the target value (480px).

```
<link rel="stylesheet" type="text/css" media="screen and (max-device width: 480px)" href="shetland.css" />
```

Also you can test multiple property values in a single query by chaining them together with the and keyword:

```
<link rel="stylesheet" type="text/css" media="screen and (max-devicewidth: 480px) and (resolution: 163dpi)" href="
shetland.css" />
```

Understanding Media Queries

We're not limited to incorporating media queries in our links. We can include them in our CSS either as part of a **@media rule**:

```
@media screen and (max-device-width: 480px) {
        .column {
                float: none;
        }
}
```

Media Features:

1. width - 2. height - 3. device-width - 4. device-height - 5. orientation.
6. Aspect-ratio - 7. device-aspect-ratio - 8. color - 9. color-index
10. monochrome - 11. resolution - 12. scan - 13. grid

Creating Media Queries Breakpoint

A breakpoint indicates the moment your layout switches from one layout to another, and is usually triggered by the width of a screen.
The first things that you need to do when planning a responsive design is define the breakpoints for your layout.

Here you see a list of three of the most common breakpoints:

```
@media only screen and (max-width: 480px) {
        //mobile styles go here
}

@media only screen and (min-width: 481px) and (max-width: 768px) {
        //tablet styles go here
}

@media only screen and (min-width: 769px) {
        //desktop styles go here
}
```

Hackathon 2014

Run Examples

Octubre 2014

Globant
we are ready

Using Fluid Grids

A fluid grid by definition is flexible and response to the width of the screen as much as possible.
Ethan Marcotte who really codified the idea of fluid grids with his article that appeared in "A list apart".
eg: http://www.alistapart.com/articles/fluidgrids/

Fluid grids have a basic proportional **element called "em"**. The "em" is used to express font size in relation to the browser standard font size.

The standard size of basic text in most browsers is 16 pixels (context)
The formula to calculate the width of the elements is:

target ÷ context = result

Using Fluid Grids

If we assume the body's default type size to be 16px, we can plug each desired font-size value into this formula. So to properly match our header to the comp, we **divide the target value (24px) by the font-size of its container (16px)**:

target ÷ context = result

24px ÷ 16px = 1.5

So the header is 1.5 times the default body size, or 1.5em, which we can plug directly into our stylesheet.

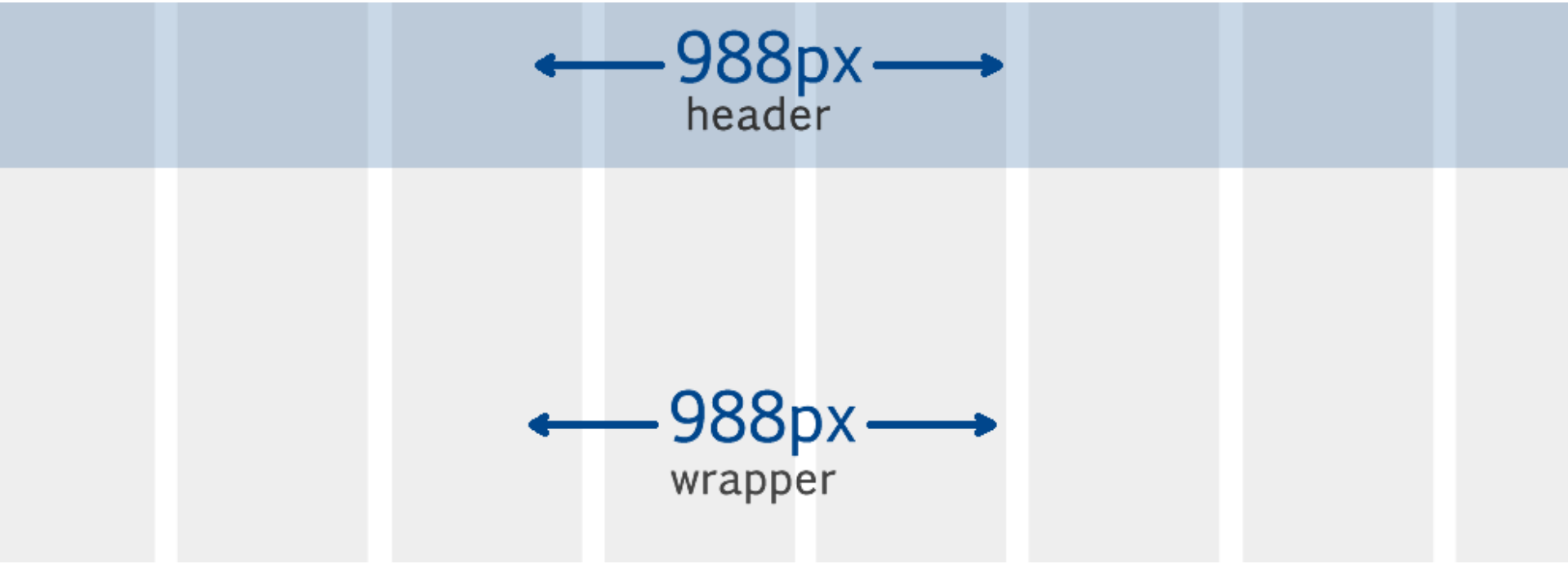
```
h1 {
   font-family: Georgia, serif;
   font-size: 1.5em;       /* 24px / 16px = 1.5em */
}
```

Using Fluid Grids

The same basic formula target divided by context equals result, can be used for calculating areas of the page so that they are proportional to their container with one small change, because we're dealing with percentages rather than ems you'll need to multiply the result by 100 to get the percentage.

target ÷ context = result x 100 = %

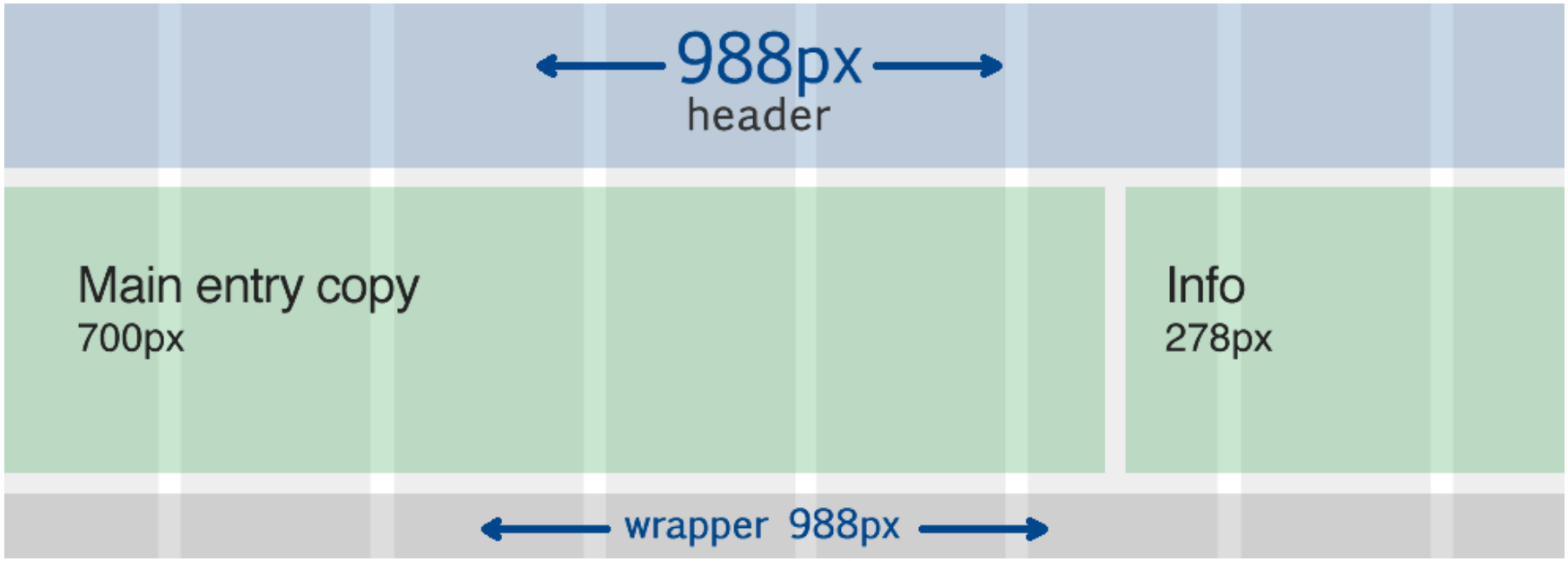
Using Fluid Grids



988px (header) ÷ 988px (wrapper) = 1 x 100 = 100%

```
header {
    width:100%;  /* 988px / 988px = 1 x 100 = 100% */
}
```

Using Fluid Grids



700px (target) ÷ 988px (content) = 0,70850202 x 100 = 70.850%
278px (target) ÷ 988px (content) = 0,28137652 x 100 = 28.137%

```
.entry .main {
    float: left;
    width: 70.850%;
    margin: 10px 0px 0px 20px;
}
```

```
.entry .info {
    float: left;
    width: 28.137%;
    margin: 10px 0px 0px 20px;
}
```

Using Fluid Grids

How do we figure the width of the margins between columns in the fluid grid?

We have to use the width of element's container. We'll use the "entry info" as an example. We have a sidebar that has a margin-left set to 20px. Our element's container is set to 278px, so that determines our context.

```
.entry .info {
        float: left;
        width: 28.137%;  /* 278px (target) ÷ 988px (content) = 0,28137652 x 100 = 28.137% */
        margin: 10px 0px 0px 20px;
}
```

```
.entry .info {
        float: left;
        width: 28.137%;  /* 278px (target) ÷ 988px (content) = 0,28137652 x 100 = 28.137% */
        margin: 10px 0px 0px 7.194%;
}
```

20px (target) ÷ 278px (content) = 0,07194245 x 100 = 7.194%

Using Fluid Grids

Flexible padding

We have to use the width of element itself. For example, in our sidebar we have a paragraph padding set to 5px 10px 0 and as we seen before the width of the sidebar is 300px.

```
.entry .info p {
        padding: 5px 10px 0;
}
```

```
.entry .info p {
        padding: 5px 3.333% 0;
}
```

10px (target) ÷ 300px (content) = 0,03333333 x 100 = 3.333%

Images Responsive Techniques

Fluid layouts works seamlessly for contents such as text which naturally reflows along with its containing element, but **what about content that is by default defined at a fixed size, such as images or other media?**

An image designed for a large desktop monitor is unlikely to fit into a layout designed for a 320-pixel-wide smartphone. On a mobile screen would probably look like thumbnails when viewed within desktop layouts.

This problem underscores **how important it is to find ways to make images and other media responsive.**

There are a some techniques that you can choose to obtain the best responsive results depending your project's requirement.

```
<img src="sample-content/running-sml.jpg?full=sample-content/running-lrg.jpg">
```

Images Responsive Techniques

1. Using media queries to targeting different images according to screen size. This is could be a good way but **you only have to use background-image instead of img tag** and that could be not appropriated.

2. There is another technique that you may use and it's built in Javascript.
Link: https://github.com/filamentgroup/Responsive-Images

3. Recently, the WHATWG (Web Hypertext Application Technology Working Group) the organization behind the HTML5 specification, proposed a modification to the existing image element. **They proposed adding a new attribute called srcset.** The proposed syntax would allow authors to provide a comma-separated list of images and related property such as viewport width, height, and pixel density, **that the user agent could then use to determine which image to use**. The nice thing about this syntax is that it's fairly straightforward and uncluttered **it doesn't require any new HTML elements, and it's backward-compatible**. User agents that don't understand srcset will simply use the default source attributes image.

Link: http://dev.w3.org/html5/srcset/

Images Responsive Techniques

Example of srcet:

```
<img alt="my awesome image"
    src="banner.jpeg"
    srcset="banner-HD.jpeg 2x, banner-phone-HD.jpeg 640w 2x">
```

"w" and "x" values which correspond to the size of the viewport, attempting to serve the most relevant version.

srcet attribute assign banner-HD.jpeg to high DPI devices with screens greater than 640px, and banner.jpeg to everything else.

Form Techniques

Making sure your forms are clear and easy to fill out is an important part of any good user experience. There are a few practices and techniques that you can use and make your forms more responsive.

1. Eliminate external markup on your forms and use structural elements like lists and field sets to properly mark up form structure.

2. It's perfectly acceptable to have labels beside form elements for wider screens, but as the screen sizes become smaller, you need to start putting the labels on top of the input elements.

3. Make sure the color choices that you make and how you organize form sections help guide mobile users through your forms as well.

4. It's helpful to wrap input elements with label tags rather than associating labels with the for attribute.

5. New input types, such as email, URL, and search, not only add more semantic information about the form element, but you'll provide more interaction between user and those inputs.

Hackathon 2014

Responsive Frameworks & Tools

Octubre 2014

Globant
we are ready

Modernizr

Is a small JavaScript library that detects the availability of native implementations for next generation web technologies, i.e. features from the HTML5 and CSS3 specifications. Many of these features are already implemented in at least one major browser (most of them in two or more), and what Modernizr does is, very simply, **tell you whether the current browser has this feature natively implemented or not.**

You can take advantage of these new features in the browsers that can render or utilize them, and still have easy and reliable means of controlling the situation for the browsers that cannot.

Link: http://modernizr.com

Link: http://www.alistapart.com/articles/taking-advantage-of-html5-and-css3-with-modernizr/

We can use media queries to add our own js code depending the current screen size:

```
if (Modernizr.mq("screen and (max-width:480px)")) {
    //your code
}
```

Bootstrap

Twitter Bootstrap is a collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, charts, navigation and other interface components, as well as optional JavaScript extensions.

This framework offers:

Grid system (standard with a 940 pixel wide, grid layout)

CSS stylesheet with media queries

Re-usable components (buttons, typography, breadcrumb, navigation, etc)

Javascript plugins (Modal, Dropdown, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel)

Link: http://twitter.github.com/bootstrap

Foundation

Foundation is a responsive front-end framework to build web sites and apps fast. It has all the basic as well as advanced features for rapid prototyping.

This framework offers:

Grid system (standard with a 940 pixel wide, grid layout)

CSS stylesheet with media queries

Re-usable components (buttons, typography, breadcrumb, navigation, etc)

Javascript plugins (Modal, Drop Down, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel)

Link: http://foundation.zurb.com

HTML5 Boilerplate

HTML5 Boilerplate is a professional front-end template for building fast, robust, and adaptable web apps or sites.

Includes Normalize.css v1.0.x a modern, HTML5-ready alternative to CSS resets and further base styles, helpers, media queries, and print styles.

HTML5 Boilerplate has modernizr and jQuery library ready to use.

Link: http://html5boilerplate.com

iScroll

iScroll was born because mobile webkit (on iPhone, iPod, Android) doesn't provide a native way to scroll content inside a fixed width/height element.
So basically it was impossible to have a fixed header/footer and a scrolling central area.

Link: http://cubiq.org/iscroll

Thank you!

Globant
we are ready