Responsive Web Design

Agenda

- **Definition and Components**
- **Steps to have a responsive web page**
- **Mobile first**
- **Frameworks & Grid Systems**
- **Hands-on**

Responsive Web Design Definition and Components

Școala informală de IT

Responsive Web Design

- **Responsive web design** is the practice of building a website suitable to work on every device and every screen size, no matter how large or small, mobile or desktop.
- The terms was coined by Ethan Marcotte in 2009 in [A List Apart](#)
- Ways of delivering content on multiple resolutions:
  - **Mobile-specific website** - but then there were tablets, and phablets
  - **Adaptive web design** - multiple fixed width layouts
  - **Responsive web design** - *respond* to the size of the browser at *any* given point (multiple fluid grid layouts)

RWD Components

- **3 main components:**
  - **Media queries**
  - **Flexible grid-based layout**
  - **Flexible media**

Media Queries

- **= simple filters that provide the ability to specify different styles for individual browser and device circumstances** (e.g., the **width of the viewport** or **device orientation**)

- [Syntax](#)

- **How to use a media query:**

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

```
@media (min-width: 700px) { ... }
@media (min-width: 700px) and (orientation: landscape) { ... }
@media tv and (min-width: 700px) and (orientation: landscape) {...}
```

Media Queries

- **Open Discussion: How should the code be organized?**
  - Split by the media query
  - Split by the component

Flexible Layout

- **= a layout that is built with a flexible grid, capable of dynamically resizing to any width**
- **Grid system = a structure that allows for content to be stacked both vertically and horizontally in a consistent and easily manageable way**
  - defines the maximum amount of columns per row
  - popular choice: 12 (it is divisible by 1, 2, 3, 4, and 6)
- Flexible grids are built using **relative length units (percentages, ems)**
- The relative lengths are used to declare **common grid property values** such as **width**, (**min-width, max-width**), **margin**, or **padding**

Grid System (link)

Școala
informală
de IT

Flexible Media

- **Problem:** As the viewport grows larger the contained image/video grows wider as well, which means that it needs to grow taller as well.
- **Solution:**
  - Use **max-width: 100%** instead of **width**
  - Use **height: auto** to preserve the aspect ratio

```css
img, video, canvas {
  max-width: 100%;
  height: auto;
}
```

- **Flexible Embedded Media**

Steps to Have a Responsive Web Page

Școala informală de IT

Steps to Have a Responsive Web Page

- **Step 1: Set the Viewport**
  - Use the **meta viewport tag** to **control the width and scaling of the browser's viewport**
  - Include **width=device-width** to **match the screen's width in device-independent pixels**
  - Include **initial-scale=1** to **establish a 1:1 relationship between CSS pixels and device-independent pixels**
  - **Ensure your page is accessible by not disabling user scaling**

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

Școala informală de IT

Steps to Have a Responsive Web Page

- **Step 2: Size Content to the Viewport**
  - Do not use large fixed width elements
  - Content should not rely on a particular viewport width to render well
  - Use CSS media queries to apply different styling for small and large screens
- **Dos and don'ts:**
  - Throw out the Pixels and embrace the % and EM units
  - Don't use images displayed at a width wider than the viewport (this could cause the viewport to scroll horizontally)
  - Beware of using large absolute positioning values that may cause the element to fall outside the viewport on small screens

Steps to Have a Responsive Web Page

- **Step 3:** Use CSS media queries for responsiveness
  - Use media queries to apply styles based on device characteristics
  - Use min-width over min-device-width to ensure the broadest experience
  - Use relative sizes for elements to avoid breaking layout

`min-width`	Rules applied for any browser width greater than the value defined in the query.
`max-width`	Rules applied for any browser width less than the value defined in the query.
`min-height`	Rules applied for any browser height greater than the value defined in the query.
`max-height`	Rules applied for any browser height less than the value defined in the query.
`orientation=portrait`	Rules applied for any browser where the height is greater than or equal to the width.
`orientation=landscape`	Rules for any browser where the width is greater than the height.

Steps to Have a Responsive Web Page

- **Step 4: Choose Breakpoints Based on Content**
  - Create breakpoints based on content, never on specific devices, products, or brands
  - Design for the smallest mobile device first; then progressively enhance the experience as more screen real estate becomes available
  - Keep lines of text to a maximum of around 70 or 80 characters
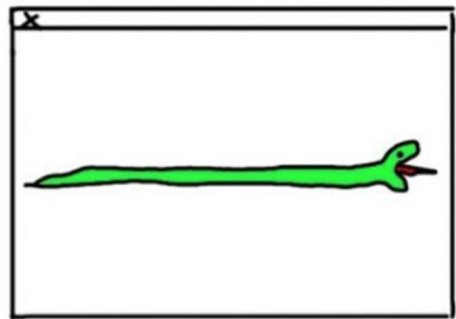- **Demo: Look at the breakpoints in Developer Tools**

Mobile First

Mobile First

```css
.product-img {
  width: 100%;
}

@media screen and (min-width: 700px) {
  .product-img {
    width: 50%;
    float: left;
  }
}
```
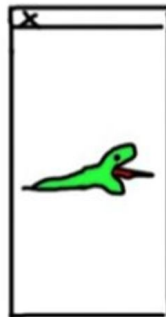

if a snake was responsive would it be

like this   or   like this?

Școala
informală
de IT

Progressive enhancement vs graceful degradation

Hands-on

- **Create a responsive HTML page**
- **Remember to:**
  - **Set the viewport**
  - **Use media queries (start small!)**
  - **Define breakpoints**

Frameworks & Grid Systems

Grid System ([link](#))

Școala
informală
de IT

Grid Systems

- **Primary components: rows and columns**
- **Steps to implement a grid system:**
  - **Reset the border model: set the box-sizing property to border-box for both rows and columns**
  - **Clear floats on rows**
  - **Define columns**
  - **Create gutters:** defining paddings within each column or using a *percentage-based left margin for each column (e.g., 1.6%)*
  - **Calculate column widths**
    - **max amount of columns per row: 12**
    - **need to fulfill that amount for every row, regardless of how many columns you really want (2 columns: 2 elements of 6 columns width)**

Școala
informală
de IT

Grid Systems

- **Calculating column widths**
  - **Width of a single column:** $scw = (100 - (m * (mc - 1))) / mc$
    - **m = margin (1.6%)**
    - **mc = maximum columns (12)**
  - **Width of the other columns:** $cw = (scw * cs) + (m * (cs - 1))$
    - **cs = column span (1-12)**
    - **m = margin**
- **Plugging in the numbers, scw = 6.86666666667%**
- **Mobile first: only institute the column widths until the viewport reaches a minimum width of x pixels**

Frameworks & Grid Systems

- **(Twitter) Bootstrap**
- **Foundation**
- **Skeleton**

Hands-on

- **Include (Twitter) Bootstrap in your page**
- **Update it to respect Bootstrap's grid system, keeping in mind that it should still be responsive**
- **Bootstrap Grid Documentation:**

  https://getbootstrap.com/docs/4.0/layout/grid/

Homework

- **Include [(Twitter) Bootstrap](#) in your blog**
- **Use it to make your blog responsive**
- **Use the grid system it has**

Readings & Tutorials

- Ethan Marcotte's article in A List Apart: https://alistapart.com/article/responsive-web-design

- https://developers.google.com/web/fundamentals/design-and-ui/responsive/

- https://developers.google.com/web/fundamentals/design-and-ui/responsive/patterns

- https://css-tricks.com/the-difference-between-responsive-and-adaptive-design/

- http://johnpolacek.github.io/scrolldeck.js/decks/responsive/

- https://responsivedesign.is/guidelines/

- http://learn.shayhowe.com/advanced-html-css/responsive-web-design/

- https://css-tricks.com/rundown-of-handling-flexible-media/

- Great book: https://resilientwebdesign.com/