Designing for Multiple Device Screens

F27WD – Web Design and Databases
School of Mathematical & Computer Sciences

S.Chumbe@hw.ac.uk
EMB G41

HERIOT WATT UNIVERSITY

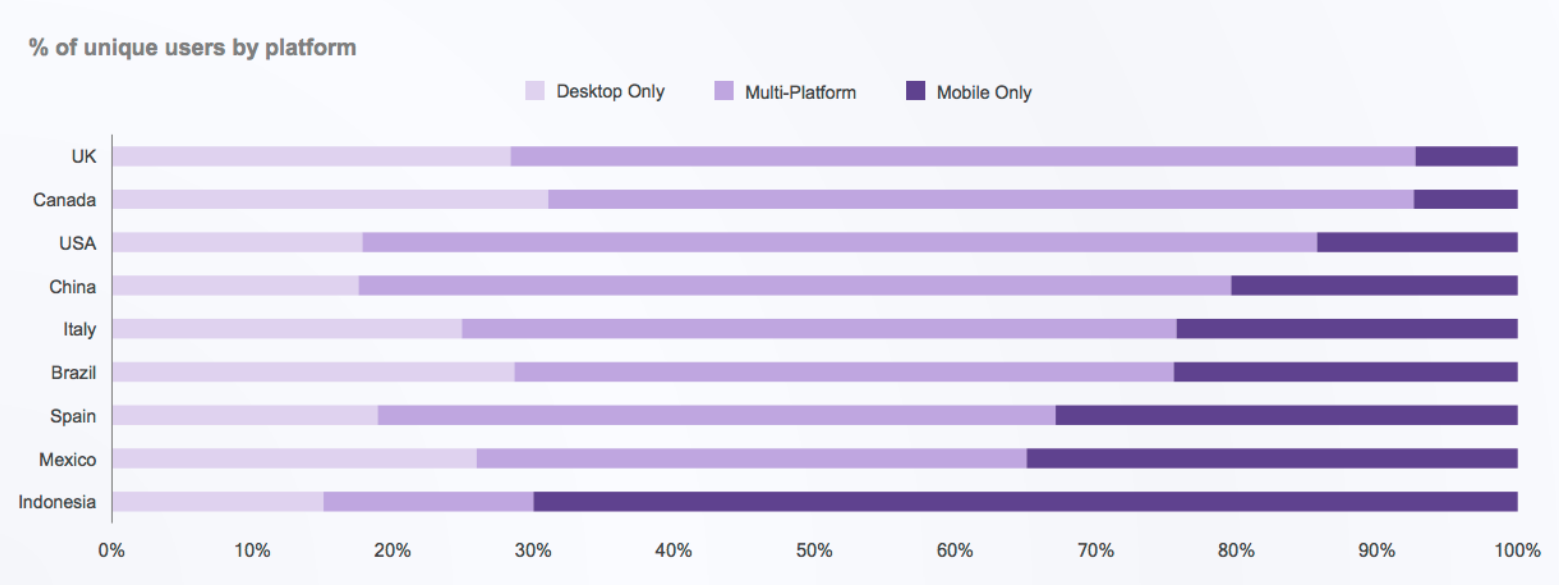
One Design, Multiple Devices

- ✓ Nowadays, websites should be designed to be viewed and interacted with across a range of screen sizes.
- ✓ But designing for different devices is not just enabling the usability of content on different screens. It involves maintaining the same level of UX across all the devices used to access the Internet.
- ✓ You will succeed if your users believe that the website was actually designed for their devices instead of being simply stretched to fit their particular screen.
- ✓ Therefore, in order to create a good UX, it's paramount that your websites works perfectly in a diverse range of devices and screen sizes.

One Design, Multiple Devices

- ✓ **Points to take into account:**
- ✓ **Mobile-first should be your first choice because:**
  "*Google primarily will use the mobile version of a site's content to rank pages from that site*"
- ✓ https://webmasters.googleblog.com/2016/11/mobile-first-indexing.html
- ✓ **You should continue designing for your audience first, the masses second but don't ignore the increasing importance of mobile devices. Three years ago, mobile overtook desktop Internet access and currently the majority of users are multiplatform users, who are continually switching between mobile and desktop.**

One Design, Multiple Devices



(source image: https://www.smartinsights.com/mobile-marketing/mobile-share-total-digital-minutes/attachment/multi-platform-device-use-2017/)

How we can design websites for multiple device screens?

- ✓ Basically you have three options: Using a front-end solution, a back-end solution or a combination of those two solutions.
- ✓ The front-end solution is generally known as **Responsive Web Design (RWD).**
- ✓ The back-end solution is called **Adaptive Web Design (AWD).**
- ✓ Which you choose totally depends on your skills and the findings of your IA study.

How we can design websites for multiple device screens?

- ✓ **An additional and radical way to design websites for mobile devices is Progressive Web Apps (PWA).**
- ✓ **PWA brings features of native apps to the mobile browser experience using web standards-based technologies such as JSON & JavaScript.**
- ✓ **Three technologies are behind PWA:**
  - ✓ Service Workers: JavaScript middleware independently working between the web app and the networks.
  - ✓ The App Shell model to keep the shell of the UI web app and its content separate, and separately cacheable.
  - ✓ Manifest, a standard W3C specification written in JSON to enable interoperability of the web app with other web apps.

RWD and AWD

- ✓ **Better Together**

  RWD and AWD are not mutually exclusive solutions. They can be combined to acquire the benefits of both. They are complementary solutions that can work together to create designs for multiple devices and thus provide rich UXs.

Technology we need to implement RWD and AWD

- ✓ **RWD**
  HTML, CSS & JavaScript (JS)

- ✓ **AWD**
  HTML, CSS, JavaScript (JS), a server-side script language and a database

“Can we design for Multiple Device Screens with HTML only?”

Yes

For example, the first ever webpage is a Responsive Website:

http://info.cern.ch/hypertext/WWW/TheProject.html

But, the demands of current websites need more than simple HTML

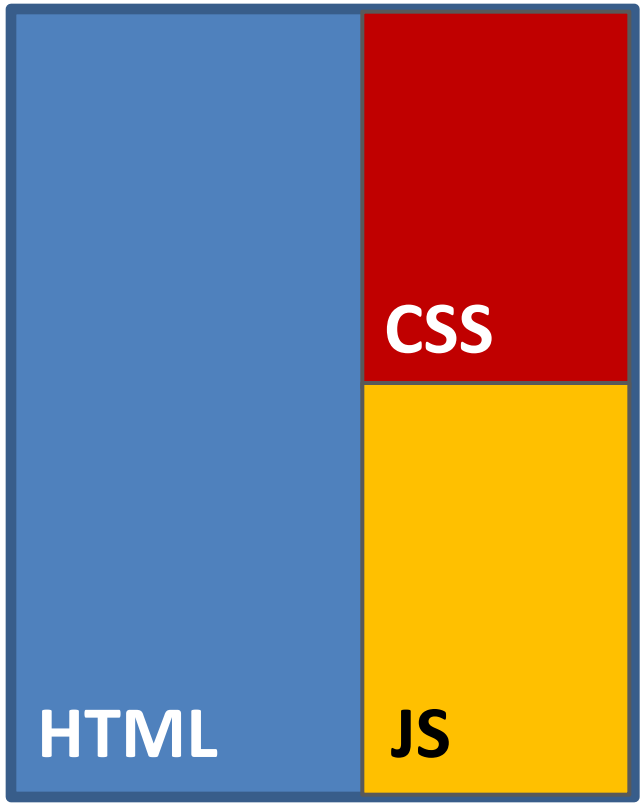
One Design, Multiple Devices

Responsive Web Design (RWD)

Responsive Web Design (RWD)

- ✓ With RWD, your website will have the ability to "respond" to the size of the viewport (generally the screen size) of your device.
- ✓ RWD uses client side, front-end or UI software tools and technologies, such as pure CSS and JavaScript or UI frameworks or ready-to-use toolkits.
- ✓ RWD works by hiding, showing, stacking, shrinking and propagating HTML elements on the screen in a way that is appropriate for the size of the device's screen.
- ✓ In essence, with RWD we can use a single code base to produce many user interfaces.

Responsive Web Design (RWD)



RWD techniques make your webpages **fluidly** change and **respond** to fit any screen or device size.

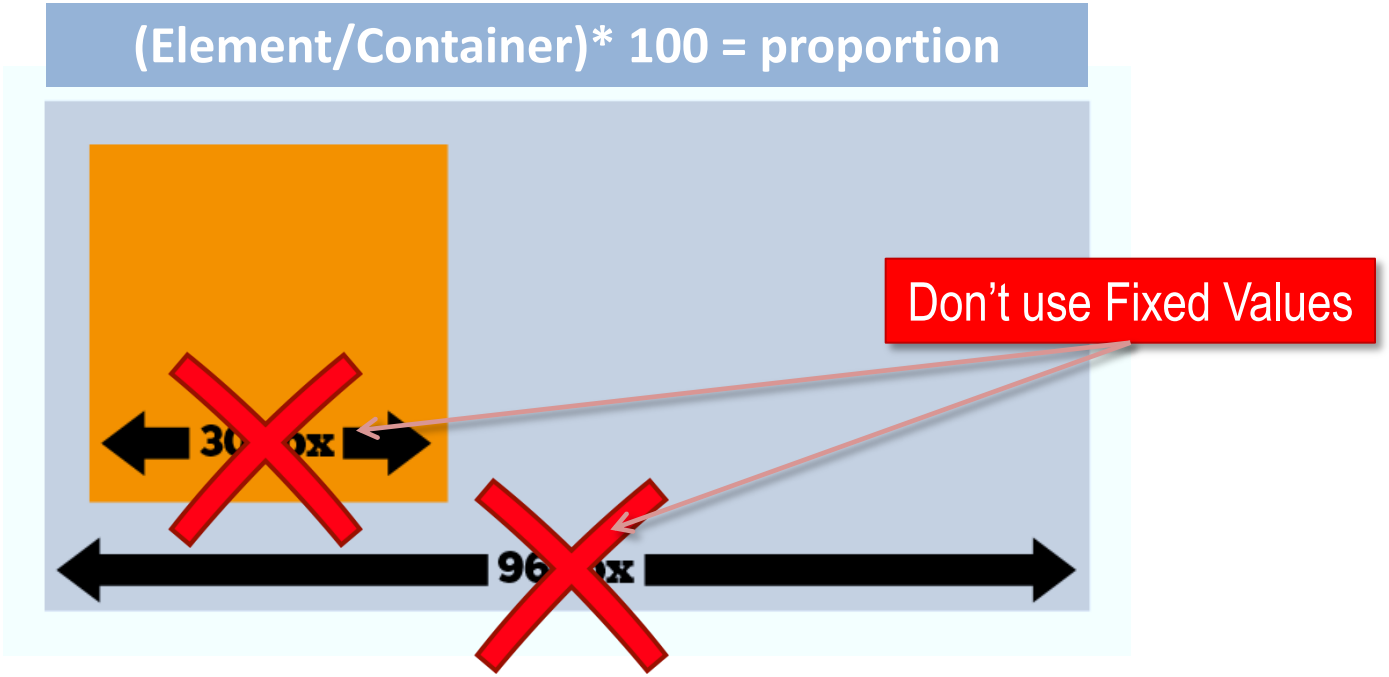
You only need to develop and maintain one single code (layer) which deals with the content, presentation and interactivity of a website on different devices. One HTML, one CSS and one JS.

Responsive Web Design Techniques

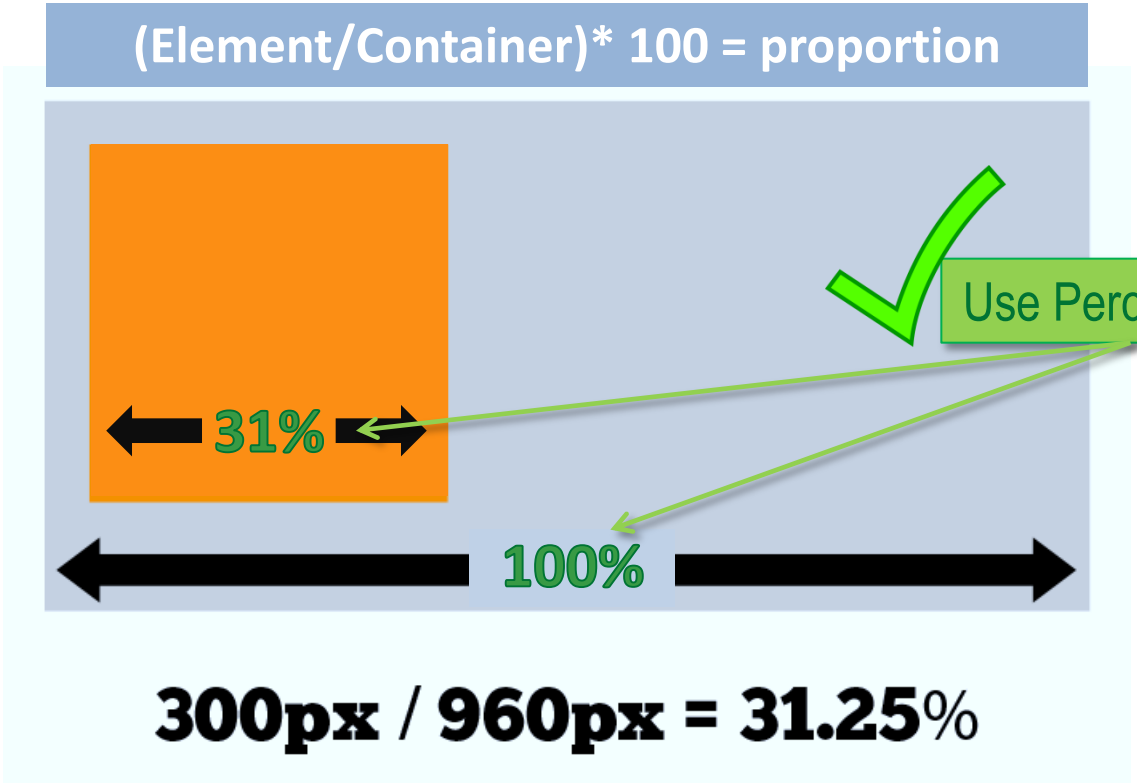
1. Fluid Grids
2. Fluid Images
3. Media Queries
4. Content Load Management
5. Viewport Meta Tag

- ✓ Truly Responsive Web Design requires all five techniques to be implemented
- ✓ Fluid (Liquid) Website Design is RWD withou media queries

1. Fluid Grids



1. Fluid Grids



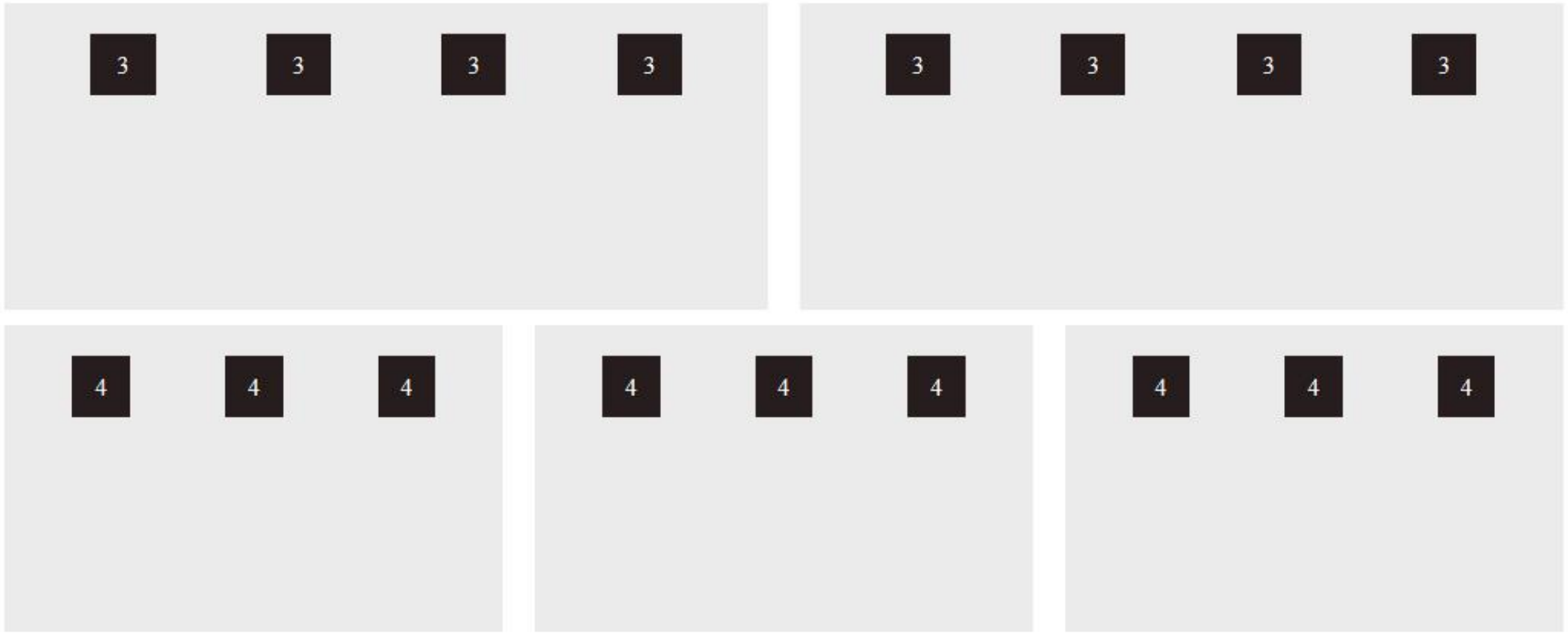
Set widths in terms of **PERCENTAGES**. So when squeezed or stretched the elements will resize in relation to one another.

1. Fluid Grids

Basically Fluid Grids means
stop using pixel-based sizes
and start using **percentages**,
`rem`-based or `em`-based sizes
for HTML elements*

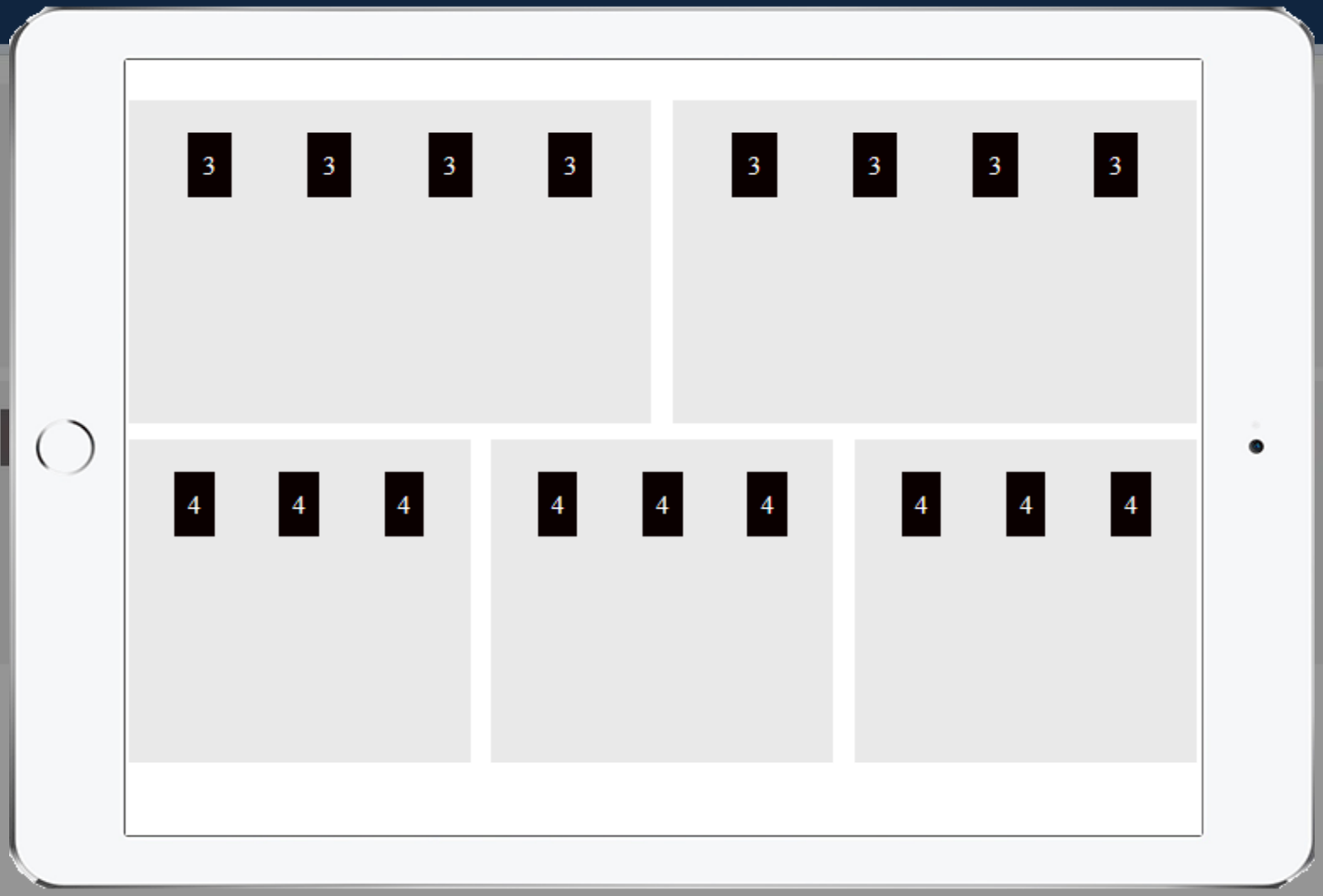
** You don't need to worry about text. Text is fluid by default: as the browser window narrows, text reflows to occupy the remaining space*

1. Fluid Grids



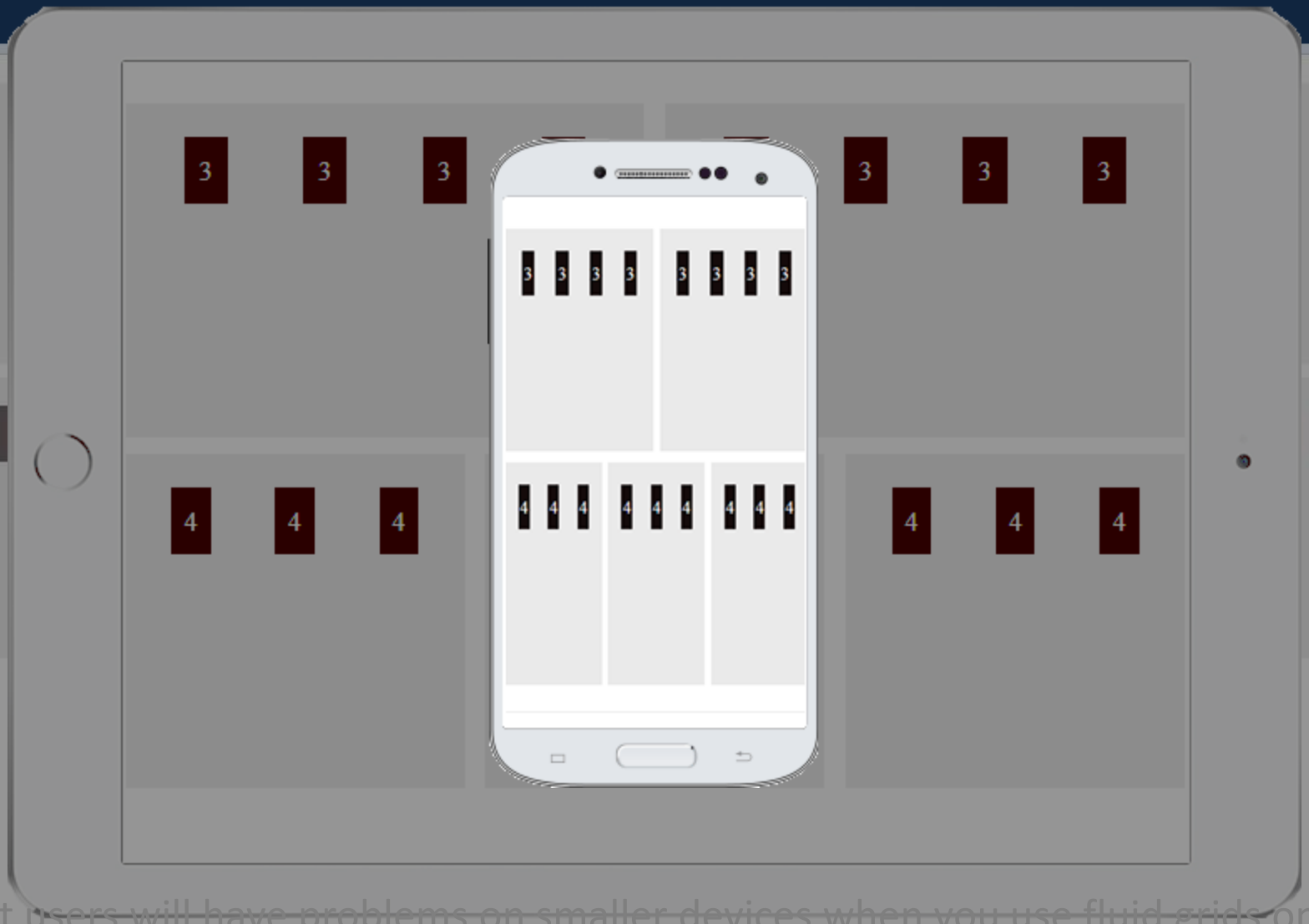
With Fluid Grids, HTML elements will adapt to the change in browser window or device.

1. Fluid Grids



But users will have problems on smaller devices when you use fluid grids only.
Using only fluid grids is not RWD.

1. Fluid Grids



But users will have problems on smaller devices when you use fluid grids only.

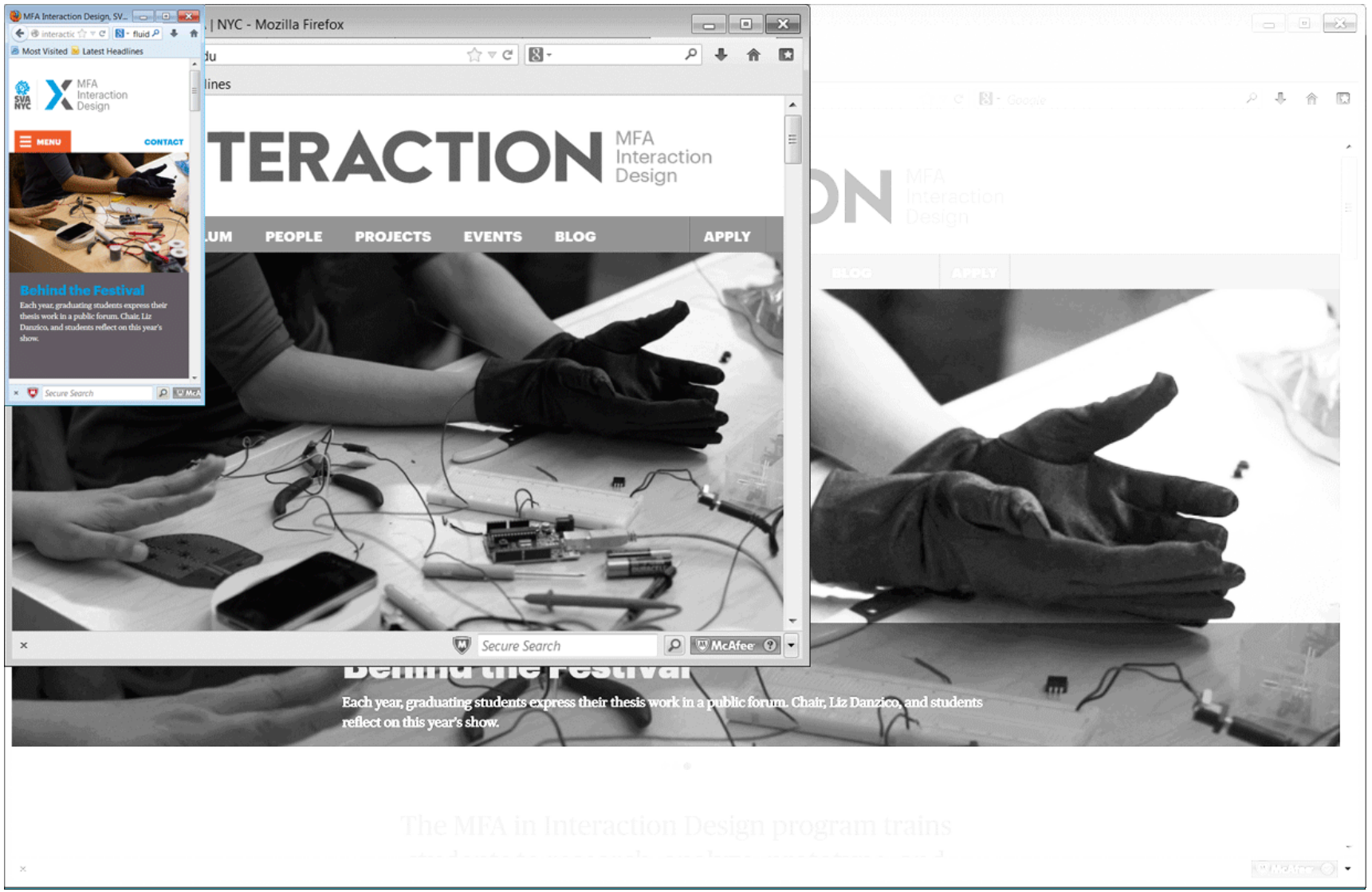
Using only fluid grids is not RWD.

1. Fluid Grids

A very basic example of the effect of using Fluid Grids only

http://www2.macs.hw.ac.uk/~santiago/F27WD/rwd1.html

2. Fluid Images



2. Fluid Images

This feature allows you to avoid your images from being cut off or displayed out-of-scale on small devices
You implement Fluid Images either by **scaling** or by using the **CSS overflow** property.

You should supply the image in the best quality and size possible and then let CSS adapt the image to the right size.

2. Fluid Images

CSS Scaling Solution

Scaling is implemented by setting the media element's max-width to 100 percent:

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

The image would display at their native dimension so long as there is enough room in the HTML container to do so; otherwise the image will scale to fit the screen.

Browsers keep the image's proportions intact, when the page scales up or down accordingly.

Better solutions?

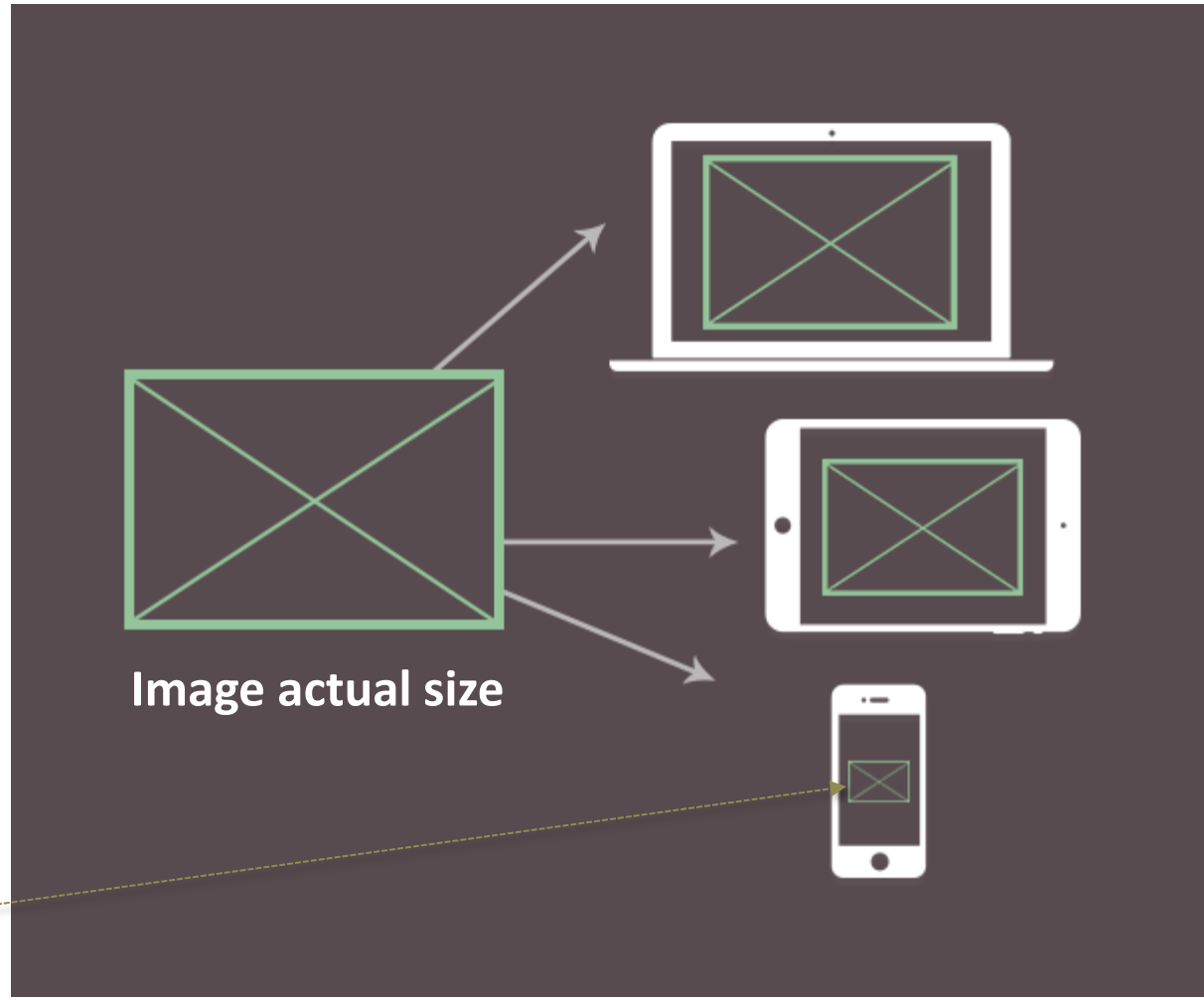
YES, but they do not work for all cases:

- Add JavaScript to dynamically preserve image quality
http://unstoppablerobotninja.com/entry/fluid-images
- Using proportions for the image width:
http://demosthenes.info/blog/586/CSS-Fluid-Image-Techniques-for-Responsive-Site-Design

2. Fluid Images

RWD CSS Solution

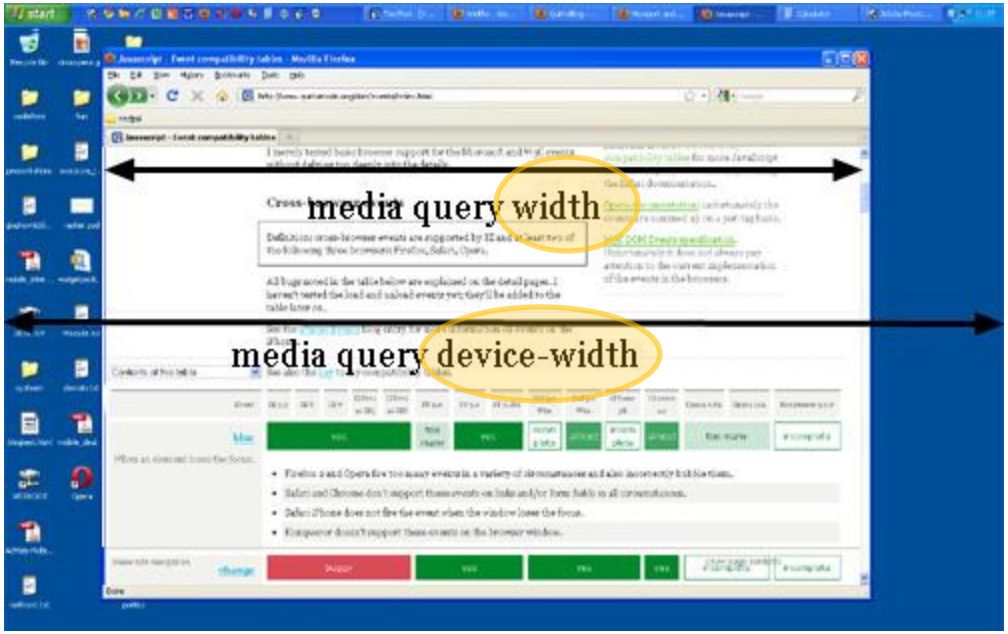
```
img, object {
    max-width: 100%;
    width: 100%;
}
```



The image could be rendered too small to be seen properly

3. CSS3 Media Queries

Media queries are useful to apply different CSS styles depending on a device's specific characteristics (such as the width of the browser viewport (part of the webpage that is visible).



device-width > width

3. CSS3 Media Queries

Media queries are CSS declarations that allow us to detect certain features of our browser then write different CSS to for that purpose.

device-width < width

3. CSS3 Media Queries

@media

```
@media not|only mediatype and (media condition) {
    CSS-Code;
}
```

Media types: all | screen | print | speech

Media conditions: expressions created with over 20 features & three logical operators (AND, NOT, ONLY)

3. CSS3 Media Queries

The following CSS will apply if the viewing area is bigger than 600px.

screen's width is at least

```
@media screen and (min-width: 600px) {
    .content { float: left; }
    .social_icons { display: block; }
    // and so on...
}
```

- width and height of the viewport (max-width (available display screen's width))
- width and height of the device (max-device-width (actual device screen's width))
- device orientation (landscape/portrait)
- device resolution

3. CSS3 Media Queries

Using media queries

Start styling the small screen first, then add media queries afterwards to adjust the layout for bigger screens.

```
body { background: red; }
@media (min-width: 600px) {
    body { background: green; }
}
@media (min-width: 800px) {
    body { background: blue; }
}
```

3. CSS3 Media Queries

```
h1 {
   color: red;
}

@media only screen and (min-width: 650px) {
    h1 {
        color: green;
        text-decoration: underline;
    }
}

@media only screen and (min-width: 980px) {
     h1 {
          color: blue;
     }
}
```

```
<h1>Hello</h1>
```

Screen width < 650px

Hello

Screen width > 650px

Hello

Screen width > 980px

Hello

4. Content Load Management

Hidden content is still loaded
and
it's wasting bandwidth, processor cycles and battery life

Content Load Management

First Try: Just use **display: none**

```
@media all and (max-width: 600px) {
    #try1 { display:none; }
}

<div id="try1">
    <img src="largeImage.jpg" alt="" />
</div>
```

Result:
It doesn't prevent the downloading of the image. Avoid it!

Content Load Management

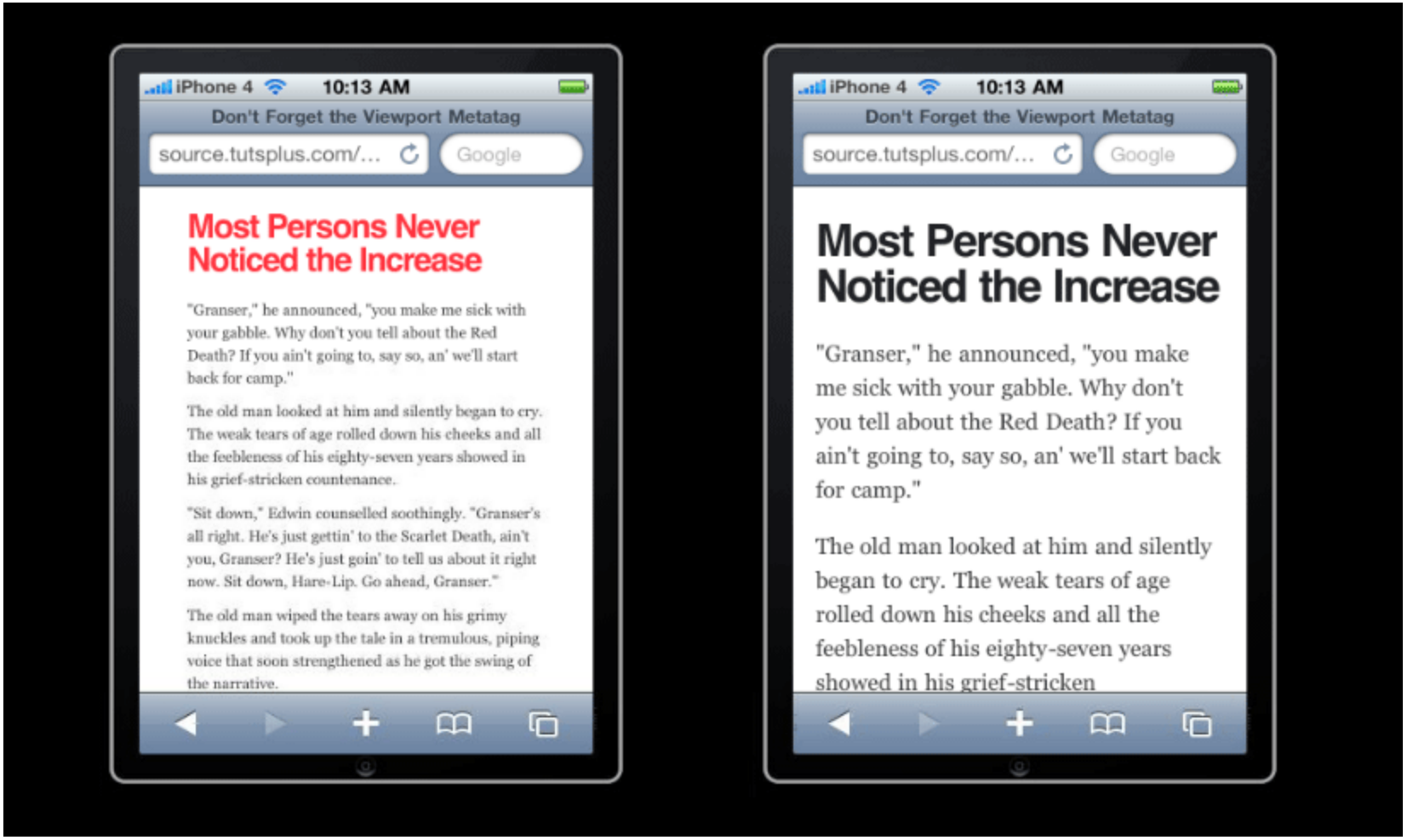
BETTER SOLUTION: **Use JavaScript matchMedia**

```
var wsize = window.matchMedia("(min-width: 400px)");

 if (wsize.matches) {
     /* the view port is at least 400 pixels wide */
     $("#featured").load("bigContent.php");
 }
```

jQuery **AJAX**

5. Viewport Meta Tag



without viewport

with viewport

Viewport Meta Tag

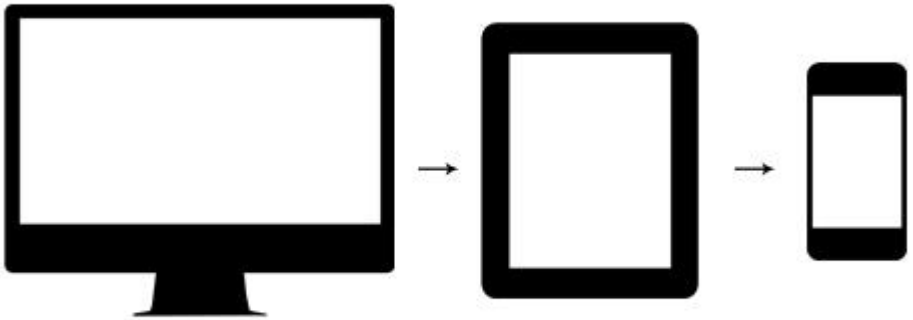
A typical mobile-optimized site contains something like this:

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

One Design per Device

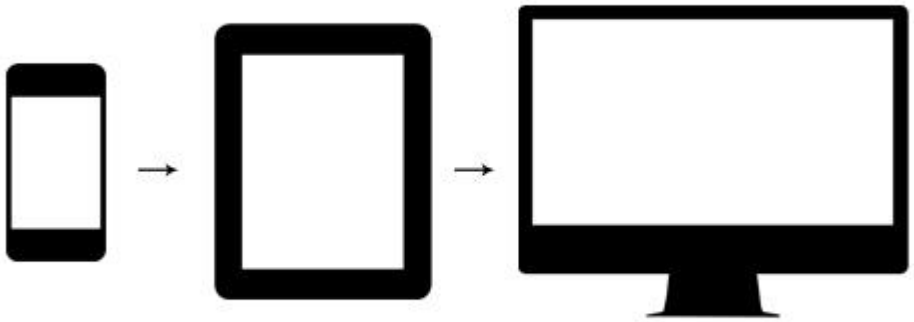
Adaptive Web Design (AWD)

Graceful Degradation (desktop-first)



OR

Progressive Enhancement (mobile-first)?



Adaptive Web Design Techniques

1. Progressive Enhancement
2. Adapting content with HTML5
3. Using CSS3 to detect device

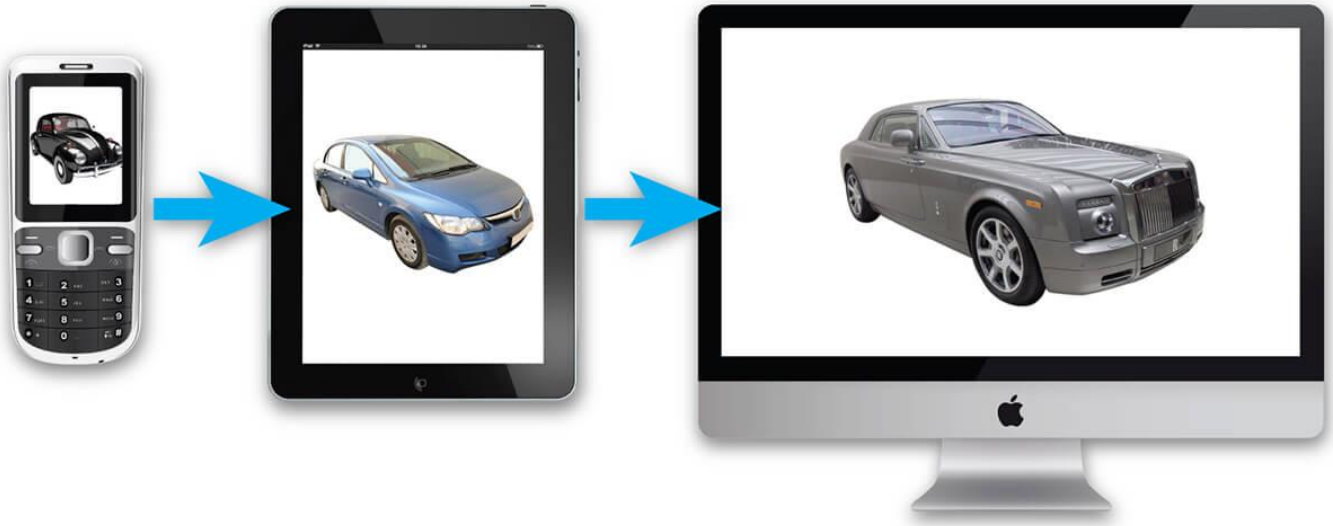
AWD uses the concept of Progressive Enhancement:

The content of a website should be able to be enjoyed without the added enhancements.

Like a peanut that is always a great food even without enhancements such as chocolate to make it test better and, a colourful coating to make it look beautiful.

Progressive Enhancement (PE)

When constructing any web interface, PE primarily focuses on information, to make sure that it is always both available to and usable by anyone on any device, using any browser.

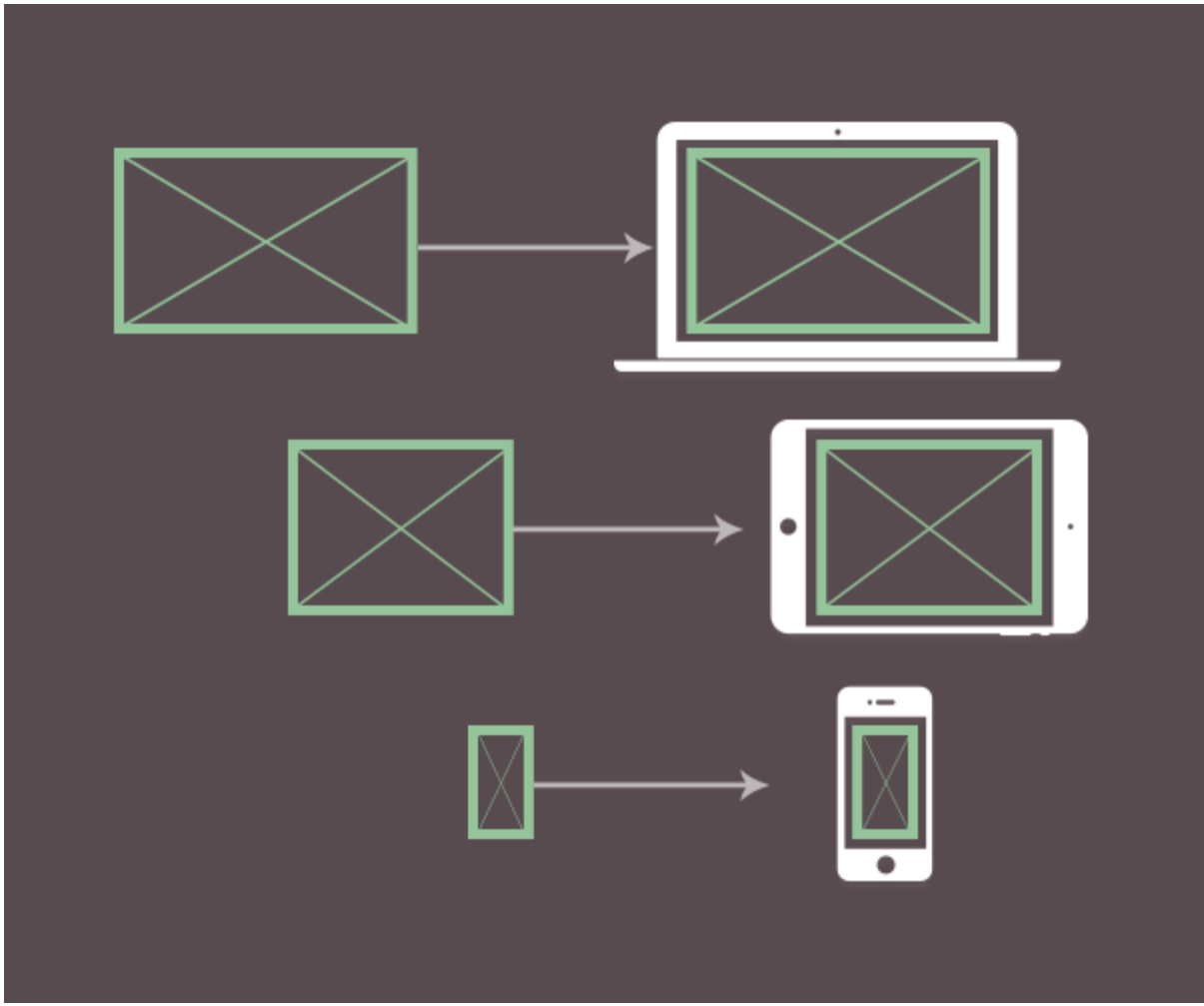


PE doesn’t mean providing the same content & design in different browsers; it means adapting your content and design to different devices to always provide the same wonderful UX.

Adapting Content

Using HTML5's picture

```
<picture>
  <source srcset="smaller.jpg"
  media="(max-width: 300px)">
  <source srcset="small.jpg"
  media="(max-width: 600px)">
 <source srcset="default.jpg">
  <img srcset="default.jpg"
   alt="fall-back image">
</picture>
```



Adapting Content

Using HTML5's template

The HTML **<template>** element is a place-holder of content that is not to be automatically loaded when a page is used but only after it has been loaded by a JavaScript programme that will do so by checking the device properties.

Example of Content Load Management with <template>

```
<table id="producttable">
   <thead>
     <tr>
        <td>Product Name</td>
        <td>Product Image</td>
     </tr>
   </thead>
   <tbody>
   <!-- existing data could be loaded here -->
   </tbody>
 </table>
 <template id="productrow">
    <tr>
       <td class="record"></td>
       <td></td>
    </tr>
 </template>
```

```
<script>
if ('content' in document.createElement('template')) {
   var t = document.querySelector('#productrow'),
   td = t.content.querySelectorAll("td");
   td[0].textContent = "Large Image";
   td[1].textContent = "<img src='bigimage.jpg' alt=''>";
   // Clone the new row and insert it into the table
   var tb = document.querySelector("tbody");
   var clone = document.importNode(t.content, true);
   tb.appendChild(clone);
} else {
   // Sorry your browser doesn't support templates.
 }
</script>
```

Using CSS3 to detect device

```
@media all and (max-width: 600px) {
   #try1 { display:none; }
}

<div id="try1">
   <img src="largeImage.jpg" alt="" />
</div>
```

Content Load Management with <template>

```
2    <thead>
3      <tr>
4        <td>UPC_Code</td>
5        <td>Product_Name</td>
6      </tr>
7    </thead>
8    <tbody>
9      <!-- existing data could optionally be included here -->
10   </tbody>
11 </table>
12
13 <template id="productrow">
14   <tr>
15     <td class="record"></td>
16     <td></td>
17   </tr>
18 </template>
```

AWD Pros and Cons

Pros

- ✓ AWD makes room for a tailored UX, with individual devices treated as unique mediums, and unique designs: ‘device-first’.
- ✓ Fast page loads are common in AWD because only the essential elements are loaded on mobile devices.
- ✓ Existing sites don’t need alteration, the layout is already set up!

AWD Pros and Cons

Cons

- AWD requires advanced knowledge of web front-end development
- Its initial implementation can be a rather costly, time consuming and resource demanding.
- Making enhancements and modifications to multiple layout templates can be uneconomical, and difficult to manage when it comes to multiple views.
- AWD can result in a less consistent cross-device experience than RWD because the “jumps” when changing the elements to adapt the page to the screen size.