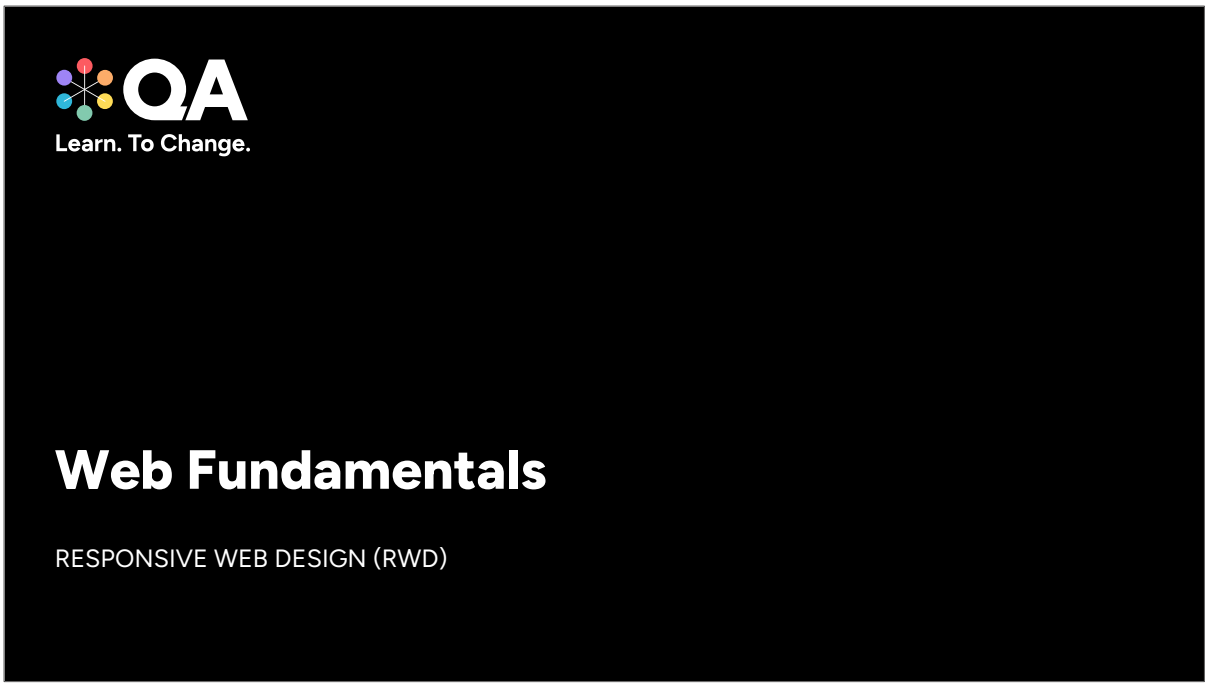
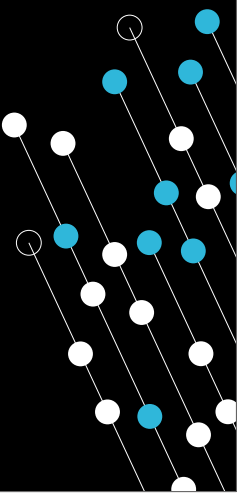

QA
Learn. To Change.
Web Fundamentals
RESPONSIVE WEB DESIGN (RWD)

## Learning Objectives

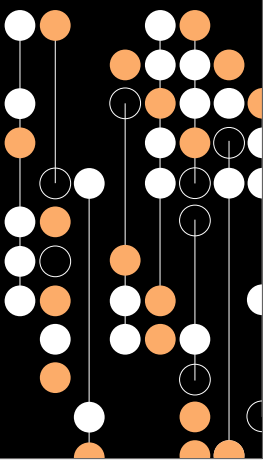
- Understand why 'Mobile First' and 'Responsive Web Design'
- Be able to apply RWD principles
- Be able to use Media Queries
- Understand and implement grid systems
- Understand flexbox
- Be able to create and use responsive images




Responsive Web Design
Mobile First and Responsive Design
Viewports
Media Queries
Grids
Flex Box
Responsive Images

# Mobile First and Responsive Web Design

Responsive Web Design



QA

# Mobile First

UI design idea first coined by Luke Wroblewski in 2011

- Encourages UI to be designed for a mobile device before a large screen
- Helps to identify what most important content is
- Ensures that this content is displayed prominently on a mobile device

Why 'Mobile First'?

- Prepares you for the explosive growth and new opportunities emerging on mobile today
- Forces you to focus and prioritise your products by embracing the constraints inherent in mobile design
- Allows you to deliver innovative experiences by building on new capabilities native to mobile devices and modes of use

**Taken from Mobile First, L Wroblewski, P1**

QA

# Responsive Web Design

- **Pioneered by Ethan Marcotte in 2010**
- **Ensures that web pages render well regardless of device/screen/window size**
  - Adapts the layout by using fluid, proportion-based grids, flexible images and media queries
    - Fluid grid concept requires sizing to be in relative units (percentages rather than pixels or points)
    - Flexible images sized in relative units
    - Media queries allow different rules to be applied usually dependent on width of screen available
- **Only uses HTML and CSS**

https://abookapart.com/products/responsive-web-design

QA

# Content First - Manipulate

RWD is more than changing layouts

Media queries do not just change layouts

- Allow designers to manipulate content dependent on viewport size and device type

Make sure main content is visible on phone when user views homepage

- Hamburger navigation, move ads down, etc

Changing content is common

- Lower resolution videos for small screens, etc

Removing content completely is last resort

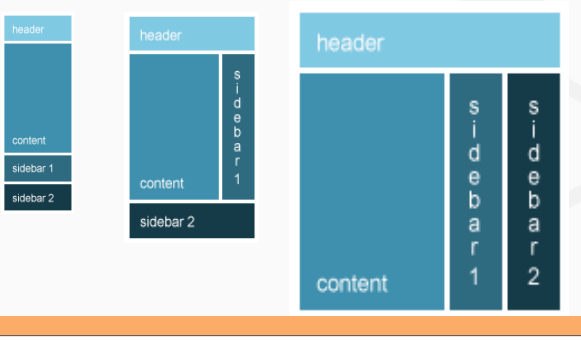
## General Rules for Manipulating Content

- Reorder
- Reposition
- Replace
- Remove (last resort)

QA

## Responsive is not just smaller text

- Responsive development is about re-envisioning content
- Relative sizing can make content too big on mobile devices and too small on desktop sized screens and above




Response design starts with the concept of mobile first which means you start with the smallest device and consider the devices that may view website. You build from this smallest device adding new content and taking advantage of the larger website as you develop.

Working from the mobile first perspective we must consider what information is essential to the site.

QA

# The Viewport

User's visible area of the page

Varies from device to device

- Small on a mobile phone, larger on a full size screen

Setting the viewport

- Done in a meta tag with the head of the HTML page
- Has to be included for responsiveness to work

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Size content to the viewport

- Users should only scroll vertically – never horizontally
- Content should not rely on a particular viewport size
- Media queries should be used to apply different styles on different sized screens

QA

# max-width

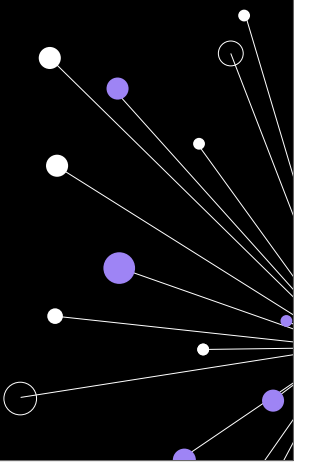
- Will solve most problems with images
- Sets the maximum width of a given element (if % it is that of the parent element)
- Elements will not appear wider than the maximum specified for the element

```
img {
  max-width: 100%
}

img.biggest {
  max-width: 300px
}
```

# Media Queries

Responsive Web Design



QA

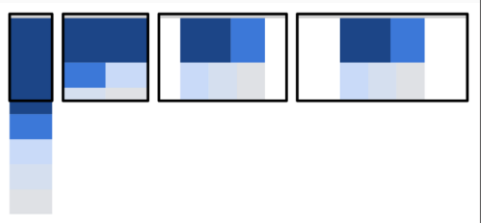
# Media Queries

Modify the layout of your site based on different criteria

Same layout not necessarily appropriate for every device

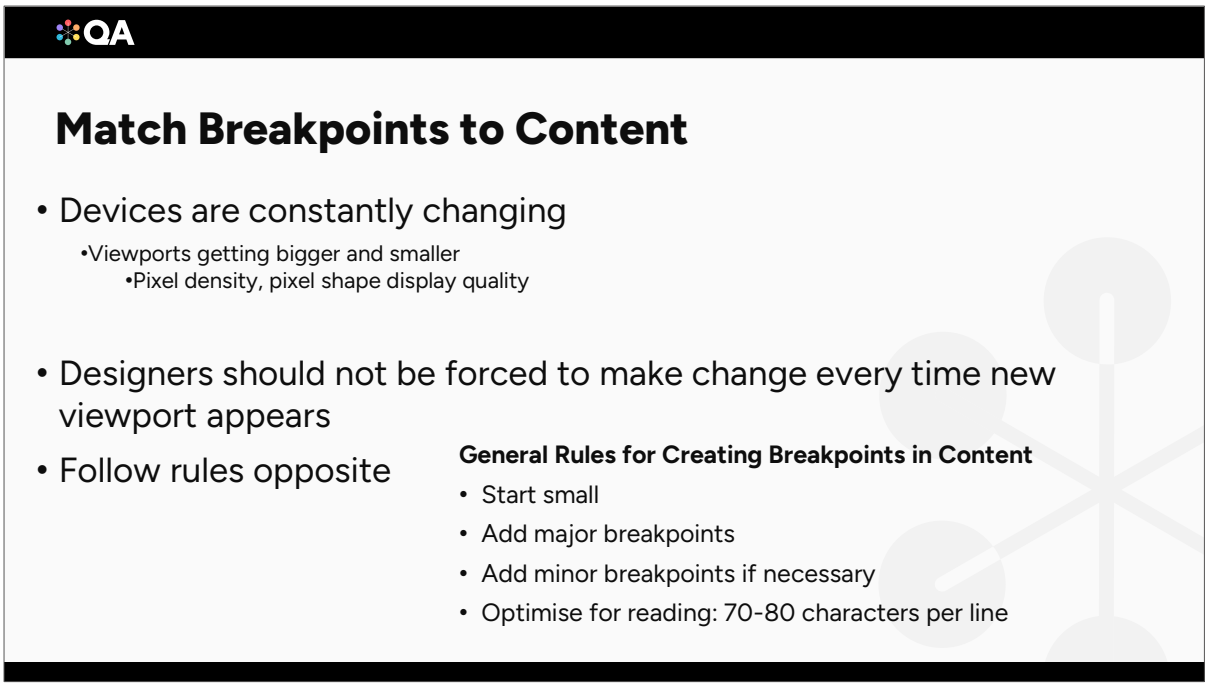
Might have

- Single column on phone
- Two columns on tablet
- Three columns on desktop
- etc



```
@media (max-width: 480px){
  /* layout for phones */
}

@media (max-width: 720px) {
  /* layout for tablets */
}
```




Start the design process with the smallest form factor.

Then add the major breakpoints for the form factors that you will work with: phone, tablets, laptops and wide screen devices.

You can then create minor breakpoints to handle specific changes to elements that don't affect all elements.

The final detail to keep in mind is to optimize the content for reading. Ideally keep the width of your content to 70 to 80 characters. Wider than that value makes content harder to read.

QA

# Mobile First Media Queries

- Look at the minimum width of a device to display content in particular way instead
- No fixed rule about whether to include media queries inline or use a separate file
- Might want to consider using ems or rems

```
/* small by default */

@media (min-width: 480px){
  /* medium */
}

@media (min-width: 720px) {
  /* large */
}
```

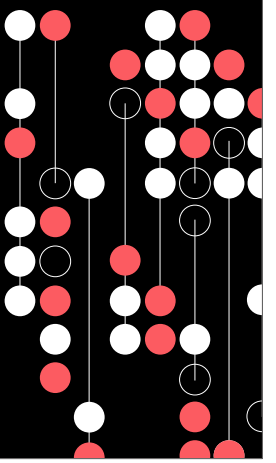
QA

# Quick Lab 11 – Media Queries

Create some media queries to make a page adapt to the width available.

# Grids

Responsive Web Design



QA

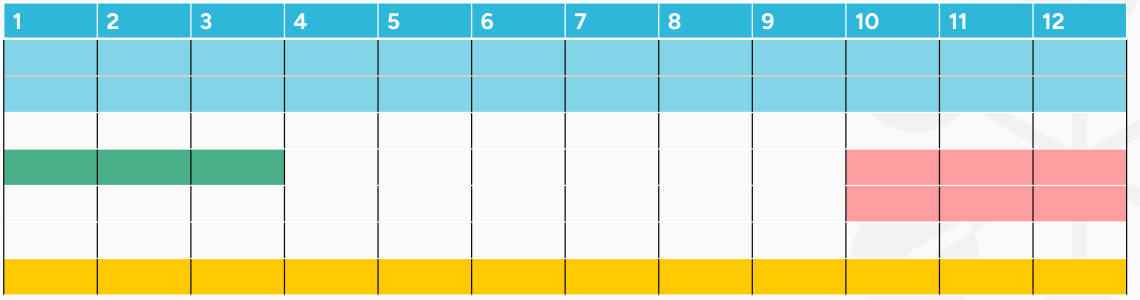
# Changing Layouts – Grid View

Modern websites often based on a grid-view

Pages are divided into (usually 12) equal columns

Helpful when designing pages for different devices as easier to place elements on page

- Total width is 100% and will shrink and expand as the browser window is resized



QA

## Responsive Grids

All HTML elements have to have box-sizing set to border-box

- Ensures padding and border are included in total width and height of elements

```
* { box-sizing: border-box; }
```

Can create simple responsive page with 2 columns

```
.left { width: 25%; float: left; }
.right { width: 75%; float: left; }
```


For a 12 column grid, each would take up 8.33%.

We could create 12 classes, prefixed with "col-" and then the number of columns the section should span. The width property would then be set, as a percentage, to the number of columns multiplied by 8.33.

The number of columns inside each container labelled as a row should always add up to 12

QA

## CSS Grid Layout Module

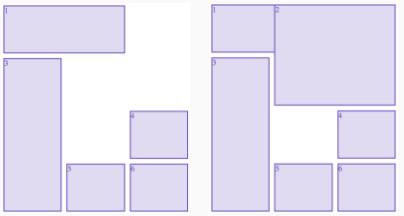
- Offers a grid-based layout system, with rows and columns

- HTML

```
<div class="grid-container">
  <div class="one">1</div>
  <div class="two">2</div>
  <div class="three">3</div>
  <div class="four">4</div>
  <div class="five">5</div>
  <div class="six">6</div>
</div>
```

- CSS

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 10px;
  grid-auto-rows: minmax(100px, auto);
}
.one { grid-column: 1/3; grid-row: 1 }
.two { grid-column: 2/4; grid-row: 1/3 }
.three { grid-column: 1; grid-row: 2/5 }
.four { grid-column: 3; grid-row: 3 }
.five { grid-column: 2; grid-row: 4 }
.six { grid-column: 3; grid-row: 4 }
```



The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

The left image shows the layout with div2 removed to show the rendering of div1.

The right image shows how div2 overlays div1.

QA

## CSS Grid Layout Module

There are 18 different properties that can be set for the CSS Grid Layout module:

| | | |
|---|---|---|
| `column-gap` | `grid-auto-rows` | `grid-row-start` |
| `gap` | `grid-column` | `grid-template` |
| `grid` | `grid-column-start` | `grid-template-areas` |
| `grid-area` | `grid-column-end` | `grid-template-columns` |
| `grid-auto-columns` | `grid-row` | `grid-template-rows` |
| `grid-auto-flow` | `grid-row-end` | `row-gap` |

There are 3 functions that can be used:

| | | |
|---|---|---|
| `fit-content()` | `minmax()` | `repeat()` |

For more information on the CSS Grid Layout module, see:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

QA

## Grid Frameworks

- **No need to define all of the classes by hand**
- **Several Responsive CSS Frameworks exist that already have the CSS classes defined along with many other useful features**
  - All developer has to do is add correct classes to the HTML
- **Examples are Bootstrap and Foundation**

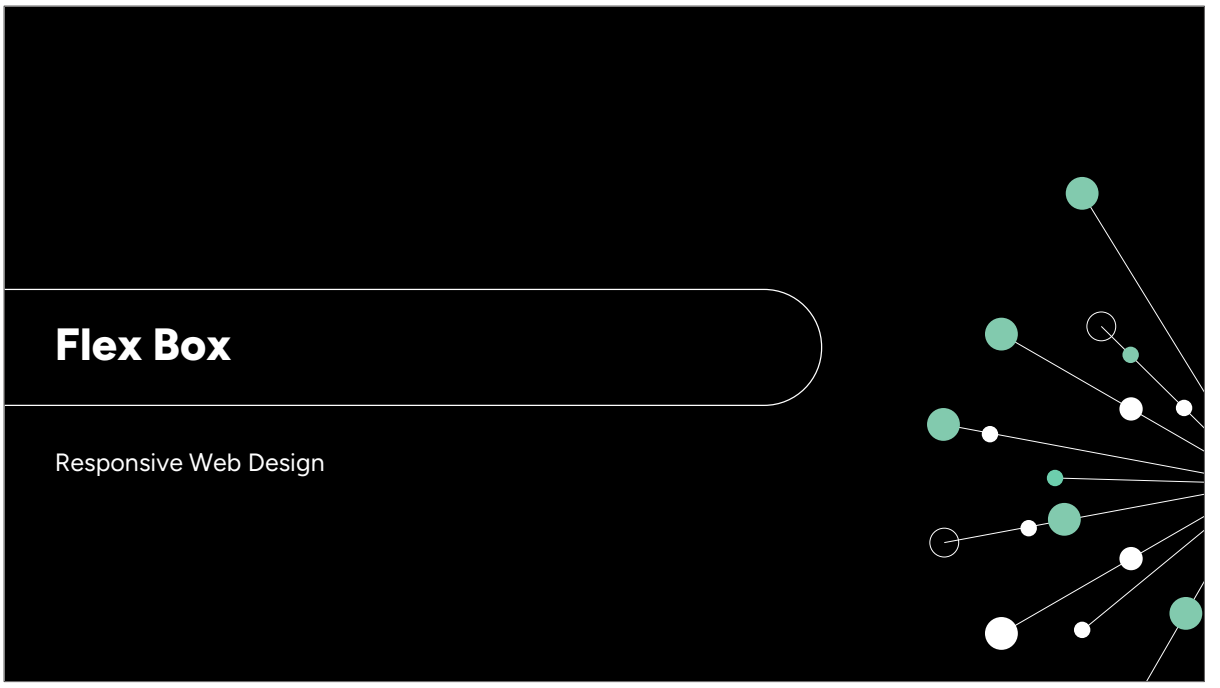
Bootstrap: https://getbootstrap.com/

Zurb Foundation: https://foundation.zurb.com

QA

# Quick Lab 12 - Grids

Add CSS classes to HTML elements to utilise a provided CSS grid system.


Flex Box
Responsive Web Design

QA

# Flex Box

Enables easy design of responsive layout structure without using float or positioning

Requires a flex container

- Element becomes flex container by setting display property to flex

```
.flex-container { display: flex; }
```

QA

## Flex Container Properties

| Property | Description | Common Values |
|---|---|---|
| `flex` | Specifies how flex item will grow or shrink to fit the container – shorthand to set `flex-grow`, `flex-shrink` and `flex-basis` | `auto, initial, none, <+ve num>` |
| `flex-grow` | Specifies how much of available space should be assigned to the item | `<+ve num>` |
| `flex-shrink` | Specifies how items will shrink to fit container when default size is larger than flex container | `<+ve num>` |
| `flex-basis` | Specifies the initial main size of a flex item (determines size of `content-box` unless `box-sizing` is defined) | `<width> (% or units), auto, fill, max-content, min-content, fit-content, content, unset` |

flex-direction: https://developer.mozilla.org/en-US/docs/Web/CSS/flex-direction

QA

## Flex Container Properties

| Property | Description | Common Values |
|---|---|---|
| `flex-direction` | Defines direction the container stacks the flex items | `row, row-reverse, column, column-reverse` |
| `flex-wrap` | Specifies if the flex items will wrap if necessary | `nowrap, wrap, wrap-reverse` |
| `flex-flow` | Shorthand property to set `flex-direction` and `flex-wrap` | As above |
| `justify-content` | Defines how browser distributes space between and around content along main axis of container | `center, end, flex-end, flex-start, left, right, start, norm, space-between, space-around, space-evenly, stretch, safe center, unsafe center, unset` |

flex-direction: https://developer.mozilla.org/en-US/docs/Web/CSS/flex-direction

flex-wrap: https://developer.mozilla.org/en-US/docs/Web/CSS/flex-wrap

flex-flow: https://developer.mozilla.org/en-US/docs/Web/CSS/flex-flow

justify-content: https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content

QA

## Flex Container Properties

| Property | Description | Common Values |
|---|---|---|
| `align-items` | Sets `align-self` value on all direct children as a gourp | `normal, stretch, baseline, self-start, self-end, flex-start, flex-end, end, center, start` |
| `align-content` | Specifies how browser distributes space between and around content items along cross axis of their container | `As above` |
| `align-self` | Aligns flex items of current flex line overriding `align-items` – ignored if any item's cross-axis margin is `auto` | `auto, as above` |
| `order` | Specifies order used to lay out flex or grid item in flex or grid container | `<integer>` |

align-items: https://developer.mozilla.org/en-US/docs/Web/CSS/align-items

align-content: https://developer.mozilla.org/en-US/docs/Web/CSS/align-content

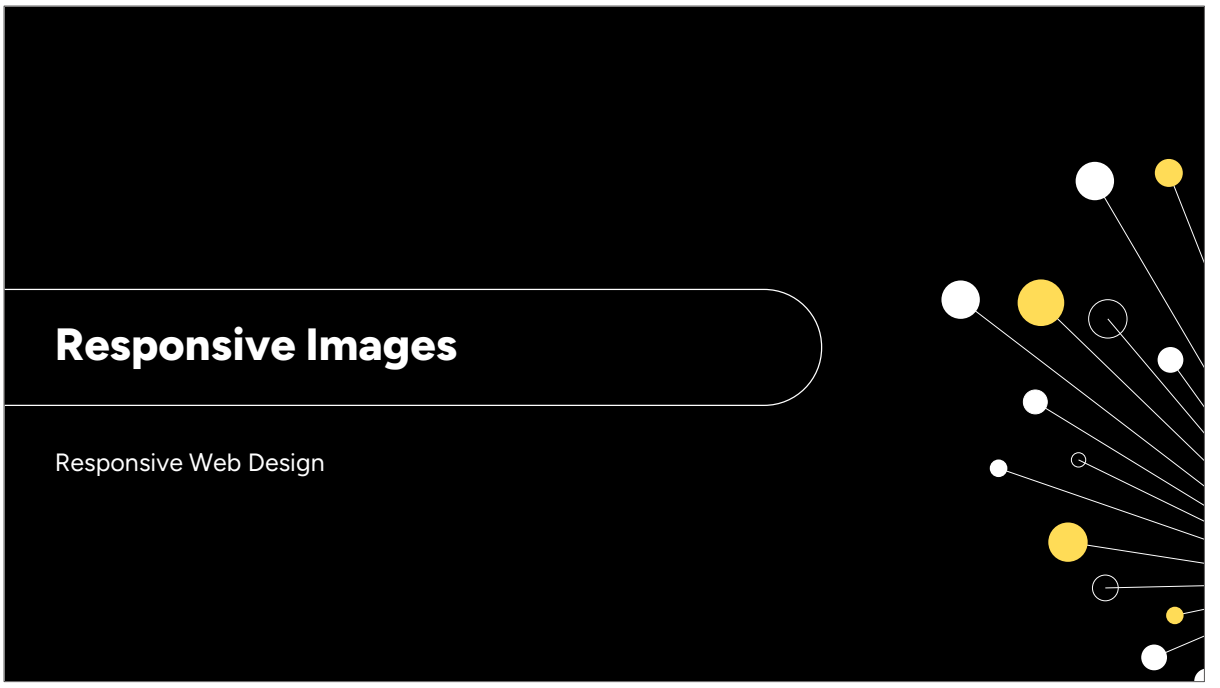
align-self: https://developer.mozilla.org/en-US/docs/Web/CSS/align-self

order: : https://developer.mozilla.org/en-US/docs/Web/CSS/order

QA

# Quick Lab 13 – Flex Box

Create a flex-container and experiment with flexbox attributes and settings.


Responsive Images
Responsive Web Design

QA

## What are Responsive Images

*A method for providing the browser with multiple image sources depending on display density, size of the image element in the page, or any number of other factors.*
*-- Jason Grigsby*

Responsive images refer to one or more methods for providing browsers with the correct image based on attributes of the user's device

- Such as display density, image and page size or any number of factors.

General principles for performance

- Avoid images wherever possible
- Use vector formats where possible: SVG or icon fonts
- Use the lowest possible resolution and quality
- Use the right format for the image type: WebP, PNG, JPEG

Before we jump into specifics we'll take a bird's eye view of how to use responsive images.

The goal here is to increase performance.

1. Avoid using images wherever possible. Before using an image, ask yourself whether the image is really necessary or useful. Maybe the same effect could be achieved with simple text and CSS. We can now do gradients and rounded corners and blended backgrounds directly with CSS rather than using images to accomplish the same effect.
2. Depending on the purpose and the type of image it may be better to use vector graphic formats instead of raster images like PNG, WebP, JPG or GIF. Vector graphics are usually smaller and easier to compress. Again, depending on the purpose, you may want to use icon fonts instead of raster graphics; there are some arguments against icon fonts, but they are beyond the scope of this presentation.
3. It makes no sense to send a 1MB image to a mobile device… it's important to use the lowest resolution and quality for each device you're trying to target.
4. Different image formats are better for different things:

1. GIF images are better for animations and images with a limited color palette (256 colors). You can also make animated GIFs
2. JPG images are better for photographs. Does not support transparencies
3. PNG images take the best of JPG and GIF. It supports same colors as JPG and also support transparencies like GIF
4. WebP images are generally smaller than equivalent JPG or PNG images. It also supports animations and some people have considered a replacement for animated GIFs

Which format you use will depend on your needs. As we'll discuss later we can have multiple formats in the same image delivered depending on browser support or the current size of the content.

QA

## The <picture> and <source> elements

**<picture>** and **<source>** elements enable us to provide alternative sources for the same resource

- Holds two different tags: one or more **<source>** tags and one **<img>** tag
- **<source>** element has the following attributes
  - **srcset** (required) - defines the URL of the image to show
  - **media** - accepts any valid media query that would normally be defined in a CSS
  - **sizes** - defines a single width descriptor
  - **type** - defines the MIME type

```
<picture>
   <source media="(min-width: 1024px)" srcset="kitten-large.png">
   <source media="(min-width: 667px)" srcset="kitten-medium.png">
   <img src="kitten-small.png" alt="Cute kitten">
</picture>
```

The picture and source elements enable us to provide alternative sources for the same resource.

The browser will stop and load the first source element that it understands and load that image. If the browser can't read the files specified in the source elements or if the browser doesn't support the picture and source elements the default image will be loaded.

The source elements can include different file formats. In this example we use WebP first and then JPG and provide the same JPG file as the default image in the img element.

QA

## The width descriptor

Sizes attribute

- Tells the browser the size or sizes of the element the **`srcset`** is attached to so that the browser can use the appropriate image
  - **`sizes="50vw"`**: telling the browser that the image will be displayed at 50% of the viewport width
  - **`w`** unit : width of each image in pixels, enabling the browser to choose the right image to retrieve, depending on the screen pixel density and the viewport size

```
<img src="small.jpg"
     srcset="kitten-small.png 500w,
          kitten-medium.png 1000w,
          kitten-large.png 1500w"
     sizes="50vw"
     alt="cute kitten"
/>
```

For a browser, there's a Catch-22 when it comes to choosing which image to download: the browser needs to know the dimensions of each image, but it can't know that without downloading each image to check.

Enter the w unit...

The w unit *tells* the browser the width of each image in pixels, thereby enabling the browser to choose the right image to retrieve – depending on the screen pixel density and the viewport size.

Note:
We can't specify both a pixel density and width descriptor in the same srcset. It must be all pixel densities or all width descriptors.

QA

# Combining Media Queries and srcset

Combining media queries and **`srcset`** to specify images for different viewports

- Also providing different images for different pixel densities
- Tools like *responsivebreakpoints.com* will generate the images and the corresponding code for you so you don't have to do

```
<picture>
    <source media="(min-width: 1024px)"
      srcset="kitten-large.png 1x, kitten-large_2x.png 2x">
    <source media="(min-width: 667px)"
      srcset="kitten-medium.png 1x, kitten-medium_2x.png 2x">
    <img src="kitten-small.png" alt="Cute kitten">
 </picture>
```

QA

# Quick Lab 14 - Responsive Images

Add images to HTML that respond to the device/screen that is being used to view them.

QA

## Learning Objectives

- Understand why mobile first and responsive web design
- Be able to apply responsive design principles
- Be able to use Media Queries
- Understand and implement grid systems
- Understand flexbox
- Be able to create and use responsive images

QA

# Hackathon – QA Cinemas

**You have a brief for a website to be set up using the skills, knowledge and understanding of HTML, CSS and RWD gained on the course**

- The brief has a number of user stories to tackle
- You are expected to:
  - Have a planning meeting to decide how the work will be distributed
  - Create the website
  - Have a review meeting to demonstrate the website to your trainer
  - Hold a retrospective to review what has been learnt as part of the process